EFFICIENT PARALLELIZATION OF NON-UNIFORM FAST MULTIPOLE ALGORITHMS

By

Stephen Michael Hughey

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical and Computer Engineering – Doctor of Philosophy
Computational Mathematics, Science, and Engineering – Dual Major

2018

ABSTRACT

EFFICIENT PARALLELIZATION OF NON-UNIFORM FAST MULTIPOLE ALGORITHMS

By

Stephen Michael Hughey

Many applications of the N-body problem today involve distributions of bodies that are (i) very large and (ii) highly non-uniform. A variety of fast multipole algorithms have been devised to reduce the cost from $O(N^2)$ to $O(N \log N)$ or $O(N)$ for oscillatory and non-oscillatory problems, respectively. The issue of non-uniformity, however, presents significant challenges in parallelization, requiring a much more nuanced approach. Compounding this challenge, oscillatory N-body problems arising from wave physics (electromagnetics, acoustics, etc.) are burdened with capturing both phase and amplitude information as opposed to just the amplitude; non-uniformity even further complicates things. As a result, the algorithm and underlying data structures become extremely complicated, and parallelization becomes quite difficult.

This thesis aims to develop novel parallel fast multipole methods for both oscillatory and non-oscillatory problems that (i) are controllably accurate to arbitrary precision, (ii) are capable of efficiently handling highly non-uniform distributions, and (iii) scale well up to extremely large problem sizes and numbers of CPU cores. The accelerated Cartesian expansion (ACE) method and wideband multilevel fast multipole algorithm (MLFMA) are modified to accurately and efficiently accommodate non-uniform, and in the case of MLFMA extremely large, distributions in parallel. Several parallel algorithms for efficiently building the distributed non-uniform tree data structures are developed. Effective, novel algorithms are introduced to reduce load imbalances arising from non-uniformity and certain idiosyncrasies of the parallel wideband MLFMA which hamper scalability. The algorithms presented here meet each of the stated goals, enabling computations involving several hundred million degrees of freedom on 2048 cores for an electromagnetics problem and several billion particles on 16,384 cores for non-oscillatory problems.

ACKNOWLEDGEMENTS

I am sincerely grateful for the many people who have so generously supported me in the course of my education and have contributed to my development as a researcher, as a professional, and as a person. From my family and friends I have received boundless love and encouragement, without which I could scarcely have succeeded. From my mentors, invaluable insights on matters ranging from the academic to the interpersonal that continue to prove useful. From my committee, sound advice and a better perspective on communicating with those outside my own area of research. My lab-mates' ready willingness to explore and discuss ideas helped strengthen my grasp of both electromagnetics and computational science. My advisers' patience, encouragement, support, advice, and seriousness have been central to all of my successes.

Finally, my fiancé has for years been an unflagging source of love, empathy, compassion, wisdom, and measured criticism whenever I needed any of these things. I cannot adequately express the depth of my gratitude to her in writing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

KEY TO ABBREVIATIONS

ACE Accelerated Cartesian expansion

CFIE Combined-field integral equation

FMM Fast multipole method

IE Integral equation

Matvec Matrix-vector multiplication

MLFMA Multilevel fast multipole algorithm

RCS Radar cross-section

CHAPTER 1

INTRODUCTION

1.1 N-body problems

The N-body problem refers to a common computational pattern arising in a significant number of fields in computational physics and related fields. Consider a collection of N particles (bodies) located at points $\mathbf{r}_i$, $i = 1, \ldots, N$ and with strengths u_i, $i = 1, \ldots, N$. Broadly speaking, the N-body problem may be defined in the following manner. Given the source strengths and locations $u_i, \mathbf{r}_i$ and kernel function ψ encoding action at a distance, the task is to compute

$$\Psi(\mathbf{r}_i) = \sum_{j=1}^{N} \psi(\mathbf{r}_i - \mathbf{r}_j) u_j, \quad i = 1, \ldots, N, \tag{1.1}$$

where the function Ψ is referred to as the potential function associated with the kernel ψ. While it is sometimes the case that the target locations (i.e., those points at which Ψ is evaluated) do not coincide with the source locations, this thesis assumes the source and target locations are always coincident. The principal bottleneck in evaluating (1.1) is the $O(N^2)$ computational complexity.

The N-body problem (1.1) appears in a number of applications too large to list here comprehensively; an abridged list will suffice to highlight its ubiquity. The N-body framework is used for gravitational force calculation in galaxy formation problems in computational astrophysics, electrostatic force calculation in molecular dynamics and physical electronics, kernel classification methods in machine learning, integral equation solvers in electromagnetics, acoustics, elastodynamics, fluid mechanics, and many more.

Given the importance and number of applications of this problem, it is not surprising that a significant amount of research effort has been expended to reduce the cost of evaluating (1.1), particularly in light of the success of the fast Fourier transforms developed in the 1960s. The algorithms of Appel [4] and Barnes and Hut [8] reduced the cost of (1.1) for gravitational force calculations from $O(N^2) \rightarrow O(N \log N)$. At the same time, Rokhlin and Greengard proposed the

elegant Fast Multipole Method (FMM) [59, 34] for the same problem, reducing the cost to the optimal $O(N)$.

1.2 Fast multipole methods (FMMs)

The FMM idea quickly gained traction in the computational physics community. Its optimal runtime, control over accuracy, and geometry adaptivity make it a highly attractive option for large-scale simulation. The idea was soon generalized for a number of other kernels, including Green's functions in elastodynamics [17], Stokes flow [31], and electromagnetics [21, 67]. The electromagnetic variant of the FMM, deemed the multilevel fast multipole algorithm (MLFMA), is different from the other methods listed here in a very fundamental way. The electromagnetic Green's function imparts a notion of phase to go along with amplitude information, as opposed to the amplitude-only FMM. Adequately resolving these oscillations requires satisfaction of a sampling theorem over a domain proportional to the size of the computational domain. As a result, the MLFMA's memory requirements are significantly higher than its non-oscillatory counterparts, and the algorithm scales as $O(N \log N)$ rather than $O(N)$.

Historically, these methods relied on the existence of an analytic factorization of the kernel function into a sum-product of functions that depend separately on the source and observer coordinate frames; however, the FMM has also been extended to a general kernel-independent [2, 47, 30] or almost kernel-independent [61] setting. The latter method, deemed the accelerated Cartesian expansion (ACE) method, is *almost* kernel-independent because, while it requires the derivatives of the kernel function, only one stage of the algorithm requires any information about the kernel, and the method can be applied to any non-oscillatory kernel.

The basic mechanism of acceleration is the separation of interactions into "near" and "far" field interactions classified by some separation criterion, and approximating the far-field interactions using intermediate bulk interactions between clusters of particles. In other words,

$$\Psi(\mathbf{r}) = \Psi^{NF}(\mathbf{r}) + \Psi^{FF}(\mathbf{r}), \tag{1.2}$$

where the superscripts denote the near- and far-field interaction potentials. Rather than classify each

point-to-point interaction directly by comparison of the interaction distance with some threshold, the simulation domain is broken up into a multi-level hierarchy of boxes organized into a tree data structure, and points are assigned to their containing box. Near- and far-field interactions are classified on a box-by-box basis on this grid using some number of near-field buffer boxes. The near-field interactions, i.e. those between points in neighboring boxes, are evaluated directly at $O(N)$ cost. Far-field interactions are typically approximated using some variant of the following:

1. For each box at the finest level of refinement, compute the *multipole expansion* representing the sources inside the box;

2. For each coarser-level box, form multipole expansions by shifting and combining those from the finer levels;

3. For each box, form *local expansions* of the observed field within by applying *translation operators* to multipole expansions of all far-field boxes and combining the results;

4. For each finer-level box, combine local expansion with a shifted local expansion of containing upper-level box;

5. Evaluate the field at each observation point by evaluating the local expansion within each box at the finest level.

The FMM, ACE, and MLFMA approximations are *error-controllable*, meaning that arbitrarily high accuracy can be achieved by increasing the order of approximation, at the expense of extra computational time.

1.3 Parallelization of FMMs

While these methods allow the fast evaluation of large N-body sums, the size of the problems that can be solved is limited by the computational resources of a single computer. Hence, parallelization of these algorithms is necessary for solving truly large-scale problems. However, parallelization is not at all straightforward. An efficient parallel algorithm relies on an equipartitioning of the

overall workload associated with the tree structure. In the FMM for non-oscillatory kernels, most of the work to be done lies at the finest level of refinement, or the *leaf* level, greatly simplifying the problem. In this case, a simple partitioning of the boxes at this level along a locality-preserving space-filling curve suffices to form the base of highly scalable parallel algorithms [64], though employing an adaptive form of the FMM complicates the parallelization.

In the MLFMA, however, the necessity of capturing both phase and amplitude information, as opposed to strictly amplitude in non-oscillatory FMMs, introduces significant challenges to parallelization. The MLFMA can be viewed as a decomposition of the kernel into a sum of plane waves traveling in all directions. The sampling theorem requires the number of plane waves in the sum to be proportional to the surface area of the sphere enclosing each collection of sources/observers, and so the information content per box increases as one ascends the tree. Consequently, in MLFMA simulations involving electrically large objects, not only is the cost per tree level approximately constant, but the storage and computational costs of nodes at the uppermost levels of the tree can become prohibitive without an intelligent parallelization strategy.

Work on parallelization of the MLFMA has continued for almost two decades. The basic technique common to the most successful approaches is that of partitioning over space and then over the plane waves at some chosen granularity [28, 48, 50]; however, as will be discussed in a later chapter, the necessity for accurate interpolation and downsampling in the upward and downward tree traversal stages of the *matvec*, respectively, has important consequences for the partitioning scheme. While some progress has been made toward a robust, scalable, and accurate MLFMA for solving large-scale problems in EM, there remains plenty of work to be done. In particular, efficient parallelization of adaptive wideband variants remains an open problem.

1.4 Thesis structure

The remainder of this thesis aims to put forth a set of algorithms advancing the state of the art in efficient parallel, non-uniform, fast multipole-like algorithms. We will concern ourselves first with non-oscillatory potentials. We will then turn our attention to the electromagnetic case.

Chapter 2 describes a scalable parallel ACE algorithm for evaluating linear operators acting on arbitrary non-oscillatory potentials. We first present an example of potentials defined in terms of linear operators acting on a common kernel function and then provide an algorithmic prescription for evaluating such potentials. Next, we present a suite of parallel algorithms for tree construction, load balancing, and potential evaluation. Finally, a set of numerical experiments demonstrates error control of the method for several kernel functions and good parallel scalability for problems as large as 5 billion particles on 16,384 processes.

Chapter 3 focuses on the solution of EM scattering problems using the non-uniform wideband MLFMA. We introduce methods which greatly increase the scale of EM problems that can be solved using the wideband MLFMA presented originally in [73]. We first present a numerical method for breaking a significant bottleneck of the original algorithm that limited the size of problems to which it could be applied. We then present an adaptive form of the algorithm which significantly improves performance for multiscale geometries, i.e. those with highly non-uniform spatial distributions of unknowns. We demonstrate the error control of these methods and performance of the parallel algorithm on thousands of processes for several electrically-large scatterers.

Chapter 4 focuses on the parallelization details of the non-uniform wideband MLFMA used in the previous chapter. We identify several computational bottlenecks and their causes in the original parallel wideband MLFMA presented in [48], upon which this work is based. We then propose and implement remedies to their resolution. We also present explicit algorithms for building the non-uniform tree in parallel, computing interaction lists in the non-uniform tree, and load balancing the potential evaluation stage for highly non-uniform distributions. The benefits of the proposed methods are then proven through a series of numerical experiments.

CHAPTER 2

PARALLEL ALGORITHM FOR FAST EVALUATION OF STOKES POTENTIALS

2.1 Introduction

A number of physical systems rely on the computation of pair potentials to analyze/understand the relevant physics. By pair potentials, we imply the means through which two entities interact, viz., the relevant Green's function that depends on relative position of the two particles (and possibly the relative times). Examples of such potentials are abundant. For instance, the Coulomb potential is used in a number of different fields ranging from gravitational physics to electro-statics to molecular dynamics to density functional theory and so on. Likewise, we see application of potentials such as Lennard-Jones, Yukawa, Buckingham, and so on in molecular dynamics, lattice potential in electronic structure calculations, Stokes potentials in low Reynolds number fluid flows, potentials arising from the Helmholtz equation, wave equation, heat equation, diffusion equation, Klein-Gordon equation, and more. Often, the pair potentials necessary are tensorial, obtained via action of linear differential operators on simpler potentials. A rather simple example is the force on a particle due to electrostatic/gravitational interactions. More generally, the pair potential can be expressed as $\mathcal{L}_\nabla \{\psi(\mathbf{r})\}$ where $\mathcal{L}_\nabla \{\cdot\}$ is a linear operator.

Evaluation of these potentials is expensive and scales as $O(N^2)$ where N is the number of degrees of freedom. Reduction of this cost complexity, from $O(N^2) \longrightarrow O(N^\alpha)$ with $\alpha < 2$ for evaluation of the Coulomb potential for gravitational physics was the origin of the classical N-body problem. The earliest efforts toward its resolution can be traced back to Barnes and Hut [8], which is the ancestor of a class of methods called tree algorithms. A slightly different approach, the fast multipole method (FMM), introduced by Rokhlin [59], Zhao [82], and then Greengard [32], has been celebrated as one of the top ten algorithms of the 20th century [25]. For volumetric distributions, tree algorithms and FMM reduce the computational costs from $O(N^2)$ to $O(N \log N)$ and $O(N)$, respectively. Despite similarities between the two methods, there are fundamental

differences between their asymptotic complexities; these have been elucidated in [32, 35, 61]. The principal difference between the two is in how one traverses the tree. The bottom line, however, is that both approaches effectively approximate the pair potential in terms of low-order polynomials and their error bounds are well understood.

While the most extensive body of work on N-body problems has been on Coulomb potentials [34, 35, 18, 24], there has been significant interest and effort in other areas as well. These include Yukawa [13, 38, 72], Gauss [36, 77], radial basis functions [43, 30], lattice gas [61], Klein-Gordon and diffusion [74], Helmholtz [21, 66], retarded [62, 72], and periodization of some of these [7, 16, 44]. In general, methods used for potentials arising from the Helmholtz and wave equations need to be treated differently [21, 27] than the others due to the need to capture phase information as well as amplitude. But a unifying theme to the methods developed for all other potentials is representation of these in an observation domain in terms of low order polynomials. This can be effected either using Taylor series (or an optimal variant in the case of FMMs) [34, 82], kernel independent FMMs [79], or the black-box approaches [30].

A fast multipole-like algorithm that uses Cartesian tensors in an optimal manner (hence called accelerated Cartesian expansions, or ACE) was introduced in 2007 [61] for potentials of the form $r^{-\nu}$. The salient features of the ACE algorithm are (a) the ability to accelerate smooth, non-oscillatory potentials, $\psi(\mathbf{r})$, of *arbitrary form* and can be extended to linear operators acting on ψ, or $\mathcal{L}_\nabla\{\psi(\mathbf{r})\}$, with minimal change in cost; (b) *exact* traversal up and down the tree in that if the multipole expansion were to be computed at a given level directly from the actual particles, it would be identical –to machine precision– to that computed from its grand-children; (c) mapping for traversal up the tree from level $l \longrightarrow l+1$ and $l+1 \longrightarrow l+2$ differ only by a multiplicative constant and is independent of the potential $\psi(\mathbf{r})$; (d) the same is true for traversing down the tree, as well as across if $\psi(\mathbf{r}) = |\mathbf{r}|^{-\nu}$, $\forall \nu \in \mathbb{R}$. The complexity of ACE scales as $O(\alpha N P^6)$ (and can be reduced trivially to $O(\alpha N P^5)$) where P is the order of representation and $\alpha = 1/(720s)$. Here, s is the average number of particles in a leaf box and can be tuned to optimize the cost. There also exits a specialization of ACE for r^{-1} using *traceless* Cartesian tensors that (a) can

be mapped directly onto the classical fast multipole algorithm, and (b) has a complexity that scales as $O(\beta NP^4)$ with $\beta = 1/(2s)$. This drew upon the relationship between traceless Cartesian tensors and spherical harmonics. The ACE algorithm has been extended to Yukawa, low-frequency Helmholtz, diffusion, Klein-Gordon, dispersive and low-frequency retarded potentials (and some of their periodic variations) [74, 6]. Given these features of ACE, in this paper we will build upon this framework to accelerate the evaluation of non-oscillatory potentials defined by linear operators.

Since acceleration methods are commonly used for the Coulomb potential, the effort spent on developing effective parallel algorithms has been equally impressive; see [33, 11, 63, 76, 56] and references therein. More recently, the emergence of kernel independent FMM and potential applications of non-oscillatory potentials to PDEs alluded to earlier in this section as well as to machine learning [14, 10] has resulted in a body of work on parallelization that is most current and pertinent [79, 45, 1, 47]. These methods typically employ a local essential tree (LET) structure, which compartmentalizes communication and computation, allowing the overlapping of communication with computations from different stages of the FMM algorithm. However, this approach requires redundant computation and storage of remote source data. A different parallel algorithm that uses a bottom-up partitioning of the tree based on a post-order traversal sequence and eliminates the need for redundant computations at the higher levels of the tree has been reported by Melapudi *et al.* [75, 48]. As demonstrated through the evaluation of multiple pair potentials in a particle dynamics simulation using the ACE algorithm, the implementation by Melapudi *et al.* exhibits excellent strong scaling properties.

In this paper, our goal is to develop parallel algorithms to evaluate $\psi(\mathbf{r})$ and $\mathcal{L}_\nabla\{\psi(\mathbf{r})\}$ efficiently. A few examples of potentials of this form are the set of Stokes potential (Stokeslet, rotlet and stresslet) [42], force or field calculations in electrostatics/gravitation, as well as those encountered in low frequency electromagnetics wherein one transitions across scales (quantum to classical) [3]. Given the landscape of both fast methods and the attendant parallelization algorithms mentioned above, it is apparent that each has its (dis)advantages. However, in keeping with our goal, ACE is an ideal framework as will be demonstrated through both mathematical and numerical analyses. To

this end, the main contributions of this paper are (a) development of a modified ACE framework to enable evaluation of $\mathcal{L}\{\psi(\mathbf{r})\}$ with minimal cost overhead compared to that for $\psi(\mathbf{r})$, (b) an efficient parallel implementation of ACE for highly non-uniform particle distributions, and (c) demonstration of the efficiency and scalability of the described methods for several different potentials.

The rest of this paper is organized as follows: In the next section, we define the problem addressed by this work and state the Stokes kernels in terms of an $\mathcal{L}_\nabla$-operation. Section 2.3 reviews the ACE algorithm and introduces the necessary modifications to effect such operators for the Stokes potentials, and presents a complexity estimate for the modified ACE algorithm. Section 2.4 describes our parallel algorithm in detail, including tree construction, interaction lists, load balancing, and parallel potential evaluation. Finally, in Section 2.5, we present a number of numerical results demonstrating the accuracy and efficiency of our algorithm.

2.2 Problem statement

Consider a collection of N sources distributed randomly in $\mathbb{R}^3$, described by $u(\mathbf{r}) = \sum_j^N u_j \delta(\mathbf{r} - \mathbf{r}_j)$. The effect of these sources at a point $\mathbf{r}$ can be written as

$$\Phi(\mathbf{r}) = \psi(r) \star_s u(\mathbf{r}), \tag{2.1}$$

where $r = |\mathbf{r}|$, and $\star_s$ represents a spatial convolution. In several applications, one needs to compute

$$\mathbf{\Phi}^{(n)}(\mathbf{r}) = \mathcal{L}_\nabla\{\psi(|\mathbf{r}|)\} \star_s u(\mathbf{r}) \tag{2.2}$$

where the superscript (n) denotes a tensor of rank n as required by the operator $\mathcal{L}_\nabla$. In the exposition, we have restricted $u(\mathbf{r})$ to a scalar source. It is, however, trivial to generalize ideas presented here to vector sources.

In what follows, we will illustrate our ideas in the context of Stokes kernels. These potentials can be expressed as a set of linear operators acting on non-oscillatory pair potentials. Stokes potentials arise in boundary integral formulations in fluid dynamics for low Reynolds numbers, and are denoted by the Stokeslet $\mathbf{S}^{(n)}$, rotlet $\mathbf{\Omega}^{(n)}$, and the stresslet $\sigma^{(n)}$ kernels which, respectively, can

be expressed in index notation as

$$S_{ij}(\mathbf{r}) = \frac{\delta ij}{r} - \frac{r_i r_j}{r^3}, \tag{2.3a}$$

$$\Omega_{ij}(\mathbf{r}) = \epsilon_{ijk}\frac{r_k}{r^3}, \tag{2.3b}$$

$$\sigma_{ijk}(\mathbf{r}) = -6\frac{r_i r_j r_k}{r^5} \tag{2.3c}$$

Alternatively, the above potentials can be rearranged and expressed in terms of derivatives on r [57, 29, 42]:

$$S_{ij}(\mathbf{r}) \doteq \left(\delta_{ij}\nabla^2 - \partial_i\partial_j\right)\psi(\mathbf{r}), \tag{2.4a}$$

$$\Omega_{ij}(\mathbf{r}) \doteq \left(-\epsilon_{ijk}\partial_k\nabla^2\right)\psi(\mathbf{r}), \tag{2.4b}$$

$$\sigma_{ijk}(\mathbf{r}) \doteq \left[\left(\delta_{ij}\partial_k + \delta_{jk}\partial_i + \delta_{ki}\partial_j\right)\nabla^2 - 2\partial_i\partial_j\partial_k\right]\psi(\mathbf{r}) \tag{2.4c}$$

where $\psi(\mathbf{r}) = r$, δ_{ij} is the Kronecker delta, ∇^2 denotes the Laplacian, ∂_i represents the derivative along the i^{th} direction (from the set $\{x, y, z\}$), and ϵ_{ijk} denotes the Levi-Civita symbol. From these equations, one can arrive at the appropriate $\mathcal{L}_\nabla\{\cdot\}$ for each of the tensors.

Assuming N particles, the cost of evaluating (2.2) scales as $O(3^n N^2)$. In what follows, we introduce modifications to the ACE algorithm which bring down the complexity of evaluating all 3^n components down to $O(NP^6)$.

2.3 Generalization of ACE to $\mathcal{L}_\nabla\{\psi(\mathbf{r})\}$

In this section, we discuss the overall computational framework, provide a brief summary of the ACE algorithm, present the modification that permit efficient computation of $\mathcal{L}_\nabla\{\psi(\mathbf{r})\}$ using ACE, and conclude with a computational complexity analysis.

2.3.1 Framework for fast N-body computations

To begin, let us assume that the entire computational domain $\Omega \in \mathbb{R}^3$ can be embedded within a cube. All FMM-like fast algorithms start with building an oct-tree data structure by hierarchically partitioning the cube encompassing the entire computational domain. At each level, the cube is

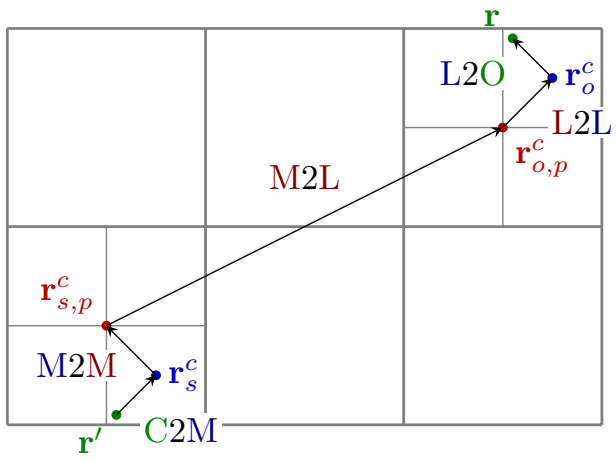


Figure 2.1: Illustration of the operations in the ACE algorithm and notation used in this section.

subdivided into eight equi-sized sub-cubes, and this process continues recursively until the desired level of refinement (e.g., defined by a threshold on the number of sources/observers contained within each sub-cube) is reached; an L-level scheme implies $L-1$ recursive divisions of the domain. At any level, the (sub)domain that is being partitioned is called the parent of all the eight children that it is being partitioned into. At the lowest level, all sources/observers are mapped onto the leaf boxes. Interactions between all source/observer pairs are now computed using the tree structure. Specifically, near and far fields are identified for all boxes/subdomains at any level in the tree according to the following criteria: Two subdomains are classified as being in the near field of each other, if they share a common corner, edge or surface. Otherwise two subdomains are in the far field of each other, if the distance between their centers is at least twice the side length of the subdomain, and their parents are in the near field of each other.

Once the interaction lists are built for all levels, the computation proceeds as follows. At the leaf levels, interactions between elements of boxes that are in the near field of each other are computed directly, *i.e.*, using $\psi(\mathbf{r})$. Far field interactions are computed using a three stage algorithm: (i) compute multipoles of sources that reside in each box (C2M); (ii) convert these to local expansion at all boxes that are in its far field (M2L); (iii) from the local expansion, compute the field at each observer (L2O). It is apparent that one can cut down computational costs by embedding this scheme within itself. That is, if two domains interacting with each other are far away, then these clusters

may be combined to form larger clusters that then interact with each other at a higher level and so on. As has been shown elsewhere [34, 61], this computational strategy considerably mitigates the overall cost. Additional theorems exist that enable (a) shifting the origins of multipoles so that effects of small clusters can be grouped together to form larger clusters (M2M) and (b) move the origin of local expansion so that expansions at the origin of the parent may be disaggregated to those of its children (L2L). The overall computational scheme is illustrated in Fig. 2.1 and proceeds as follows; (c) compute nearfield interactions at the leaf levels, and traverse the tree (C2M$\longrightarrow$M2M$\longrightarrow$M2L$\longrightarrow$L2L$\longrightarrow$L2O) as appropriate to compute the remaining interactions. Theorems to effect the tree traversal are summarized in the remainder of this section.

2.3.2 Brief background on ACE

To begin describing ACE, consider two boxes Ω_s and Ω_o that contain source and observers, respectively, in the oct-tree decomposition of Ω, and let the parents of Ω_s and Ω_o be denoted using Ω_s^p and Ω_o^p. Centers of these boxes are denoted using $\mathbf{r}_s^c$, $\mathbf{r}_o^c$, $\mathbf{r}_{s,p}^c$, and $\mathbf{r}_{o,p}^c$. To set the stage for the relevant descriptions, we start with some remarks and notations; (i) an n-th rank totally symmetric tensor $\mathbf{A}^{(n)}$ contains $(n+1)(n+2)/2$ independent components as opposed to 3^n components; (ii) in compressed form this tensor can be represented using $A^{(n)}(n_1, n_2, n_3)$ where $n = n_1 + n_2 + n_3$; (iii) an example of such a tensor is the polyadic associated with $\mathbf{r}$ which is given by $\underbrace{\mathbf{rr}\cdots\mathbf{rr}}_{n \text{ times}} = \mathbf{r}^n$ and can be represented in compressed form as $r^{(n)} = x^{n_1} y^{n_2} z^{n_3}$; (iv) an m-fold contractions between two tensors $\mathbf{A}^{(n)}$ and $\mathbf{B}^{(m)}$ is denoted using $\mathbf{A}^{(n)} \cdot m \cdot \mathbf{B}^{(m)} = \mathbf{C}^{(n-m)}$; and (v) a direct product between two tensors can be written as $\mathbf{C}^{(n+m)} = \mathbf{A}^{(n)}\mathbf{B}^{(m)}$.

The foundation of ACE is based on a Taylor series expansion which provides a natural framework for developing addition theorems. A Taylor series expansion for $\psi(|\mathbf{r} - \mathbf{r}'|)$ about the origin,

$$\psi(\mathbf{r} - \mathbf{r}') = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \left(\mathbf{r}'\right)^n \cdot n \cdot \nabla^n \psi(\mathbf{r}), \tag{2.5}$$

for $|\mathbf{r}| > |\mathbf{r}'|$, results in a series of theorems summarized next; this is provided only for completeness, and proofs for all can be found in [61].

Theorem 1 (Charge-to-multipole (C2M))**.** *Assuming that k sources exist in the domain Ω_s, the potential function $\Phi(\mathbf{r})$ at any point $\mathbf{r}$ significantly away from Ω_s is given by*

$$\begin{aligned}\Phi(\mathbf{r}) &= \sum_{n=0}^{\infty} \mathbf{M}^{(n)}(\mathbf{r}_s^c) \cdot n \cdot \nabla^n \psi(\mathbf{r}) \\ \mathbf{M}^{(n)}(\mathbf{r}_s^c) &= \sum_{i=1}^{k} \frac{(-1)^n}{n!} (\mathbf{r}_i - \mathbf{r}_s^c)^n u_i,\end{aligned} \tag{2.6}$$

where $\mathbf{M}^{(n)}(\mathbf{r}_s^c)$ is the rank-n multipole tensor about $\mathbf{r}_s^c$.

Next, these multipoles can be re-expressed about $\mathbf{r}_{s,p}^c$ using

Theorem 2 (Multipole-to-multipole (M2M))**.** *The multipole expansion $\mathbf{M}(\mathbf{r}_s^c)$ may be re-centered about the center $\mathbf{r}_{s,p}^c$ of a domain $\Omega_s^p \supset \Omega_s$ via*

$$\mathbf{M}_p^{(n)}(\mathbf{r}_{s,p}^c) = \sum_{m=0}^{n} \sum_{P(n,m)} \frac{m!}{n!} (\mathbf{r}_{s,p}^c - \mathbf{r}_s^c)^{n-m} \mathbf{M}^{(m)}(\mathbf{r}_s^c), \tag{2.7}$$

where $P(n, m)$ is the permutation of all partitions of n into sets of size $n - m$ and m, and $\mathbf{M}_p(\mathbf{r}_{s,p}^c)$ is the re-centered multipole expansion.

Next, we translate these multipoles about Ω_s^p to Ω_o^p.

Theorem 3 (Multipole-to-local (M2L))**.** *Assume that a domain Ω_o^p centered at $\mathbf{r}_{o,p}^c$ exists at some distance from Ω_s^p, and $\Omega_o^p \cap \Omega_s^p = \emptyset$. Then, given a multipole expansion $\mathbf{M}_p^{(n)}(\mathbf{r}_{s,p}^c)$ centered at $\mathbf{r}_{s,p}^c$, the potential $\Phi(\mathbf{r})$ may be alternatively expressed in the form*

$$\begin{aligned}\Phi(\mathbf{r}) &= \sum_{n=0}^{\infty} (\mathbf{r} - \mathbf{r}_{o,p}^c)^n \cdot n \cdot \mathbf{L}_p^{(n)}(\mathbf{r}_{o,p}^c), \\ \mathbf{L}_p^{(n)}(\mathbf{r}_{o,p}^c) &= \frac{1}{n!} \sum_{m=n}^{\infty} \mathbf{M}_p^{(m-n)}(\mathbf{r}_{s,p}^c) \cdot (m - n) \cdot \nabla^m \psi(|\mathbf{r}_{o,p}^c - \mathbf{r}_{s,p}^c|),\end{aligned} \tag{2.8}$$

where $\mathbf{L}_p(\mathbf{r}_{o,p}^c)$ is referred to as the local expansion for Ω_o^p.

Next, we map the local expansions in Ω_o^p to Ω_o.

Theorem 4 (Local-to-local (L2L))**.** *Given a local expansion* $\mathbf{L}^{(n)}(\mathbf{r}^c_{o,p})$ *within the domain* Ω^p_o *about the center* $\mathbf{r}^c_{o,p}$*, the local expansion within* $\Omega_o \subset \Omega^p_o$ *centered at* $\mathbf{r}^c_o$ *is given by*

$$\mathbf{L}^{(n)}(\mathbf{r}^c_o) = \sum_{m=n}^{\infty} \begin{pmatrix} m \\ m-n \end{pmatrix} \mathbf{L}^{(m)}(\mathbf{r}^c_{o,p}) \cdot (m-n) \cdot (\mathbf{r}^c_o - \mathbf{r}^c_{o,p})^{m-n}. \tag{2.9}$$

Finally, the local expansions are mapped to observers.

Theorem 5 (Local-to-observer (L2O))**.** *The potential at a point* $\mathbf{r} \in \Omega_o$ *can be obtained from the local expansion within* Ω_o *centered at* $\mathbf{r}^c_o$ *via*

$$\Phi(\mathbf{r}) = \sum_{n=0}^{\infty} (\mathbf{r} - \mathbf{r}^c_o)^n \cdot n \cdot \mathbf{L}^{(n)}(\mathbf{r}^c_o). \tag{2.10}$$

Given these theorems, we next specialize it to Stokes potentials.

2.3.3 Modifications for evaluating $\mathbf{\Phi}^{(n)}(\mathbf{r})$

Thus far, we have developed a method to evaluate $\Phi(\mathbf{r})$. Examination of (2.2) provides a methodology to evaluate the $\mathbf{\Phi}^{(n)}(\mathbf{r})$ as a minor modification of that for $\Phi(\mathbf{r})$; given Thm. 5, it follows that one can write

$$\begin{aligned} \mathbf{\Phi}^{(n)}(\mathbf{r}) &= \mathcal{L}_\nabla \{\psi(\mathbf{r})\} \\ &= \sum_{n=0}^{\infty} \mathcal{L}_\nabla \left\{(\mathbf{r} - \mathbf{r}^c_o)^n\right\} \cdot n \cdot \mathbf{L}^{(n)}(\mathbf{r}^c_o) \end{aligned} \tag{2.11}$$

This implies that the only change (both procedurally as well as in the computational cost) arises from a change in the L2O stage. Specifically, the Stokes potentials can be written as

$$S_{ij}(\mathbf{r}) * u(\mathbf{r}) = \sum_{n=0}^{\infty} \left[(\delta_{ij}\nabla^2 - \partial^2_{ij})(\mathbf{r} - \mathbf{r}^c_o)^n \right] \cdot n \cdot \mathbf{L}^{(n)} \tag{2.12a}$$

$$\Omega_{ij}(\mathbf{r}) * u(\mathbf{r}) = \sum_{n=0}^{\infty} \left[\left(-\epsilon_{ijk}\partial_k \nabla^2 \right) (\mathbf{r} - \mathbf{r}^c_o)^n \right] \cdot n \cdot \mathbf{L}^{(n)} \tag{2.12b}$$

$$(\sigma)_{ijk}(\mathbf{r}) * u(\mathbf{r}) = \sum_{n=0}^{\infty} \left[\left(\left(\delta_{ij}\partial_k + \delta_{jk}\partial_i + \delta_{ki}\partial_j \right) \nabla^2 - 2\partial^3_{ijk} \right) (\mathbf{r} - \mathbf{r}^c_o)^n \right] \cdot n \cdot \mathbf{L}^{(n)}_k. \tag{2.12c}$$

For each of the Stokes potentials, the above operation requires contracting a local expansion of rank n with another of rank $(n+2)$, $(n+2)$, and $(n+3)$ for the Stokeslet, rotlet, and stresslet, respectively. Furthermore, examination of these equations alongside (2.12) suggests that we can precompute and store the 6+10=16 unique second and third derivatives of the tensor $(\mathbf{r} - \mathbf{r}_o^c)^n$, combining them as needed to construct each disaggregation operator (the bracketed terms in (2.12)).

Furthermore, if the source $u(\mathbf{r})$ excites all three potentials, the simple mapping between $\psi(\mathbf{r}) = r$ and Stokes potentials enables significant optimization. That is, only three tree traversals are required to construct the 19 independent components of the Stokeslet, rotlet, and stresslet collectively at the observer. Finally, it is easily shown that the error in the evaluated potentials is commensurate with the representation polynomial order; the error bounds for this approximation follow from what was derived for ACE in [61].

2.3.4 Cost of evaluating $\mathcal{L}_\nabla \{\Phi\}$

The cost of computing the action of $\mathcal{L}_\nabla \{\Phi(\mathbf{r})\}$ by taking derivatives on the disaggregation tensors $\mathbf{r}^n, n = 0, \ldots, P$ is the same as that of simply computing $\Phi(\mathbf{r})$. To show this, we invoke Theorem 2.4 of [61] to give us

$$\nabla^k \Phi(\mathbf{r}) = \sum_{n=k}^{\infty} \frac{n!}{(n-k)!} \mathbf{L}^{(n)} \cdot (n-k) \cdot (\mathbf{r} - \mathbf{r}_o^c)^{n-k}, \qquad k \leq n. \tag{2.13}$$

Truncating at P terms to make use of all the information contained in $\mathbf{L}$ and evaluating the sum clearly requires at most the same amount of work as evaluation of (2.10) truncated at the same number of terms P. The resulting rank-k tensor contains all derivatives of degree k, from which the elements of $\mathcal{L}_\nabla \Phi$ are assembled. It follows that the cost of evaluating *all* components of the tensor $\mathbf{\Phi}^{(n)}$ is not significantly different from evaluating Φ by itself and scales as $O(NP^6)$.

2.4 Parallel ACE algorithm

In this section, we outline our parallel algorithms for constructing, adapting, and load balancing the tree, as well as parallel potential evaluation. While parallelization of FMMs is well-documented

in the literature [76, 80], as we will demonstrate in Sect. 3.5, redundant M2L computations in the commonly used locally essential tree (LET) based algorithms would significantly hamper the scaling of our proposed framework. For this reason, our parallel evaluation strategy differs from the usual LET based implementations, and therefore we present our parallel implementation in detail. In the ensuing discussion, let L denote the number of levels in the octree as determined by the parameter d_0, which represents the edge length of the smallest boxes in the tree. Level L denotes the level of finest refinement, *i.e.*, where boxes are of diameter d_0, while level 1 denotes the root, or the box containing the entire computational domain.

2.4.1 Construction of the distributed octree

Construction of the distributed octree starts by partitioning the input particles equally among all processes. On each process, the equidistributed particles are first hashed into Morton keys, which is a binary encoding of the leaf boxes according to their space-filling Morton-Z curve ordering. A parallel bucket sort is then used to assign each process a distinct set of leaves contiguous in the Morton-Z ordering such that each process gets a roughly equal number of particles. This initial partitioning is performed at the finest refinement level, *i.e.*, at level L, in preparation for the ensuing adaptive tree coarsening and load balancing stages, and is concluded with a local postorder traversal tree construction on each process from leaves all the way up to the root.

The above partitioning scheme necessarily incurs duplicate copies of internal tree nodes, corresponding to common ancestors of leaf nodes residing on different processes. Such nodes are referred to as *plural* nodes. Treatment of computations associated with plural nodes is one aspect of our parallelization scheme that differentiates it from the LET based implementations. Hence, we provide some terminology to aid the discussion. For any plural node, the process with the highest rank which owns a copy is designated as the *resident* process, and as such is deemed responsible for all its tree-based interactions. Other processes with a copy of the plural node are called *users* of the node, and are directed only to send and receive data to and from the resident process. A plural node is referred to as a *shared* node on its resident process, and the users' copies are called

duplicate nodes. As a consequence of the postorder traversal tree construction, it can be shown that all duplicate nodes in a process' local subtree appear consecutively at the end of the post-order traversal sequence. Shared nodes may appear anywhere within the post-order sequence, though they tend to appear toward the beginning. As we will discuss, these properties allow effective overlapping of communication and computation in the potential evaluation stage.

2.4.2 Adaptive tree

Random (or homogeneous) distributions of source or observer particles are well-represented by a uniform tree structure. Within such distributions, the number of particles per box is approximately constant across the entire simulation domain, facilitating the linear scaling of the ACE algorithm or any other FMM implementation for that matter. For non-uniform distributions though, *i.e.*, those with subregions of relatively high particle density, the minimum box size must be decreased to prevent near-field costs from dominating the computation. Hence, in a uniform tree framework, the densest region of discretization dictates the leaf box size throughout the *entire* tree. The result is large swaths of space with a poor work-to-particle ratio in sparsely populated regions, degrading the cost scaling of parallel ACE computations.

To prevent such inefficiencies, we adapt the tree structure to the particle distribution so as to restore the approximate uniformity in leaf box population throughout the tree. Adaptation of trees for N-body simulation has been well-studied and has become a standard feature of modern tree-based simulation methods for non-oscillatory kernels [63, 64, 68, 47]. We employ a bottom-up scheme for both constructing and merging the tree to minimize communication costs. Our parallel algorithm for merging the distributed octree is given in Algorithm 1. Simply put, this algorithm starts with a uniform tree constructed according to the description in Sect 2.4.1 and merges each set of sibling leaf boxes into their parents, if the total number of particles in these leaf boxes do not exceed a pre-determined particle threshold M. This scheme ensures that leaf boxes in more sparsely-populated regions contain roughly as many points as those in the dense regions, thereby avoiding excessive tree interactions (in sparse regions) or expensive near field computations (in

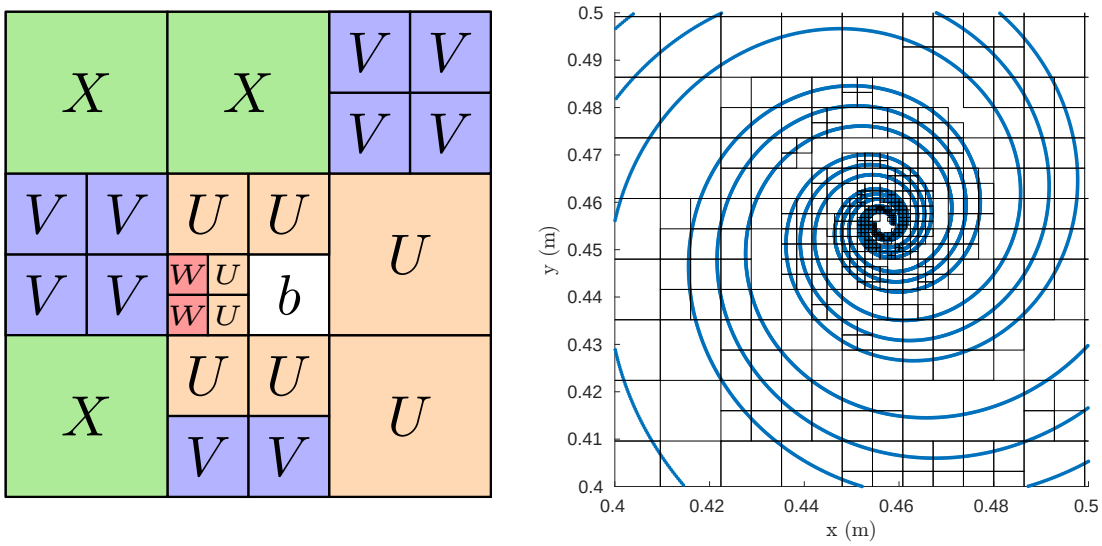
dense regions).

Algorithm 1: Parallel algorithm for merging the tree with the option to impose 2:1 balance constraint

Define:
$\mathcal{T}_{loc}$: local subtree (including plural nodes)
$\mathcal{T}_{loc}^{\ell}$: local subtree at level ℓ
M: maximum number of points per leaf box
$S(b)$: set of siblings of a box b
$n(b)$: number of points contained within the *complete* subtree rooted at box b
$\mathcal{N}(b)$: set of box b's near-neighbors at the same level
Votes: "1": vote to merge; "0": indeterminate; "−1": veto
`Votes`$\leftarrow 0$
for $\ell = L, L-1, \ldots, 4$ **do**
  $C \leftarrow \emptyset$
  for each $b \in \mathcal{T}_{loc}^{\ell}$ **do**
    if `Votes`(b)==0 **then**
      if $n(P(b)) \leq M$ **then**
        `Votes`$(S(b)) \leftarrow 1$
      else
        `Votes`$(S(b)) \leftarrow -1$
      end if
    end if
    if `Votes`$(b) \leq 0$ **and** 2:1 balance desired **then**
      $C \leftarrow C \cup \mathcal{N}(P(b))$
    end if
  end for
  `Votes`$(S(C)) \leftarrow -1$ {Global step; involves communication of vetoes to resident processes of boxes in C}
  Synchronize votes of level ℓ plural nodes and their siblings; vetoes dominate
end for
Prune each $b \in \mathcal{T}_{loc}$ and re-assign particles accordingly

2.4.2.1 Evaluation of ACE interactions in adaptive trees

In adaptive octree algorithms, it is necessary to evaluate interactions between boxes of different sizes. As usual, these interactions are classified into U, V, W, X-lists based on the adjacency and relative size of source and observer box pairs [15, 32]. For the sake of completeness, these lists are defined as follows:



(a) (b)

Figure 2.2: (a) Graphical illustration of the interaction lists in the adaptive tree for a leaf box b; (b) example of non-uniform distribution with non-uniform tree in two dimensions.

- **U-list**: source and observer boxes are adjacent and are both leaves, *i.e.*, they are within the near field of each other;
- **V-list**: source and observer boxes are not adjacent but their parents are, *i.e.*, they are inside the far field of each other;
- **X-list**: source and observer boxes are not adjacent, but the observer box is (i) adjacent to the parent of the source box, (ii) a leaf, and (iii) larger than the source box, corresponding to a cross-level interaction with an observer node higher up in the octree;
- **W-list**: reciprocal interactions of those in the X-list, corresponding to a cross-level interaction with an observer node lower down in the octree.

These interaction lists are illustrated in Fig. 2.2a. U-list interactions are handled by directly computing the interactions between particles contained within each box pair. The V-list interactions correspond to the M2L operations defined by (3). The X- and W-list interactions, however, require special treatment. Consider the evaluation of a potential function ϕ with kernel ψ using ACE for an X-list interaction between two boxes Ω_o, Ω_s, centered at $\mathbf{r}_o, \mathbf{r}_s$ respectively. If the diameter of

the sphere enclosing Ω_o is greater than the diameter of that for Ω_s, the local expansion about $\mathbf{r}_o$ formed from the source multipole expansion $\mathbf{M}^{(n)}(\mathbf{r}_s)$ is not valid everywhere inside Ω_o. However, $\mathbf{M}^{(n)}(\mathbf{r}_s)$ can be used to construct a different representation that is valid in this region. Consequently, the contribution $\phi_X(\mathbf{r})$ to the potential $\phi(\mathbf{r})$ observed at a point $\mathbf{r}$ within the observer box is given by direct evaluation of the Taylor series. For the Stokes potentials, the derivative on the observer coordinate frame in L2O moves directly onto the kernel function K. For integer derivative order d and truncation number P, it is clear that

$$\nabla^d \phi_X(\mathbf{r}) = \sum_{n=0}^{P} \mathbf{M}^{(n)}(\mathbf{r}_s) \cdot n \cdot \nabla^{n+d} \psi(\mathbf{r} - \mathbf{r}_s) \tag{2.14}$$

yields a symmetric rank-d tensor whose entries contain the full complement of derivatives of order d. All three Stokes potentials can be constructed using $d = 2, 3$ for each particle within Ω_o.

The W-list interactions have a somewhat simpler implementation. Here, the multipole expansion of the source box Ω_s is not quickly convergent anywhere within the observer box Ω_o, and therefore a local expansion must be formed about $\mathbf{r}_o$ directly from the particles contained in Ω_s. Viewing the Taylor series from the perspective of the observer coordinate frame, the contribution ϕ_W of the W-list sources to ϕ at the point $\mathbf{r} \in \Omega_o$ is given by

$$\phi_W(\mathbf{r}) = \sum_{n=0}^{P} (\mathbf{r} - \mathbf{r}_o)^n \cdot n \cdot \sum_{i=1}^{n_s} \nabla^n \psi(\mathbf{r}_o - \mathbf{r}'_i), \tag{2.15}$$

where $\mathbf{r}'_i$ is the location of the ith of n_s particles in the source box. The inner summation constitutes the local expansion about $\mathbf{r}_o$ of the sources within Ω_s. Derivatives for constructing the Stokes potentials may then be taken in the usual manner in the L2O stage.

2.4.2.2 2:1 balance constraint

For highly non-uniform distributions, unconstrained merging of the tree results in a tree with a matching degree of non-uniformity. While this is normally desirable from the perspective of reducing computational costs (less tree nodes means less number of interactions overall), it has a downside in terms of memory utilization. In a uniform tree where no X-list or W-list interactions

exist, there can be at most 6^3 (total number of boxes in the neighborhood of a box) - 3^3 (number of near field boxes) = 189 boxes in the far field of a box. Leveraging symmetries and scale invariance of certain kernels can significantly reduce the number of translation operators needed for V-list interactions. However, in regions containing sharp discontinuities in leaf box size, the number of X- and W-list interactions can become quite large, and the number of unique translation operators needed to perform the corresponding X- and W-list interactions can also be very large. The *2:1 balance constraint* [68, 47] remedies this storage problem by disallowing adjacent leaf boxes to differ in size by more than a factor of two, significantly reducing the number of different X and W interactions and the memory overhead for storing their particle-specific translation operators. Algorithm 1 for merging the tree includes the option to impose this constraint.

2.4.3 Load balancing

After the tree merging procedure under the 2:1 balance constraints is complete, each process is left with a compressed representation of the original uniform tree. As such, the computational profile is sufficiently different from the original tree that re-balancing the computational load on processes is necessary. To accomplish this, we follow an empirical load balancing strategy where we estimate the work attendant to each node in the distributed tree and aim to assign them in a load-balanced manner.

The cost per leaf is determined as follows. As part of the initialization operations, we first time a set of dummy operations to obtain cost estimates for each of the U, V, W, and X-list interactions. The cost for each node in the tree is then determined by multiplying the number of each interaction by its corresponding cost estimate, and summing these costs across all interaction types. As with the initial partitioning, we partition the distributed tree by determining a set of separators between contiguous chunks of leaf boxes. Therefore we account for the costs of the interior nodes by percolating their estimated costs down to the leaf boxes and adding them to the extant cost estimate at each leaf. A variant of Algorithm 1 from [68] is then used to determine $P - 1$ locations at which to split the leaf level Morton curve for a re-balancing of the computational load. While the overall

strategy is similar to the costzones approach of [63, 64], our strategy accounts for the work of the entire tree instead of just the leaves. This is important for surface geometries with non-uniform distributions, as the amount of work above the leaves is not negligible.

2.4.4 Evaluation of the potential in parallel

The parallel potential evaluation is performed in three stages: The upward pass (M2M), translation (M2L), and the downward pass (L2L). Algorithm 2 describes the M2M (upward pass) stage of our implementation, which essentially entails shifting the multipole data of each tree node to the center of its parent box and aggregating the shifted multipole data from all siblings. Each process starts processing the nodes in its local subtree from right-to-left in the post-order traversal sequence. By choosing to go from right-to-left, we ensure that duplicate nodes are encountered toward the beginning of the M2M stage, and shared nodes are encountered toward the end of this stage. Using MPI's non-blocking `Ireduce` collective calls, this pattern facilitates overlapping the communication operations for plural nodes whose children, by definition, reside on different processes with M2M computations of other nodes.

The translation stage commences on each process once the local upward pass is complete. This stage includes three substages for computations of X, V and W lists. First, X-list interactions are handled by sending the required multipole expansions to processes with X-list observers, where the observed potentials are computed. Next, V-list interactions are carried out. Typically, this is the most expensive stage of any fast multipole-like method. To hide communication overheads, the source multipole expansions to be exchanged are divided up into packets and communicated to processes that need this information. As each source expansion is received in full, all its V-list interactions are computed, making good use of temporal locality. Once all source expansions in the current packet are exhausted of work, the next packet is constructed and communicated using non-blocking primitives to facilitate overlapping of communication and computation. This process continues until all remote V-list interactions are completed. Local V-list interactions, *i.e.*, those in which both the source and observer boxes belong to the same process, are computed next. Finally,

W-list interactions are computed by exchanging the source weights, i.e. u_i in (2.6), and applying the appropriate translation operators to form the local expansions of these sources.

To complete the evaluation, Algorithm 3 describes the L2L (downward pass) stage which involves re-centering and adding parents' local expansions to their childrens'. Contrary to the M2M stage, here we employ a pre-order traversal of the local subtree so that the shared nodes are now encountered at the beginning and the duplicates are encountered toward the end of the L2L stage. This way, by using MPI's non-blocking `IBcast` primitive, we ensure that broadcast for a shared node can be completed in the background, while L2L computations of other nodes are being performed. Also, the result of the V-list interaction evaluations for duplicate nodes is communicated to users during the downward pass, again using non-blocking primitives for overlapping communications with computations.

As described above, updating multipole and local expansions for nodes that are shared between processes are performed efficiently in our implementation using asynchronous communications. We note that at any level, the local subtree of any process can have at *most* two plural nodes – one shared, one duplicate. It follows, then, that we may create a total of $2L$ MPI communicators, each with groupings of shared nodes and their users, so that we can take advantage of non-blocking variants of MPI's reduce and broadcast operations.

We note that our evaluation algorithm differs from that of the common LET-based implementations in that we avoid duplicating V-list interactions, which are typically the most computationally expensive part of any fast multipole-like algorithm (see Sect 3.5). Each V-list interaction is computed exactly once, and the resident process is responsible for such computations.

2.5 Results

In this section, we present an array of results demonstrating error convergence and parallel performance of the present algorithm.

Algorithm 2: Local multipole-to-multipole (M2M) computation with interleaved update of plural nodes

Define `pid` as the process rank
Define $M(b)$ as the multipole expansion for node b
Define $T(b)$ as a temporary storage space of the same size as $M(b)$ unique to node b; only required for plural nodes
Define Reqs$(\cdot)$ as the array of request handles for asynchronous comms.
$r \leftarrow 0$ {r is the request counter}
$\mathcal{R} \leftarrow \emptyset$ {Set of update nodes}
for each $b \in \mathcal{T}_{loc}$ in right-to-left post-order **do**
  if b is a leaf **then**
    Calculate $M(b)$ from particles of b
  else
    for each child c of b **do**
      $M(b) \leftarrow M(b) +$ `M2M`(c, b)
    end for
    if b is a plural node **then**
      $r \leftarrow r + 1$
      `MPI_Ireduce`($M(b)$,$T(b)$,Reqs(r),Root(b),`MPI_SUM`)
      if `pid` == Root(b) **then**
        $\mathcal{R} \leftarrow \mathcal{R} \cup b$
      end if
    end if
  end if
end for
`MPI_Waitall`(r,Reqs($1 : r$))
for each $b \in \mathcal{R}$ **do**
  $M(b) \leftarrow T(b)$
end for

2.5.1 Error convergence

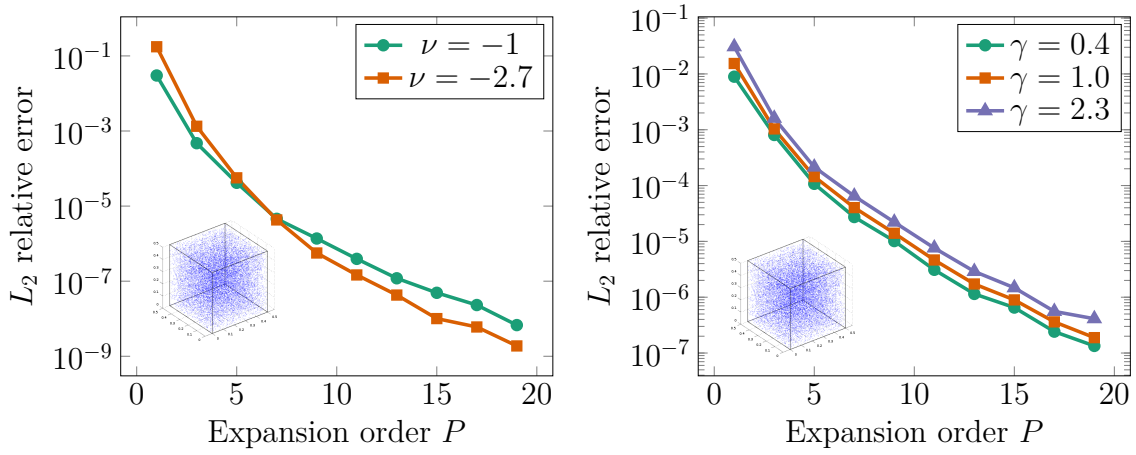
We first examine the error convergence of the ACE algorithm applied to the $r^{-\nu}$, Yukawa, and Stokes potentials. The first two are computed using scalar sources, while the Stokes potentials are calculated using vector-valued sources. The error for Stokes potentials is compared to the analytical by taking an inner product of the tensor-valued potential with each observer's polarization vector. We first evaluate the $r^{-\nu}$ and Yukawa potentials for 3.125 million points within a 0.64 m cube. The distribution is mapped onto a tree with a leaf box diameter of $d_0 = 0.01$ m, yielding a 7-level tree with 12 points per box on average. A single buffer box is used for the far-field. Fig. 2.3

Algorithm 3: Local-to-local (L2L) computation with asynchronous update of plural nodes

```
Define L(b) as the local expansion for node b
r ← 0
for each b ∈ T in pre-order do
    if b is a leaf then
        Calculate potentials due to L(b) for each particle in b
    else
        if b is a plural node then
            r ← r + 1
            MPI_Ibcast(L(b),Reqs(r),Root(b))
            if b is a duplicate node then
                MPI_Wait(Reqs(r))
            end if
        end if
        for each child c of b do
            if c is not a duplicate node then
                L(c) ← L(c) + L2L(c, b)
            end if
        end for
    end if
end for
```

shows convergence in the L_2 error for the far-field with increasing expansion order $P = 1, 3, \ldots, 19$ for both kernels. Different parameters are used to control the growth or decay of the kernels as r increases.

We next consider the evaluation of error in Stokes potentials for a collection of points within a cube with both uniform and non-uniform spatial distributions. Both distributions are characterized by 3.125 million points within a 0.64 m cube. In the case of the uniform distribution, we employ a uniform tree with increasing box size as P is increased to minimize runtime. The non-uniform distribution is generated by generating random points within the unit cube and raising the generated x, y, z positions to the powers of 1.2, 0.7, and 1.7, respectively, before rescaling the results to fit into the 0.64 m cube. In this case, we use a non-uniform tree with 2:1 balance and increase the minimum box size with P to approximately minimize runtime according to the uniform-tree cost estimate. The error convergence with increasing P for both distributions is shown in Fig. 2.4. The error metric used here relies on randomly sampling the analytical fields at one observer on each



(a) $r^{-\nu}$ potential

(b) Yukawa potential $e^{-\gamma r}/r, \gamma > 0$

Figure 2.3: Far-field only error convergence vs. ACE expansion order for different kernel functions.

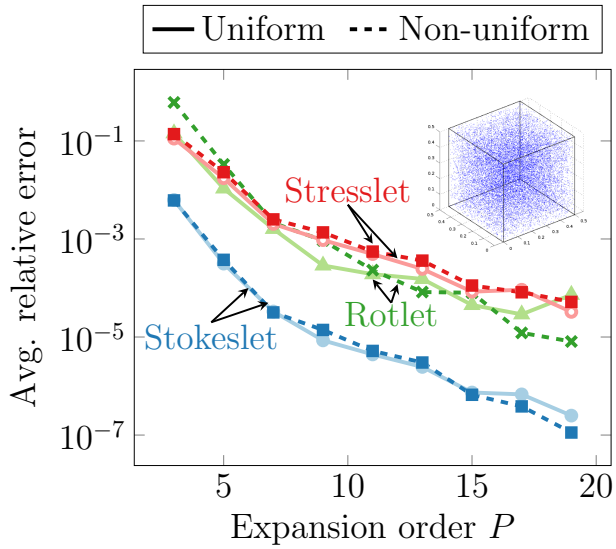


Figure 2.4: Convergence of the Stokes potentials for uniform and non-uniform volume distributions.

process, ensuring a good spatial distribution of observers, and comparing it with the computed potential using ACE. For each selected particle, the error in the potential relative to the analytic solution is computed, and the average of these errors is reported.

2.5.2 Cost comparison of evaluating $\mathcal{L}_\nabla\{\psi\}$ vs. ψ

Next, we demonstrate the low overhead for evaluating $\mathcal{L}_\nabla\{\psi\}$ using our method. Consider the evaluation of the Laplace potential and the Stokes potentials for a collection of 100,000 particles

Table 2.1: L2O timings (in seconds) for Laplace potential (scalar) and Stokeslet (vector)

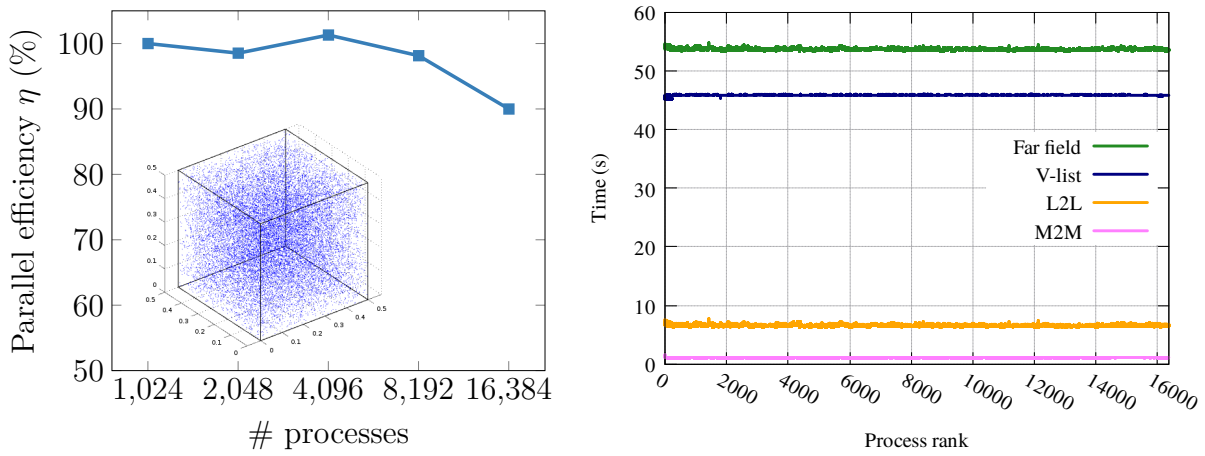
P	6	8	10	12	14	16	18
Stokeslet	0.118	0.198	0.333	0.570	0.769	1.125	1.561
Laplace	0.046	0.072	0.104	0.156	0.229	0.317	0.418
Ratio	2.57	2.75	3.20	3.65	3.36	3.55	3.73

distributed randomly inside a cube of diameter 1 m. Note, for the Stokes potentials we use vector sources. It follows from the arguments thusfar, the principal difference in cost between evaluating the Laplace and Stokes potentials should be a factor of three arising from the difference between the nature of the source distributions (scalar vs. vector). For the Stokes potentials, each of the three vector components of the local expansion must be disaggregated with the derivative operations defined in (2.12) and assembled to form the observed field. Table 2.1 presents timings of the L2O stage for both the Laplace and Stokes potentials for increasing values of P and the ratio of the Stokeslet time to the Laplace time. The increase hovers around the expected factor of 3, albeit with additional overhead for larger P which could potentially be due in part to cache effects associated with the $O(P^3)$ arrays involved in these computations.

2.5.3 Parallel performance

Finally, we examine the performance of the algorithm in evaluating all three Stokes potentials simultaneously at the L2O stage. These results were obtained on the Haswell partition of the Cori supercomputer at the National Energy Research Scientific Computing Center (NERSC). This cluster comprises 2388 compute nodes with two sockets each, populated by 16-core Intel Xeon E5-2698 v3 "Haswell" CPUs running at 2.3 GHz and 64 GB DDR4 RAM at 2133 MHz per socket. The algorithm was implemented in Fortran 90 using double-precision arithmetic and parallelized strictly in distributed-memory fashion using MPI. The code was compiled using the Intel compiler version 18 with optimization and architecture-specific instructions using the `-O3 -xHost` flags. For these runs we use vector-valued sources in $\mathbb{R}^3$ and evaluate the far-field only, selecting $P = 7$ to give $O(10^{-5})$ accuracy for the Stokeslet and $O(10^{-3})$ accuracy for the rotlet and stresslet.

We next consider a uniform random distribution of 5 billion randomly-oriented vector sources



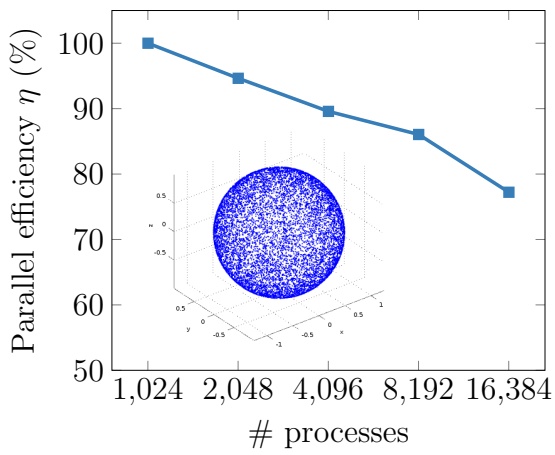
(a) Strong scaling

(b) Per-process timings

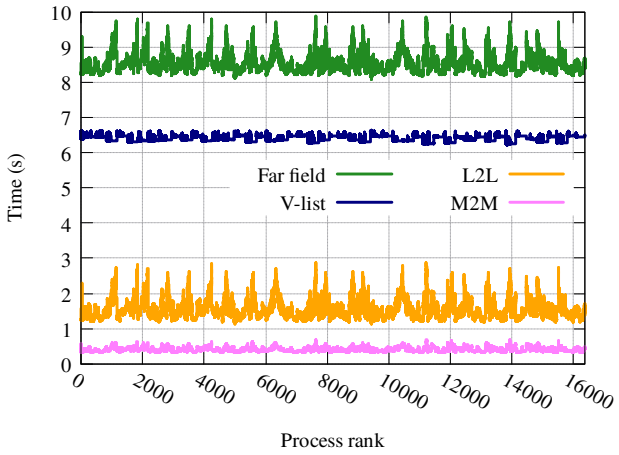
Figure 2.5: Strong scaling and per-process timings for Stokes kernel evaluation on a uniformly-distributed volume geometry with 5 billion particles.

with unit norm inside a cube of diameter 8 m. The minimum box size was set to 0.015 m, yielding an 11-level uniform tree with 33 particles per leaf box on average. Fig. 2.5a shows the parallel efficiency (strong scaling) of this distribution up to 16,384 processes with respect to 1024 processes. The increase in efficiency from 2048 to 4096 processes is due to the fact that the distribution is randomly re-generated for each run. The efficiency is over 90% for all cases. Fig. 2.5b shows the timings for each process for the 16,384 process case. The C2M stage requires on average 0.54 s while the L2O stage, modified for the Stokes potentials, requires 6.05 s, about an 11.2X increase. Most of this increase is due to the calculation of the stresslet potential. The overall computational time is dominated by the V-list calculation, taking over 45 s of the 54 s, underscoring the importance of avoiding redundant V-list calculations related to duplicate nodes at upper levels of the tree.

We now consider uniformly and non-uniformly distributed sources on the surface of a sphere. Both distributions comprise 1.024 billion particles. For the uniform distribution, the sphere diameter is 2 m. A uniform tree is used with a leaf box size of 5×10^{-4} m, yielding a 13-level tree with leaves containing 15 particles per box on average. The strong scaling and per-process timings on 16,384 processes for the uniform distribution are shown in Fig. 2.6. A mock distribution is also shown inset in the strong scaling plot. In this case, the efficiency is over 77%. The choppiness of

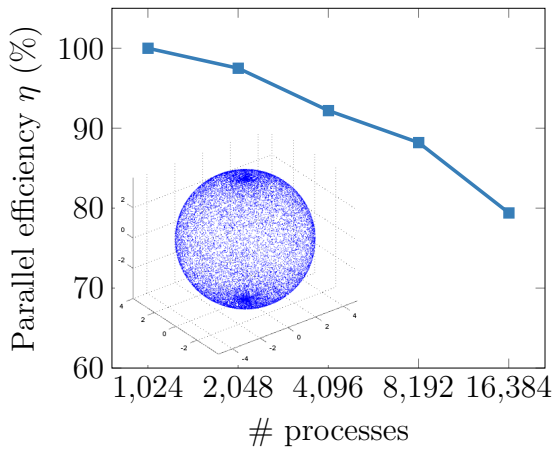


(a) Strong scaling

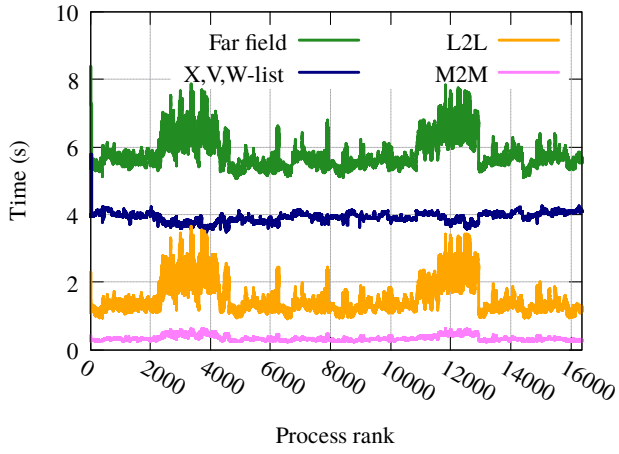


(b) Per-process timings

Figure 2.6: Strong scaling and per-process timings for Stokes kernel evaluation on a uniform spherical distribution of 1.024 billion points.



(a) Strong scaling



(b) Per-process timings

Figure 2.7: Strong scaling and per-process timings for Stokes kernel evaluation on a non-uniform spherical distribution of 1.024 billion points.

the L2L stage timings is due to the fact that the number of particles assigned to each process is uneven; instead, the load balancing algorithm aims to balance the overall computational load. As the bulk of the work lies in the V-list evaluation, Fig. 2.6 suggests the algorithm does a reasonably good job.

In the case of the non-uniform distribution, the sphere has a diameter of 8 m, and the particles are clustered around the north and south ($\pm z$) poles (as depicted inset in Fig. 2.7a). The minimum box size is chosen as 6.25×10^{-7} m in diameter, resulting in a 25-level tree. We set $s_{max} = 50$ particles per box, resulting in about 18 particles per box on average and distributing leaves over the bottom 16 levels of the tree. Despite the extremely non-uniform distribution of particles, the strong scaling shown in Fig. 2.7a is still as high as 78% efficient on 16,384 processes. In addition, the per-process timings presented in Fig. 2.7 for the far-field evaluation suggest that the load balancing algorithm does its job reasonably well. Again, the choppiness here is due to the fact that particles are not evenly distributed across processes, and processes with high particle counts must spend a lot of time computing X- and W-list interactions. Those with the most particles stick out in these plots.

2.6 Conclusion

We have presented an efficient parallel algorithm for evaluating arbitrary non-oscillatory potentials and those defined in terms of gradients on a non-oscillatory potential. We demonstrated the accuracy of the numerical method for several different potential functions and showed that the parallel algorithm scales well for surface, volume, uniform, and non-uniform distributions of billions of particles, achieving nearly 78% parallel efficiency on 16,384 processes.

CHAPTER 3

PARALLEL NON-UNIFORM WIDEBAND MLFMA FOR MULTISCALE ELECTROMAGNETIC SIMULATION

3.1 Introduction

Over the past several decades, computer simulation has become an indispensable tool for designing and prototyping in electromagnetic (EM) engineering. This includes designing microwave/RF circuits and antennas, electromagnetic interference mitigation, stealth aircraft profile reduction, THz devices, and much more. Many of these problems, however, pose significant challenges for existing computational methods. The confluence of perpetual increase in frequencies of engineering interest, the widespread availability of quality computational resources, and the demand for realism in computer models is driving a massive increase in the number of degrees of freedom, N_s, required to solve these problems.

While the literature is filled with means to tackle these problems, over the years, surface integral equations have emerged as the main tool of analysis [54, 20, 41]. The main bottleneck to widespread use of these methods was their computational complexity; since the 1990s, this has largely been overcome thanks to accelerators such as the multilevel fast multipole algorithm (MLFMA) that have reduced the computational complexity from $\mathcal{O}(N_s^2)$ to $\mathcal{O}(N_s \log N_s)$ for surfaces. As a result of these significant benefits, research into advancing nuances and application of these approaches is extensive and perhaps too numerous to enumerate; a partial list can be found in [71, 52] and references therein. As is evident from these citations, there is still considerable interest in exploring and expanding the range of these techniques at either end, i.e., extending these methods to problems with high disparity in discretization scales that are then embedded in electrically large objects. At the same time, there has been a concerted effort to develop parallel algorithms to further exploit the capabilities of these algorithms. Both these issues are explored in more detail next.

The literature is rich with efforts to bridge the length scale between the Helmholtz regime and

the low-frequency or Laplace regime. The principal challenge is the low-frequency breakdown of the classic MLFMA. The earliest work on this problem was done by Greengard [35, 37]. Soon after, Zhao and Chew [83] introduced a modification of the original addition theorem to stabilize the MLFMA at low frequencies. Since these, other methods have emerged, including [19, 26, 12, 5, 69]. Another method introduced in 2007 is based on accelerated Cartesian expansions [61, 73] blends seamlessly and intuitively with the MLFMA, and is error-controllable to arbitrary accuracy with no bound on the discretization density.

It is apparent that to extend the reach of fast multipole methods, development of parallel algorithms is a necessity [70]. The earliest efforts were based on extending methods developed for the Laplace (electrostatic) fast multipole method (FMM) to MLFMA with little success. Recent effort focused on building an algorithm based on hierarchical partitioning (HiP-MLFMA) [28] and more recently, its blockwise variant [51]. This approach works by partitioning in space at lower levels and directions at higher levels; the blockwise scheme optimizes the manner in which the directions are partitioned. In these methods, one uses a number of processors that is either a power of 2 (for hierarchical partitioning) or a power of 4 (for blockwise hierarchical partitioning).

Another algorithm [48] that has also shown excellent scalability and timings takes a different approach; using post-order traversal, the resulting self-similarity of the tree as well as direction partitioning yields an algorithm that scales very well [48]. The salient features of this algorithm are (i) wideband MLFMA for analysis from low to high frequencies, (ii) global (exact) interpolation and anterpolation, (iii) demonstrated error control (L_2 error norm against analytical data), and (iv) *adaptive* direction partitioning that is dictated by the number of duplicate nodes; for the purposes of this discussion it will be referred to as AP-MLFMA, for "adaptive partitioning". It was shown that the matrix vector product (*matvec*) times of AP-MLFMA are highly competitive with others. To a large extent, both methods assume a uniform tree.

Despite this progress, several open problems remain, particularly for wideband and multiscale MLFMA systems. These can be summarized as accuracy in deep non-uniform trees and their relative trade-offs in developing a efficient parallel algorithms. To wit, both HiP- and AP-MLFMA

offer a solution to a portion of the problem. The former relies on local inter/anterpolation, which has linear asymptotic complexity but requires significant oversampling, resulting in an increase in cost. It is also inaccurate, as one needs a filter at the anterpolation stage [27, 62, 19].

On the positive side, HiP-MLFMA is easier to parallelize. The AP-MLFMA uses global (exact) inter/anterpolations so the accuracy can be well controlled, and the sampling is optimal and predetermined. However, developing parallel algorithms for this approach is a challenge; in addition, memory costs required to store matrices for inter/anter-polation on each node limit the height of the tree. Both HiP- and AP-MLFMA do not include non-uniform trees within their parallel frameworks. Algorithms that address both wideband and multiscale nature of the problems implies that one needs both the capability of handling low frequencies as well as adaptive non-uniform trees [73, 15, 19]; while such algorithms exist in serial, the literature on parallelization is extremely sparse with each [48, 9] addressing a portion of the problem.

The principal contribution of this chapter is to address these extant deficiencies; our objective is to provide a parallel computational framework for computing fields arising from realistic distributions with the following properties: (a) it is a methodology whose accuracy can be controlled to desired precision, (b) it includes transitions to low frequencies and non-uniform distributions, (c) the sampling of the spectrum is optimal and overcomes memory bottlenecks associated with inter/anterpolations, and (d) it is efficient in terms of scalability and execution times. We present a number of results to demonstrate the accuracy of the numerical methods and the performance of the parallel algorithms presented here. The methodology extensively extends the work [73, 48] on ACE-AP-MLFMA.

The rest of the chapter is organized as follows: Section 3.2 lays out the EM scattering problem to be solved. In Section 3.3, we review the wideband MLFMA, establish the need for filters for anterpolation, develop transitions between spherical harmonics and Fourier series for interpolation/anterpolation, non-uniform trees for multiscale features, and an interpolation method for saving memory. Section 3.4 outlines the parallel algorithm, and finally, Section 3.5 demonstrates the error control and performance of the serial and parallel algorithms.

3.2 Problem Statement

Consider an object residing in a region $\mathcal{D} \in \mathbb{R}^3$ bounded by the surface $\partial\mathcal{D}$, the perfectly-conducting portion of which is denoted by S. Let $\left\{\mathbf{E}^i(\mathbf{r}), \mathbf{H}^i(\mathbf{r})\right\}$ denote electric and magnetic fields that are incident on this object, μ and ε denote the magnetic permeability and electric permittivity of the medium, respectively, and $k = \omega\sqrt{\mu\varepsilon}$ denote the wavenumber for frequency ω. The intrinsic impedance of the medium is denoted by $\eta = \sqrt{\mu/\varepsilon}$. The field scattered by the object may be obtained using a combined field integral equation (CFIE) formulated in terms of the unknown electric surface current density $\mathbf{J}(\mathbf{r})$ on S. For $\mathbf{r} \in S$,

$$\begin{aligned} &\alpha\hat{n} \times \hat{n} \times \mathbf{E}^i(\mathbf{r}) + (1-\alpha)\hat{n} \times \mathbf{H}^i(\mathbf{r}) \\ &\quad = -\alpha\mathcal{L}\{\mathbf{J}\}(\mathbf{r}) + (1-\alpha)\mathcal{K}\{\mathbf{J}\}(\mathbf{r}) \end{aligned} \tag{3.1}$$

$$\mathcal{L}\{\mathbf{J}\}(\mathbf{r}) \doteq \hat{n} \times \hat{n} \times \int_S \mathbf{G}\left(\mathbf{r}-\mathbf{r}'\right) \cdot \mathbf{J}(\mathbf{r}')\, dS' \tag{3.2}$$

$$\mathcal{K}\{\mathbf{J}\}(\mathbf{r}) \doteq \hat{n} \times \frac{1}{jk\eta}\nabla \times \int_S \mathbf{G}\left(\mathbf{r}-\mathbf{r}'\right) \cdot \mathbf{J}(\mathbf{r}')\, dS' \tag{3.3}$$

$$\mathbf{G}\left(\mathbf{r}-\mathbf{r}'\right) \doteq -jk\eta\left[\mathbb{I} + \frac{\nabla\nabla}{k^2}\right] g(\mathbf{r}-\mathbf{r}') \tag{3.4}$$

where

$$g(\mathbf{r}-\mathbf{r}') \doteq \frac{e^{-jk|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}-\mathbf{r}'|}, \tag{3.5}$$

is the scalar Helmholtz Green's function and $\hat{n} \doteq \hat{n}(\mathbf{r})$ denotes the unit normal to S at the point $\mathbf{r}$. Solving for $\mathbf{J}$ via the method of moments (MoM) involves the discretization of (3.1), typically by expansion and testing with the classic RWG basis functions [58] defined on a triangular mesh. To do so, we expand the unknown surface current as

$$\mathbf{J}(\mathbf{r}) = \sum_{n=1}^{N_S} I_n \mathbf{f}_n(\mathbf{r}), \tag{3.6}$$

where $\mathbf{f}_n(\mathbf{r})$ denotes the RWG function associated with the nth edge, and N_s is the total number of edges in the mesh. The $N_s \times N_s$ matrix equation resulting from the MoM procedure may be expressed as

$$ZI = V, \tag{3.7}$$

where $I = \left\{I_1, \ldots, I_{N_s}\right\}^T$ is the vector of unknown coefficients, and the elements of the matrix Z and vector V are given by

$$Z_{mn} = \alpha Z^E_{mn} + (1-\alpha) Z^H_{mn} \tag{3.8}$$

$$\begin{aligned} Z^E_{mn} &= -\langle \mathbf{f}_m(\mathbf{r}), \mathcal{L}\{\mathbf{f}_n\}(\mathbf{r})\rangle \\ &= \left\langle \mathbf{f}_m(\mathbf{r}), jk\eta \int_S g(\mathbf{r}-\mathbf{r}')\mathbf{f}_n(\mathbf{r}')dS' \right\rangle \\ &\quad - \left\langle \nabla\cdot\mathbf{f}_m(\mathbf{r}), \frac{j\eta}{k}\int_S g(\mathbf{r}-\mathbf{r}')\nabla'\cdot\mathbf{f}_n(\mathbf{r}')dS' \right\rangle \end{aligned} \tag{3.9}$$

$$\begin{aligned} Z^H_{mn} &= \langle \mathbf{f}_m(\mathbf{r}), \mathcal{K}\{\mathbf{f}_n\}(\mathbf{r})\rangle \\ &= \left\langle \mathbf{f}_m(\mathbf{r})\times\hat{n}, \int_S \nabla g(\mathbf{r}-\mathbf{r}')\times\mathbf{f}_n(\mathbf{r}')dS' \right\rangle \end{aligned} \tag{3.10}$$

$$V_m = \left\langle \mathbf{f}_m(\mathbf{r}), \alpha\hat{n}\times\hat{n}\times\mathbf{E}^i(\mathbf{r}) + (1-\alpha)\hat{n}\times\mathbf{H}^i(\mathbf{r}) \right\rangle, \tag{3.11}$$

where $\langle\cdot,\cdot\rangle$ denotes the usual inner product. Here, the EFIE is represented in mixed-potential form.

Solutions to this equation are typically effected using an iterative solver, for instance, the Generalized Minimal Residual (GMRES) method. Coupling this equation with a accelerator amortizes the cost of the matrix vector product; the most popular of the accelerators being the multilevel fast multipole method (MLFMA) [67]. Next we discuss challenges and their resolution when these methods are applied for multiscale analysis.

3.3 Challenges and remedies for parallel multiscale analysis

All acceleration methods for accelerating the solution of (3.7) rely on the fact that the matrix Z may be partitioned into a matrix Z^{NF} of near interactions and a matrix Z^{FF} of far interactions, or

$$Z = Z^{NF} + Z^{FF}. \tag{3.12}$$

This partitioning is founded on the notion that *matvecs* with Z^{FF} can be computed in $O(N_s \log N_s)$ time, and those with Z^{NF} can be computed in $O(N_s)$ time. Developing criteria for such a partitioning for multiscale problems poses challenges from both a computational as well as parallelization perspectives. These challenges arise from the non-uniformity of discretization and their distribution. To set the stage, assume that the spatial distribution of unknowns can be mapped onto a uniform

octree data structure. The root of the tree contains all unknowns, and the leaves of the tree correspond to clusters of unknowns, serving as the interface between the tree and the unknowns. The *matvec* with Z^{FF} is effected by using appropriate operators to traverse up, down and across the tree. Assume that the minimum size of a leaf is around 0.2λ. In a multiscale scenario a leaf box may contain too many unknowns, thus dramatically increasing computational costs of near-field interactions and overwhelming the cost complexity of the MLFMA. This can be overcome using a judicious choice of representations for the Green's function that enable a low-frequency decomposition, as in [37, 19, 12]; here we employ the accelerated Cartesian expansion (ACE) method [61, 73]. Enabling this results in a tree that could potentially have MLFMA leaves that are then roots of subtrees that extend downwards. From a computational perspective one needs to develop operators to efficiently transfer information between nodes (both at the same level and between levels).

Another bottleneck is accurate traversal up and down the tree. As will be shown later, a provably accurate approach is to use a global representation of radiated field or use a local bandlimited representation [46, 49]. For the latter, the price that one pays is the oversampling required for the same accuracy. However, the downside of using global sampling is the challenges it poses in terms of memory requirements for storing interpolation coefficients at higher levels in the tree as well as parallelization. As a result, an efficient strategy for deep trees would use a spherical filter that uses spherical sampling up to a certain level, switches to a FFT-based filter using uniform sampling [60] to save on memory, and then a local bandlimited filter with linear complexity to leverage parallelism. This concept is illustrated in Fig. 3.1. While we leave the discussion of local bandlimited interpolation and modification of parallel algorithms to a subsequent paper, we shall extensively discuss spherical, uniform sampling and transitions between the two, and parallelization strategies that one can employ. In what follows, we will explore each of these issues in greater detail. Note, the exposition will focus on evaluation of the scalar Green's function; changes necessary to effect these operations for the dyadic Green's function are well known and only those subtleties that are *not* well known are elaborated upon.

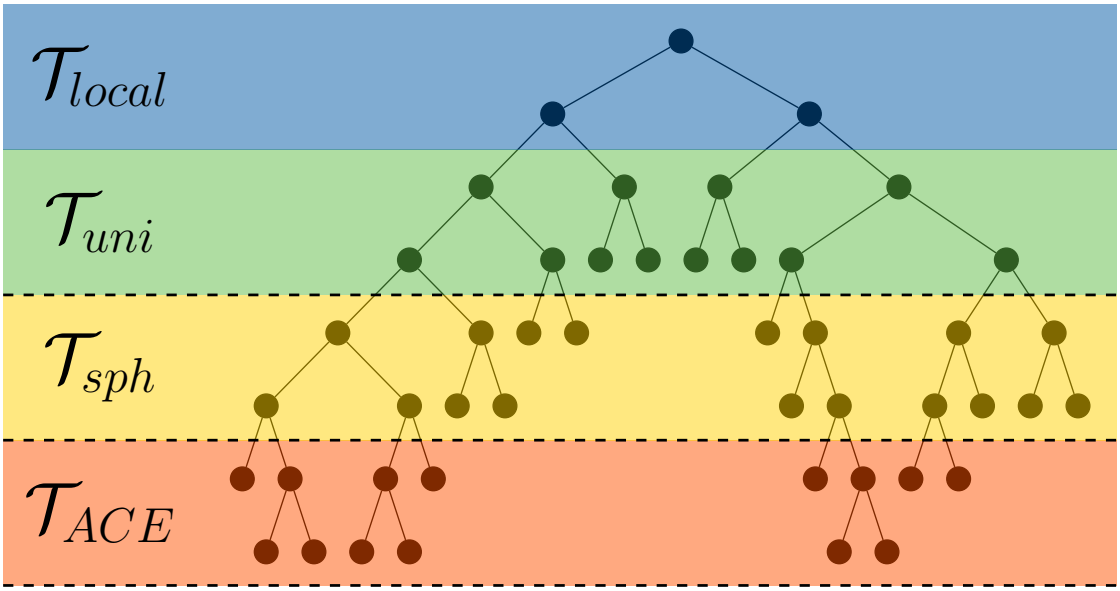


Figure 3.1: Illustration of the target computational strategy for representing interactions in a general multiscale geometry. Each $\mathcal{T}$ represents operations performed with the subscripted method, e.g. ACE, spherical and uniform interpolation/anterpolation, and local band-limited interpolation/anterpolation.

3.3.1 Non-uniform trees

MLFMA relies on construction of oct-trees to partition near and far interactions. Here we briefly discuss creating such list for non-uniform distributions. Consider a box b at any level in the tree; we denote its parent by $P(b)$ and its grandparent by $P^2(b)$ and so on up the tree. Boxes that share a spatial locations (vertex, edge or face) with b are identified as being in its near field. If two boxes are of the same size (and share a spatial location), they belong to the U-list. Likewise if two boxes are of the same size, do not share a spatial location, and their parents are in the near field of each other, they are in each other's farfield or V-list. To handle non-uniform distributions, we develop an adaptive tree [15] starting with a uniform tree representation where all leaf boxes reside precisely at the same level, merging siblings subject to the rule that their parent does not contain more than some specified s_{max} DoF.

An important consequence of adaptive methods is the introduction of far-field interactions between boxes at different levels of the tree or *cross-level* interactions. These interactions, deemed the X- and W-lists, are discovered during the construction of the near and far interaction lists in the

following manner. Consider a box b. To construct its interaction lists, we generate all hypothetically interacting boxes with b at the same level according to the scheme described earlier, and we perform a top-down search of the octree to locate each such hypothetical box h. If h itself is found in the tree, it is obviously added to the appropriate U- or V-list. If the search for h dead-ends on an ancestor $P^n(h)$ that is not a leaf, the interaction pair (b, h) is discarded; however, if $P^n(h)$ *is* a leaf and it satisfies the conditions of the present interaction list, we retain $(b, P^n(h))$ in the X-list and its reciprocal interaction pair $(P^n(h), b)$ in the W-list.

The computational procedure is as follows:

- Construct the non-uniform tree and delineate near and far field interactions
- Precompute and store near field interactions
- For any *matvec*,
  - Compute contributions due to the near-field
  - Compute contributions due to the far-field:
    * For all leaves, compute charge to multipole (C2M);
    * Construct multipole information for all boxes from those of their children (M2M). This operation is level dependent, and one needs the following operators: $\mathrm{M}_{ACE} \longrightarrow \mathrm{M}_{ACE}$, $\mathrm{M}_{ACE} \longrightarrow \mathrm{M}_{sph}$, $\mathrm{M}_{sph} \longrightarrow \mathrm{M}_{sph}$, $\mathrm{M}_{sph} \longrightarrow \mathrm{M}_{uni}$ and $\mathrm{M}_{uni} \longrightarrow \mathrm{M}_{uni}$;
    * Translate from multipole to local expansion (M2L). For non-uniform trees, this would involve constructing operators for both in- and across-level translations;
    * Construct local expansions for all boxes from those of its parent (L2L). This is a conceptual inverse of the multipole operation and one needs to develop identical operators to traverse down the tree;
    * From local expansions at leaves, construct far field information at all observers (L2O);
  - Sum the contributions of near- and far-fields.

3.3.2 Low-frequency analysis

Over distances or length scales which are small relative to a wavelength ($< 0.25\lambda$ or so), the oscillatory behavior of g over larger distances gives way to quasi-static behavior which can be accurately modeled using localized Taylor series expansions. This is the foundation of the ACE method [61, 73] for the low-frequency Helmholtz problem.

Consider the evaluation of the potential $\Psi(\mathbf{r})$ due to a source distribution $u(\mathbf{r})$

$$\Psi(\mathbf{r}) = \sum_{n=1}^{N_S} u_n g(\mathbf{r} - \mathbf{r}'_n). \tag{3.13}$$

To compute the potential potentials between regions that are sufficiently separated, ACE [61, 73] dictates that the potential due to s' sources with location and strength $\{\mathbf{r}_m, u_m\}$, $m = 1, \ldots, s'$ within the region Ω_s centered at $\mathbf{r}_s^c$ can be written as

$$\Psi^{FF}(\mathbf{r}) = \sum_{n=0}^{\infty} \mathbf{M}^{(n)}(\mathbf{r}_s^c) \cdot n \cdot \nabla^{(n)} g(\mathbf{r} - \mathbf{r}_s^c), \tag{3.14}$$

where $\cdot n \cdot$ denotes an n-fold tensor contraction, $\mathbf{M}^{(n)}$ is the multipole tensor representing the sources within Ω_s and the tensor $\nabla^{(n)} g(\mathbf{r} - \mathbf{r}_s^c)$ is an n-fold tensor operand on the Green's function. Details for these expression, as well as aggregation operators $\mathrm{M}_{ACE} \longrightarrow \mathrm{M}_{ACE}$ and $\mathrm{M}_{ACE} \longrightarrow \mathrm{M}_{sph}$ and their counterparts for disaggregation are available in [73]. Operators addressing translations across levels are presented later in this section.

3.3.3 MLFMA

The derivation of FMM and its multilevel variant are well known [21, 67, 66, 22, 60]. Given the wealth of papers in this area over the past few decades, it seems somewhat presumptuous to contend that there is significant new information to be added to the literature. The purpose of this section is somewhat different. In keeping with our goal to have a methodology with controllable accuracy, we will describe methods that permit both better understanding and error control. We start with a 2-level description of MLFMA that is based on the integral representation of the Green's function

$$g(\mathbf{X} + \mathbf{d}) \approx -\frac{jk}{(4\pi)^2} \int_{S^2} e^{-j\mathbf{k}\cdot(\mathbf{d}_S + \mathbf{d}_O)} T(\mathbf{k}, \mathbf{X}) d^2\hat{k}, \tag{3.15}$$

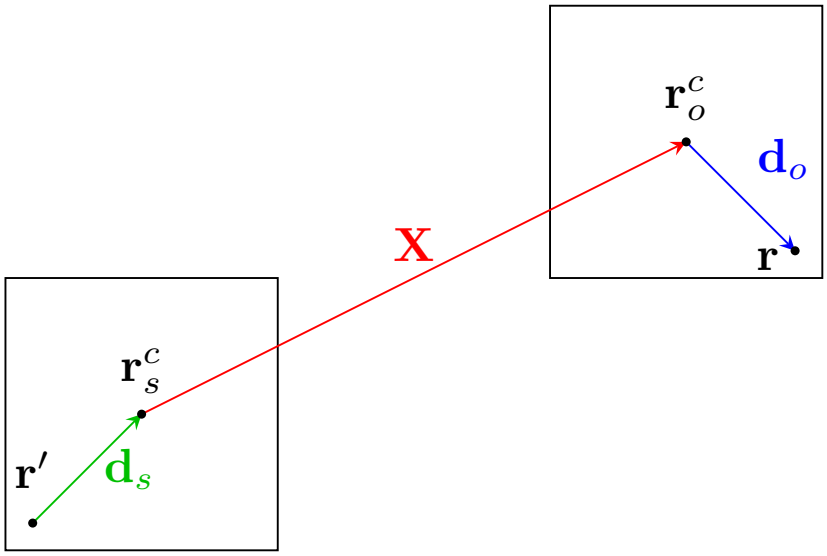


Figure 3.2: Graphical illustration of the addition theorem and some notation.

where the *translation operator* T is given by

$$T(\mathbf{k}, \mathbf{X}) \doteq \sum_{n=0}^{\infty} (-j)^n (2n+1) h_n^{(2)}(kX) P_n\left(\hat{k} \cdot \hat{X}\right), \tag{3.16}$$

with $\mathbf{k} = k\hat{k}$. Here, S^2 denotes the unit sphere, parametrized by $(\theta, \phi) \in [0, \pi] \times [0, 2\pi]$. We note that $\mathbf{k} = \mathbf{k}(\theta, \phi)$, and use these notations interchangeably. Fig. 3.2 illustrates the decomposition of $|\mathbf{r} - \mathbf{r}'|$ into $|\mathbf{X} + \mathbf{d}(1)|$, where $\mathbf{d}(l)$ denotes the sum of particle-to-center vectors at level l with $l = 1$ being leaf level. We denote the centers of boxes containing $\mathbf{r}'$ and $\mathbf{r}$ as $\mathbf{r}_s^c(1)$ and $\mathbf{r}_o^c(1)$, respectively. It follows that $\mathbf{d} = \mathbf{r} - \mathbf{r}_o^c(1) + \mathbf{r}_s^c(1) - \mathbf{r}'$. The rules to evaluate the spectral integral are well-understood and in place [27, 66, 60, 19, 73]. The far-field part of the potential (3.13) is evaluated using

$$\Psi^{FF}(\mathbf{r}) \approx -\frac{jk}{(4\pi)^2} \int_{S^2} e^{-j\mathbf{k}\cdot\mathbf{d}_o} U_1(\theta, \phi)\, d^2\hat{k}, \tag{3.17}$$

where the *local expansion* U_1 of the observer box is

$$U_1(\theta, \phi) \doteq T(\mathbf{k}, \mathbf{X}) V_1(\mathbf{k}), \tag{3.18}$$

and V_1 denotes the *multipole expansion* of the source box

$$V_1(\theta, \phi) = \sum_{i=1}^{s'} u_i e^{-j\mathbf{k}\cdot(\mathbf{r}_s^c - \mathbf{r}_i')}. \tag{3.19}$$

In what follows, we discuss the intimate connection between integration, interpolation ($V_{l+1}(\theta,\phi) \longrightarrow V_l(\theta,\phi)$) and anterpolation ($U_{l+1}(\theta,\phi) \longrightarrow U_l(\theta,\phi)$).

3.3.3.1 Inter/anterpolation and spectral integration

Let us revisit (3.15). It is well known that the plane wave expansion gives us

$$\begin{aligned} e^{-j\mathbf{k}\cdot(\mathbf{d}_s(1)+\mathbf{d}_o(1))} &\approx S(kd,\theta,\phi) \\ &= \sum_{n=0}^{N_h(1)} \sum_{m=-n}^{n} a_{nm} Y_{nm}(\theta,\phi) \end{aligned} \tag{3.20}$$

where $d = |\mathbf{d}(1)| = |\mathbf{d}_s(1) + \mathbf{d}_o(1)|$, and a_{nm} are known coefficients of the normalized spherical harmonics, $Y_{nm}(\theta,\phi)$, used to represent the plane wave. Assuming that $T(\mathbf{k},\mathbf{X})$ can be written as $T(\mathbf{k},\mathbf{X}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} b_{nm}(k|\mathbf{X}|)Y_{nm}^*(\theta,\phi)$, it follows that the (3.20) filters the above representation via (3.15), i.e.,

$$\begin{aligned} &\int_{S^2} S(kd,\theta,\phi)T(\mathbf{k},\mathbf{X})d^2\hat{k} \\ &= \int_{S^2} d^2\hat{k} \left[\sum_{n=0}^{N_h(1)} \sum_{m=-n}^{n} a_{nm} Y_{nm}(\theta,\phi)\right] \\ &\qquad \times \left[\sum_{n=0}^{\infty} \sum_{m=-n}^{n} b_{nm} Y_{nm}^*(\theta,\phi)\right] \\ &= \int_{S^2} d^2\hat{k}\, S(kd,\theta,\phi) \cdot \left[\sum_{n=0}^{\cancel{\infty}N_h(1)} \sum_{m=-n}^{n} b_{nm} Y_{nm}^*(\theta,\phi)\right] \end{aligned} \tag{3.21}$$

The filter due to $S(kd,\theta,\phi)$ follows from the fact that, if higher order terms ($n > N_h(1)$) are present, they integrate to zero *analytically*. In other words, $S(kd,\theta,\phi)$ effectively band-limits the spectrum of the incoming wave. As a result, the integration rules (and number of harmonics of the translation operator) are chosen to exactly integrate polynomials of order $2N_h(1)$ present in (3.15). If polynomials of order higher than $2N_h(1)$ were present, then, while they should *theoretically* integrate to zero. But they would not do so numerically if the integration rule is not chosen appropriately.

Given the bandwidth of $S(kd, \theta, \phi)$, the number of quadrature points/samples required to completely represent $S(kd, \theta, \phi)$, i.e, obtain all coefficients a_{nm}, is $(N_h(1)+1)\times(2N_h(1)+1)$. To evaluate the integral (3.15) numerically, one should be able to recover b_{nm} from samples of $T(\mathbf{k}, \mathbf{X})$, and vice-versa. An integration rule to be used can be defined using $N_h(1) = \lceil \chi_s k D(1) \rceil$, $N_\theta(1) = N_h(1) + 1$, and $N_\phi(1) = 2N_h(1) + 1$, where $D(1)$ is the diameter of a sphere that encloses the leaf box, and χ_s is an oversampling factor.

In a multilevel scenario, the same logic holds at every level. That is, both the contributions from that level and those from its parents should be representable in terms of spherical harmonics up to a chosen order. This implies that one needs to develop an anterpolation operator to effect a filter of the incoming spectra to enable a transition from parent to child, or in terms of representation, reduce the maximum degree of spherical harmonics used from $N_h(i + 1) \longrightarrow N_h(i)$. To control errors arising from this transition, it necessary to understand the spectral properties of the anterpolation operator used; local algebraic inter/anter-polation operators does not accomplish the necessary filtering, and errors due to lack of filtering is exacerbated in deep trees. As a result, we will use bandlimited anterpolants. As discussed in Section 3.4, using such global operations poses bottlenecks in parallelization.

Thus far, we have established the need for filters to perform interpolation on outgoing expansions ($V_l(\theta, \phi)$) and more importantly, anterpolation on incoming expansions $U_l(\theta, \phi)$. In order to treat both expeditiously, we denote both using $\Lambda_l(\theta, \phi)$. The two variations that we have used are spherical and Fourier transforms. Spherical filters as used in [40, 62] result in optimal sampling of the far field data. An alternative approach is to use Fourier transforms in both angular directions [60], requiring uniformly-spaced samples. Obviously, the tradeoff is $O(N_h(l)^3)$ memory required to effect the Legendre transforms vs. doubling of the sample count required for the latter. The approach we espouse is to choose a transition level at which the representation switches from spherical to uniform sampling.

Spherical filters Spherical filters are based on projecting an incoming or outgoing field pattern Λ_l onto spherical harmonics as

$$\Lambda_l(\theta,\phi) = \sum_{n=0}^{N_h(l)} \sum_{m=-n}^{n} \lambda_{l,nm} Y_{nm}(\theta,\phi) \tag{3.22}$$

and then performing the various down- and up-sampling operations. We shall not delve into details, as these are well known and obtainable from references [40].

Fourier filters The concept of using Fourier transforms was first introduced into the FMM literature by Sarvas [60]. The key to this approach was the realization that the angular domain of the associated Legendre function can be extended from $[0,\pi] \rightarrow [0,2\pi]$ [78]. To summarize, $\Lambda_l(\theta,\phi)$ can be extended to the entire sphere as

$$\tilde{\Lambda}_l(\theta,\phi) = \begin{cases} \Lambda(\theta,\phi) & (\theta,\phi) \in [0,\pi]\times[0,2\pi] \\ \Lambda(2\pi-\theta,\phi+\pi) & (\theta,\phi) \in [\pi,2\pi]\times[0,\pi] \\ \Lambda(2\pi-\theta,\phi-\pi) & (\theta,\phi) \in [\pi,2\pi]^2 \end{cases} \tag{3.23}$$

If the spectrum of $\Lambda_l(\theta,\phi)$ is bandlimited to $N_h(l)$ harmonics, then it can be written as

$$\tilde{\Lambda}_l(\theta,\phi) = \sum_{n=-N_h(l)}^{N_h(l)} \sum_{m=-N_h(l)}^{N_h(l)} \tilde{\lambda}_{l,nm} e^{-jn\theta} e^{-jm\phi} \tag{3.24}$$

It follows that one needs to sample at $(2N_h(l)+1)\times(2N_h(l)+1)$ uniformly on the sphere to recover the coefficients $\tilde{\lambda}_{l,nm}$. Using the above representation we can write (3.17) as

$$\Psi^{FF}(\mathbf{r}) \approx -\frac{jk}{32\pi^2} \int_0^{2\pi} d\theta \,|\sin\theta| \int_0^{2\pi} d\phi e^{-j\mathbf{k}\cdot\mathbf{d}_{O(1)}} \tilde{\Lambda}_1(\theta,\phi) \tag{3.25}$$

In the above expression, $|\sin\theta|$ is not bandlimited. But as discussed earlier, $\tilde{\Lambda}$ *filters* $|\sin\theta|$ such that one can represent it in terms of a finite Fourier series. Specifically,

$$|\sin\theta| \approx \tilde{s}(\theta) = \sum_{n=-N_h(1)}^{N_h(1)} s_n e^{-jn\theta} \tag{3.26}$$

As a result, it follows that the highest order of harmonics in the θ is $2N_h(1)$, and the integration rules must be designed such that one can evaluate (3.25) using $(4N_h(1)+1)\times(2N_h(1)+1)$. This

is done using uniform sampling in both θ and ϕ directions. Further, we note that as is the usual practice, we can embed $|\sin\theta|$ within the translation operator. As a result, one needs to evaluate an integral of the form

$$\Psi^{FF}(\mathbf{r}) \approx -\frac{jk}{32\pi^2}\int_0^{2\pi} d\theta \int_0^{2\pi} d\phi e^{-j\mathbf{k}\cdot\mathbf{d}_o(1)}\tilde{\Upsilon}_1(\theta,\phi) \tag{3.27}$$

where $\tilde{\Upsilon}_1(\theta,\phi) = \tilde{s}(\theta)T(\mathbf{k},\mathbf{X})$. As before, these ideas can be trivially extended to a multilevel setting.

Spherical to uniform and vice-versa Both spherical and uniform interpolation have their drawbacks in terms of costs. Transitioning from one method to the other yields a computationally efficient scheme. To facilitate this, in what follows we present the means to do so. Consider the transition from spherical to uniform. The procedure is straightforward, i.e.

$$\Lambda_l(\theta,\phi) \xrightarrow{sph} \lambda_{l,nm} \xrightarrow{uni} \tilde{\Lambda}_{l+1}(\theta,\phi), \tag{3.28}$$

where the stacked symbols indicate the sampling regimes and embed a shift operation for upward traversal. The above equation indicates that one traverses from the spherical samples of $\Lambda_l(\theta,\phi)$ to coefficients $\lambda_{l,nm}$ and then using these to construct the requisite samples at uniform points in the $[0,2\pi]\times[0,2\pi]$ grid, and then shift these from the center of the child box to that of its parent. Note that one can exploit symmetry to store field samples only in the $[0,\pi]\times[0,2\pi]$ grid by using an even number of samples in ϕ.

Next, to effect the transition from uniform to non-uniform sampling we consider (3.27):

$$\begin{aligned}\Psi^{FF}(\mathbf{r}) &\approx -\frac{jk}{32\pi^2}\int_0^{2\pi} d\theta \int_0^{2\pi} d\phi e^{-j\mathbf{k}\cdot\mathbf{d}_o(1)}\tilde{\Upsilon}_1(\theta,\phi)\\ &= -\frac{jk}{(4\pi)^2}\int_0^{\pi}\sin\theta d\theta \int_0^{2\pi} d\phi e^{-j\mathbf{k}\cdot\mathbf{d}_o(1)}\frac{\tilde{\Upsilon}_1(\theta,\phi)}{\sin\theta}\end{aligned} \tag{3.29}$$

As is evident from the above expression, the integration has been reduced back to the unit sphere. However, the integration is challenging as $\tilde{\Upsilon}_1(\theta,\phi)/\sin\theta$ is not bandlimited in terms of spherical harmonics. To effect the integral, we have to resort to the notion that the the effective bandwidth

of $e^{-j\mathbf{k}\cdot\mathbf{d}_o}$ *filters* the rest of the integrand. That is,

$$\frac{1}{\sin\theta}\tilde{\Upsilon}_1(\theta,\phi) = \sum_{n=0}^{N_h(1)}\sum_{m=-n}^{n} \upsilon_{1,nm} Y_{nm}(\theta,\phi) \tag{3.30}$$

To obtain $\upsilon_{1,nm}$ we evaluate

$$\upsilon_{1,nm} = \int_0^{\pi} d\theta \int_0^{2\pi} d\phi \tilde{\Upsilon}_1(\theta,\phi) Y^*_{nm}(\theta,\phi) \tag{3.31}$$

The above equation can be evaluated *exactly* by defining an extended spherical harmonic as

$$\hat{Y}_{nm}(\theta,\phi) = \begin{cases} Y_{nm}(\theta,\phi) & (\theta,\phi) \in [0,\pi]\times[0,2\pi] \\ Y_{nm}(2\pi-\theta,\phi+\pi) & (\theta,\phi) \in [\pi,2\pi]\times[0,\pi] \\ Y_{nm}(2\pi-\theta,\phi-\pi) & (\theta,\phi) \in [\pi,2\pi]^2 \end{cases} \tag{3.32}$$

The periodic extension of the normalized spherical harmonics can be represented in terms of a Fourier series, and as a result, the integral

$$\upsilon_{1,nm} = \frac{1}{2}\int_0^{2\pi} d\theta \int_0^{2\pi} d\phi \tilde{\Upsilon}_1(\theta,\phi) \hat{Y}^*_{nm}(\theta,\phi) \tag{3.33}$$

can evaluated exactly using uniform samples of $\tilde{\Upsilon}_1(\theta,\phi)$ that are available to us with a trapezoid rule in both dimensions.

3.3.3.2 Subtleties for vector fields

The algorithm of this chapter uses four trees; one each for component of the magnetic vector potential and one for the scalar potential. This is the point form of MLFMA that addresses integration issues [23]. Alternatively, one can develop MLFMA with either two or three trees. The two tree version is the usual dyadic MLFMA and requires either vector spherical harmonics [62] or the Fourier method modified to account for the anti-symmetry of the transverse fields [53], whereas the three tree version requires mapping onto the three components of electric fields, E_x, E_y, E_z, and employing the scalar filters espoused here. Multiplication of fields of the form (3.22) by the dyad $\hat{\theta}\hat{\theta} + \hat{\phi}\hat{\phi}$ increases the polynomial order by two, requiring a commensurate increase in the size of the integration rule.

3.3.4 Cross-level interactions

Next, non-uniform tree introduces the notion of cross-level interactions. Addition theorems used derive either MLFMA of ACE are invalid for these interactions. To circumvent this issue, we subsume either $\mathbf{d}_o$ or $\mathbf{d}_s$, depending on the interaction, into the translation heading $\mathbf{X}$. If the source box Ω_s is smaller than the observer box Ω_o, i.e. $|\Omega_s| < |\Omega_o|$, the source multipole expansion is valid everywhere within Ω_o, meaning $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{d}_o$. The source multipole expansion is consequently translated from its expansion center $\mathbf{r}_s^c$ directly to each particle within Ω_o. Such interactions are tabulated in the *X-list*. Conversely, if $|\Omega_s| > |\Omega_o|$, we take $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{d}_s$, translating each particle within Ω_s to the center $\mathbf{r}_o^c$ of Ω_o, adding to the local expansion. These interactions are tabulated in the *W-list*. Both of these forms of the addition theorem are always valid for well-separated interactions. The essential criterion is that the two boxes are separated by at least one box of the same size as the smaller interacting box.

The interaction "type" is determined by the field representation (ACE or MLFMA) of the smaller of the two boxes. Consider evaluation of (3.13) between source and observation points in Ω_s and Ω_o, respectively. Let $\mathbf{M}^{(n)}$ be the ACE multipole tensor representing the sources within Ω_s centered at $\mathbf{r}_s^c$. Then we may write

$$\Psi(\mathbf{r}) = \begin{cases} \sum_{n=0}^{\infty} \mathbf{M}^{(n)} \cdot n \cdot \nabla^n g(\mathbf{r} - \mathbf{r}_s^c), & |\Omega_o| > |\Omega_s| \\ \sum_{n=0}^{\infty} (\mathbf{r} - \mathbf{r}_o^c)^n \cdot n \cdot \mathbf{L}^{(n)}, & |\Omega_o| < |\Omega_s| \end{cases} \tag{3.34}$$

where

$$\mathbf{L}^{(n)} = \sum_{i=1}^{N} \frac{u_i}{n!} \nabla^n g(\mathbf{r}_o^c - \mathbf{r}_i). \tag{3.35}$$

In the MLFMA, we have

$$\Psi(\mathbf{r}) = \begin{cases} \int_{S^2} T(\mathbf{k}, \mathbf{r} - \mathbf{r}_s^c) V(\mathbf{k}, \mathbf{r}_s^c) d^2\hat{k}, & |\Omega_o| > |\Omega_s| \\ \int_{S^2} e^{-j\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}_o^c)} U(\mathbf{k}, \mathbf{r}_o^c) d^2\hat{k}, & |\Omega_o| < |\Omega_s| \end{cases} \tag{3.36}$$

where $V(\mathbf{k}, \mathbf{r}_s^c)$ is the source multipole expansion, and

$$U(\mathbf{k}, \mathbf{r}_o^c) = \sum_{i=1}^{N} u_i T(\mathbf{k}, \mathbf{r}_o^c - \mathbf{r}_i) \tag{3.37}$$

represents the local expansion within Ω_o of the sources. As with the interaction type, the band limit for the translation operator T is determined by the smaller of the two boxes.

It should be noted that the ACE translation operator is valid only over electrically-small distances [73], so care should be taken to ensure that translations from a box at the coarsest ACE level to points within MLFMA boxes fall within this radius. For this purpose, 2:1 balancing of the octree [68] at this level suffices. We also note that the minimum translation distance for X- and W-list interactions falls from $2D(l)$ to $1.5D(l)$ if all particles exist inside the interacting boxes, limiting the range of the MLFMA truncation parameter N_h for such interactions.

3.3.5 Interpolation of MLFMA translation operators

To reduce the storage overhead associated with translation operators, interpolation in $\hat{k}\cdot\hat{X}$ is typically used [65]. However, this scheme is untenable for large numbers of X- and W-list interactions, as it requires the number of unique translation distances to be small. To ameliorate this cost, we can also interpolate in the argument of the spherical Hankel function to further reduce the storage without significantly reducing the accuracy of the MLFMA. Using the well-known series representation

$$\begin{aligned} h_n^{(2)}(z) =& \frac{e^{-jz}}{z} \sum_{k=0}^{n} \frac{(-j)^{k-n-1}(n+k)!}{2^k k!(n-k)!} z^{-k} \\ =& \frac{e^{-jz}}{z} \tilde{h}_n^{(2)}(z) \end{aligned} \tag{3.38}$$

It is well known that we may pull out a phase-magnitude term independent of n, obtaining $h_n^{(2)}(z) = (e^{-jz}/z)\tilde{h}_n^{(2)}(z)$. It is then common to all elements of the series T, and we may implicitly define a smoother, phase- and amplitude-compensated translation operator $\tilde{T}$:

$$T(\mathbf{k}, \mathbf{X}) = \frac{e^{-jkX}}{kX} \tilde{T}(\mathbf{k}, \mathbf{X}). \tag{3.39}$$

We employ the Lagrange interpolating polynomials $\ell_n(x)$ of order p. For a given level, let kX_{min}, kX_{max} denote the minimum and maximum translation distances, respectively. The interpolation nodes are chosen to give a high density near kX_{min} and a lower density near kX_{max} because i) the function to be interpolated has $O((kX)^{-K})$ behavior, and ii) this is typically good practice for polynomial interpolation. The nodes are chosen as follows. Let $a = kX_{min} - \Delta_1$ and $b = kX_{max} + \Delta_2$ be the beginning and end of the sampling interval respectively, where Δ_1, Δ_2 are small extensions of the interval, and J be the number of kX values to interpolate over. Then, for $j = 1, \ldots, J$, we let

$$x_j = a + (b - a)\left[1 - \cos\left(\frac{(j-1)\pi}{2(J-1)}\right)\right] \tag{3.40}$$

denote the jth interpolation node. Let p denote the order of interpolation polynomials used. The interval extensions are chosen as

$$\begin{aligned} \Delta_1 &= p\,(kX_{max} - kX_{min})\left[1 - \cos\left(\frac{\pi}{2(J-1)}\right)\right] \\ \Delta_2 &= p\,(kX_{max} - kX_{min})\cos\left(\frac{(J-2)\pi}{2(J-1)}\right) \end{aligned} \tag{3.41}$$

so that the extensions are the size of only p subintervals at the beginning and end. This prevents extension of the interpolation interval into the explosion zone of the spherical Hankel functions.

We store samples of the compensated translator $\tilde{T}$ for level l in a matrix $\mathbf{T}_l$ defined as

$$[\mathbf{T}_l]_{mn} \doteq \tilde{T}\left(\cos\beta_m, \frac{x_n}{k}\right), \tag{3.42}$$

where $\{\beta_m\}$ is a set of uniformly-spaced samples on $\left[-q\Delta_\beta, \pi + q\Delta_\beta\right]$, with interpolation order q and spacing Δ_β. Then, for each translation distance kX in the interaction list at level l, we precompute the interpolation weights $\{w_j = \ell_j(kX)\}, j = 0, \ldots, p$ for reconstructing the compensated translator $\tilde{T}$ from the columns of $\mathbf{T}_l$ as

$$\tilde{T}(\cos\beta_m, kX) \approx \sum_{j=0}^{p} w_j\,[\mathbf{T}_l]_{m(Q+j)}, \tag{3.43}$$

where Q is the index of the first-occurring sample x_Q used in the interpolation, determined by binary search. During the evaluation phase, (3.43) is evaluated to obtain the samples for the

Table 3.1: Comparison of L_2 relative error data for translation operator interpolation with and without compensation

Sampling	ϵ_{max}	ϵ_{min}	ϵ_{avg}	$kX/D(l)$ at ϵ_{max}
Uniform	7.38e-3	0	1.43e-3	1.81411
Cosine	2.35e-3	2.39e-7	4.16e-4	3.52377
Uniform+comp.	3.28e-3	0	5.59e-4	1.81411
Cosine+comp.	2.05e-4	1.10e-7	7.50e-5	2.3004

interpolation in $\hat{k} \cdot \hat{X}$. The result of this interpolation is then multiplied by the phase-magnitude factor e^{-jkX}/kX to complete the process. The complexity is therefore increased only by a constant factor of $(p+1) \sim O(1)$. Table 3.1 summarizes the reconstruction error over a range of translation distances normalized by box size using $2N_h(l)$ samples and $p = 3$. The average error is reduced by a factor of two when switching from uniform sampling to cosine sampling, and another factor of two is gained by phase-amplitude compensation. The maximum error at any point within the interval is reduced significantly by the approach described here.

3.4 Summary of parallel wideband MLFMA algorithm

In this section, we provide a brief outline of the parallel ACE-AP-MLFMA algorithm used to obtain the numerical results presented in the next section (more details will be provided in a subsequent publication). The parallel algorithm we use is based on that presented in [48] with a number of important improvements. We would like to point out that this algorithm is effectively a bottom-up tree construction/partitioning algorithm, and compared to the top-down hierarchical partitioning schemes used in [28, 50], gives us important advantages in terms of data partitioning and load balancing as discussed below.

3.4.1 Tree construction

Our bottom-up tree construction starts by assigning points to leaf boxes according to a prescribed leaf level box size (essentially creating a uniform tree to begin with). Using a parallel bucket-sort algorithm, these leaf boxes are then partitioned into contiguous chunks with the objective

of balancing the number of points assigned to different processes. After this initial partitioning, each process constructs the upper levels of its own tree based on its leaf boxes. This scheme will obviously result in sparsely populated leaf nodes given the non-uniform particle distributions in our target problems. Therefore, starting at the leaf level and working up the tree, each process merges its sparsely populated tree nodes. To be precise, if the total number of particles owned by the children of an internal tree node is below a predetermined threshold s_{max}, then those leaf nodes are merged at the parent node, making the parent a new leaf node. Since the octree is distributed across processes, such merge operations are performed recursively at the higher levels in coordination with neighboring processes. Interaction lists are subsequently computed on the merged tree.

3.4.2 Load balancing

Even distribution of the computational work is crucial to achieve scaling to a large number of processes. Two major issues make load balancing for the MLFMA algorithm challenging: i) Electrically large objects result in deep trees that contain a significant amount of work across a small number of high level tree nodes, and ii) the non-uniform octree structure which is a natural consequence of the non-uniform point distribution has a work profile with high variability.

We address the first issue by inheriting the *adaptive direction partitioning* idea, first presented in [48]. Note that the bottom-up partitioning approach described above results in overlapped tree regions between processes, *i.e.*, internal tree nodes at process boundaries may appear in the partial trees of multiple processes. We call such nodes as *duplicate nodes*, and to balance the load in deep trees, we distribute the multipole expansion data for these nodes as well as the computations (inter/anterpolations, translations) associated with them evenly across all duplicating processes. As will be discussed in our subsequent paper, the present work significantly improves the implementation of the basic direction partitioning idea by leveraging better parallelism, non-blocking collective communication primitives, and an efficient distributed execution schedule.

The load balancing issue associated with non-uniform trees is addressed using an empirical cost evaluation technique. Specifically, each process computes the interaction lists for non-duplicate

nodes within its own partial tree and evaluates how much time it takes to perform a sample set of kernel operations (inter/anterpolations and translations) on the given hardware. It then assigns a computational load to each tree node, starting from the highest level by taking into account the empirical kernel costs and the number of interactions each tree node has. These costs are then percolated all the way down to the leaf nodes. Finally, the non-uniform MLFMA tree is repartitioned such that the empirical costs of the contiguous chunks of leaf nodes assigned to each process is evenly distributed.

3.4.3 Parallel evaluation

Evaluation of the *matvec* consists of 3 main phases: i) M2M, upward traversal of the tree to compute the multipole data for each tree node, ii) M2L, lateral translation of multipole data to effect far-field interactions, and iii) L2L, downward traversal of the tree to disaggregate local expansions to the leaf nodes and evaluate fields at observers.

Parallelism is applied in three different forms in the M2M phase. Each process first performs the serial computations for their own unique interior nodes, which are nodes that strictly belong to a single process. Secondly, plural nodes in the ACE and spherically sampled levels are handled using coarse-grained parallelism, where all processes sharing a plural node compute the interpolations for the children they own sequentially and reduce or reduce-scatter, for ACE and MLFMA respectively, the results among all sharing processes using non-blocking MPI collectives. As the third form of parallelism, duplicate nodes above the spherical-to-uniform transition level are interpolated using a fine-grained parallel version of the FFT-based interpolation algorithm. The partial multipole data for each duplicate node in this category is shifted, summed, and partitioned over the duplicating processes in parallel.

Translations in the M2L phase follow a similar strategy as above, where all processes begin with sequentially computing translations within their own trees, then exchange multipole data with neighboring processes and process them as they arrive – effectively overlapping computation and communication. Due to the presence of cross-level interactions in non-uniform trees, these com-

putations are organized into three subphases where X-list translations are performed by evaluating the spectral integral at each basis function in the first subphase, V-list translations are computed in the second subphase, and W-list translation contributions to the local expansions are computed in the third subphase. We note that one important shortcoming of the present parallel M2L implementation is that all translation data for duplicate nodes need to be communicated to the *resident process*, *i.e.*, the designated receiving process for each duplicate node, which then applies them one by one – this can potentially create performance bottlenecks for high level nodes in deep trees.

Finally, the L2L phase is performed by essentially mimicking M2M in reverse order. As an artifact of the limited parallelization of duplicate nodes mentioned above, L2L phase starts with broadcasting the local expansions of duplicate nodes from the resident process to the duplicating processes. After this initial communication, anterpolation and disaggregation of the duplicate nodes above the spherical-to-uniform transition level are performed in distributed fashion. Once all parallel anterpolations are completed, all processes can independently perform the sequential L2L operations for their internal nodes.

3.5 Numerical Results

3.5.1 Accuracy analysis

In this section, we demonstrate the controllable accuracy of each interpolation/anterpolation method discussed in this paper, namely *spherical*, *uniform*, and *hybrid*. We also discuss trade-offs between these methods in terms of memory and run time. The examples in this subsection were run using a single core on a desktop computer with an Intel Xeon E5-2630 CPU with clock speed of 2.3 GHz and 64 GB RAM. All computations use double precision arithmetic, and FFTW 3.3.5 is used for performing FFTs in the interpolation and anterpolation stages.

3.5.1.1 Hybrid sampling MLFMA error control

To illustrate the accuracy of the proposed hybrid sampling and interpolation/anterpolation method, we consider a uniformly random distribution of 256,000 randomly-oriented dipoles of unit strength

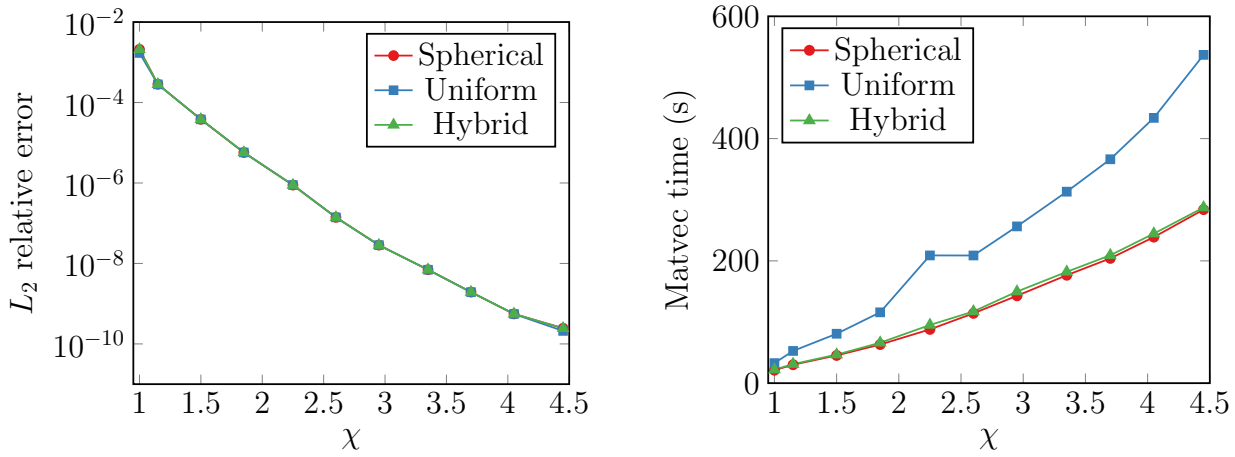


Figure 3.3: Error convergence (left) and *matvec* timings (right) vs. oversampling parameter χ for each sampling/interpolation method in the 8λ cube geometry.

within an $8\lambda \times 8\lambda \times 8\lambda$ cubical domain. The box size is chosen to be 0.25λ, resulting in a 6-level uniform octree. Neither ACE nor interpolation of the MLFMA translation operator is used in this experiment. For the upper levels of the tree, a cap was enforced on the bandwidth to prevent numerical breakdown of the translation operator. Using three buffer boxes, we study the error convergence of the far-field contributions for each of the spherical, uniform, and hybrid methods outlined in the previous section. For the hybrid scheme, the transition was chosen so that only the bottom two levels use the spherical scheme. This data is shown in Fig. 3.3a. As the oversampling parameter χ is increased, each method converges in a nearly identical manner, demonstrating that the hybrid method introduces no error. Fig. 3.3b shows the time taken per *matvec* for each method. The hybrid scheme only adds a few seconds to the baseline time taken by the purely spherical scheme, while the uniform-only scheme takes roughly twice as long on average, as expected. For contrast, direct evaluation of the far-field *matvec* takes 5128.029 s on the same machine.

3.5.1.2 Trade-offs for hybrid sampling

We have already seen the effect of purely uniform sampling on runtime due to the doubled sampling rate throughout the tree. The results for the hybrid sampling scheme suggest that the extra cost of fully-uniform sampling comes principally from performing M2L operations while oversampling at

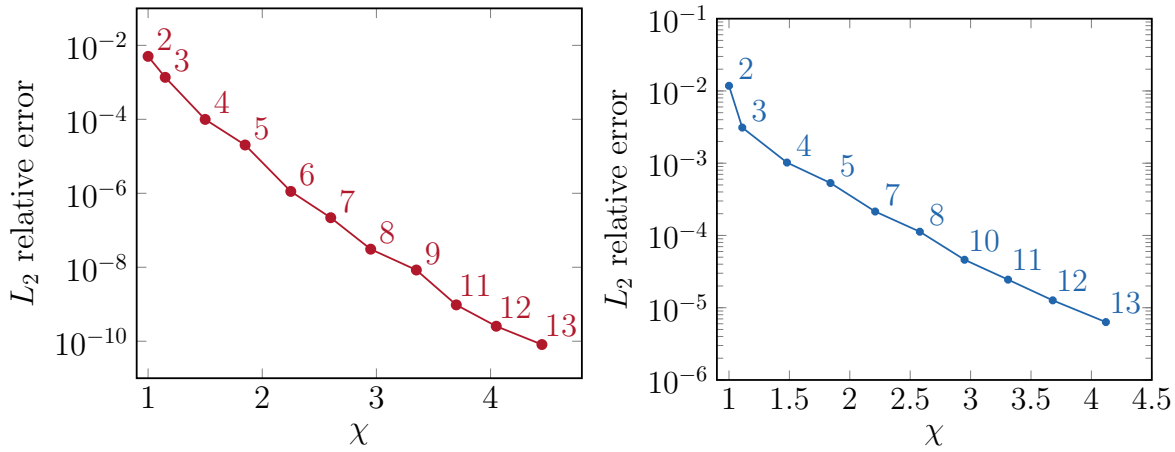


Figure 3.4: Error convergence vs oversampling parameter χ for (a) uniform tree ACE-MLFMA for 64,000 dipoles in a flattened box, and (b) non-uniform tree cone-sphere. The number next to each data point indicates the order P of ACE expansions used. Note that the example in (a) uses three buffer boxes for the far-field, while that in (b) uses only one.

the finest levels of the tree where most boxes exist, as expected. Conversely, the optimal sampling rate for the spherical scheme gives the best run-time, but at the expense of precomputing and storing samples of associated Legendre functions P_n^m to perform interpolation and anterpolation via the spherical harmonics transform. To study the trade-offs, we examine the case of 256,000 dipoles distributed on a $256\lambda \times 256\lambda$ square in the $x-y$ plane. Using leaf-level boxes of diameter $\lambda/4$, the uniform tree is 11 levels deep, and a one-buffer box rule is used. The oversampling rate is fixed at $\chi = 1.0$.

We examine memory consumption and *matvec* time for different values of the finest spherical sampling level L_{su}. Table 3.2 shows the memory required for storing P_n^m samples, multipole and local expansions, and M2L operators, and the serial *matvec* time vs. L_{su}.

While this data is collected using a serial kernel, we must emphasize that in a parallel setting the multipole and local data, which consume the majority of the overall memory, is distributed across processes in a largely non-redundant fashion, while the P_n^m storage is duplicated on every process. M2L operators must also be stored local to each process, though it is not identical on each process. In contemporary HPC environments, the maximum amount of RAM per core is typically around 4 GB, underscoring the importance of finding an acceptable balance between memory consumption

Table 3.2: Memory consumption and timings for different hybrid sampling transitions

L_{su}	3	4	5	6	7
Uni. levels	0	1	2	3	4
Sph. levels	9	8	7	6	5
P_n^m (GB)	6.0	2.0	0.3	0.04	0.005
Multipole (GB)	13.1	14.5	15.9	17.4	18.8
M2L (GB)	0.8	1.4	1.5	1.6	1.6
Matvec time (s)	730	957	1364	1377	1589

and runtime to make most efficient use of computational resources.

3.5.1.3 Uniform-tree wideband MLFMA error control

We now examine the error in the hybrid-sampling MLFMA with ACE, using a uniform tree. We consider the case of 64,000 dipoles distributed non-uniformly within an $8\lambda \times 8\lambda \times 1/16\lambda$ box. The point locations were generated using $(x, y, z) = (8r_1^{1.5}, 8r_2^{1.5}, r_3/16)$, where r_1, r_2, r_3 are random numbers on $[0, 1]$. The smallest box was chosen as $\lambda/16$, yielding an 8-level tree with 2 levels of ACE, 2 levels of spherical MLFMA, and 2 levels of uniform MLFMA. Fig. 3.4a shows error convergence of the far-field as both χ, P are increased, demonstrating fine-grained control over accuracy over both the low- and mid-frequency regimes.

3.5.1.4 Non-uniform-tree wideband MLFMA error control

We now examine the error convergence for the non-uniform wideband ACE-MLFMA applied to a multiscale object and compare with the uniform-tree algorithm. Consider a "cone-sphere" surface geometry comprising a long cone terminated on its wide end by a hemispherical surface. The cone portion is 36.33λ long, and the spherical portion has a radius of 4.49λ. The mesh contains 47,367 triangles, and we placed 6 randomly oriented dipoles within each triangle, totaling 284,202 dipoles. The smallest box size is chosen to be $\lambda/128$, and the tree is merged using $s_{max} = 30$, yielding a 14-level tree with 5 levels of ACE, 3 levels of spherical MLFMA, and 4 levels of uniform MLFMA.

Table 3.3 summarizes some of the stark differences between the uniform and non-uniform trees.

Table 3.3: Comparison of uniform and non-uniform trees for the conesphere geometry

	Uniform	Non-uniform
Nodes	1058881	29562
Leaf level	14	8-14
Dipoles/box	1	12
Inter. lists	1149.16 s	25.05 s

Notably, the pre-computation time required for computing interaction lists decreases by a factor of over 45 as a consequence of the 97% reduction in overall tree nodes. Fig. 3.4b shows convergence in the relative L_2 error of the far-field using the non-uniform tree with a single buffer box rule.

3.5.2 Parallel CFIE solver evaluation

In this section, the kernel evaluation is wrapped in a parallel MoM solver for the CFIE (3.1), and we present results demonstrating its capabilities, all with $\alpha = 0.5$. Parallel GMRES is used as the iterative solver. We note that our use of a four-tree mixed-potential MLFMA as opposed to a two- or three-tree dyadic MLFMA directly results in an increase in runtime.

3.5.2.1 Uniform trees

First, we consider the evaluation of the radar cross section (RCS) of two electrically-large conducting spheres of diameter 256λ and 512λ, discretized with 84,934,656 and 339,738,624 unknowns, respectively. A 7-point integration rule within each patch was used for both source and testing far-field integrals. In each case, the leaf-level box is chosen as 0.25λ, resulting in 11- and 12-level trees. The bottom six levels employ spherical sampling before transitioning to uniform sampling, and $\chi = 1.0$. The incident plane wave is polarized along the $\hat{x}$ direction and traveling in the $+z$ direction. Parallel GMRES with tolerance 10^{-3} and a restart value of 30 is used to solve the resulting matrix system. Both systems converged within two outer iterations. For the 256λ case, the solve took 31 minutes using 2048 processes, requiring 27.8 seconds per *matvec*. The 512λ solve on 2048 processes took 8.5 hours, requiring 5 minutes 44 seconds per *matvec*; in this case we had to resort to Lagrange interpolation of translation operators to alleviate memory bottlenecks

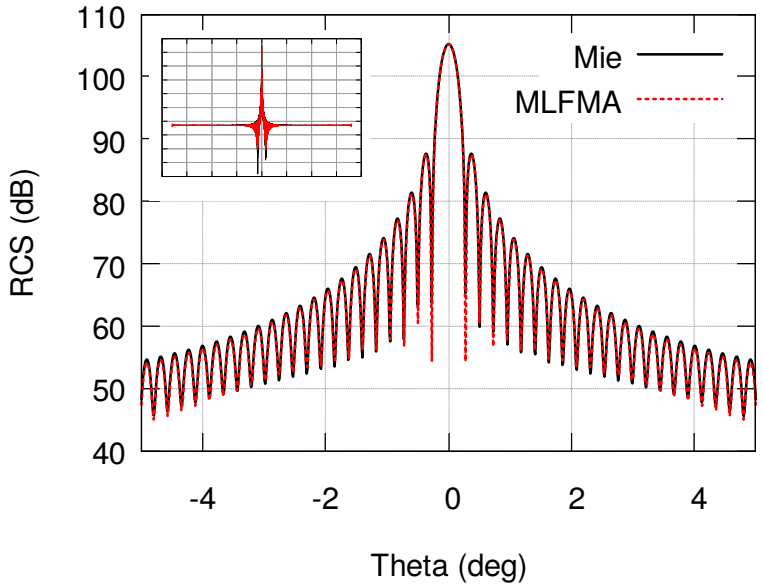


Figure 3.5: Comparison of bistatic RCS calculated by parallel MLFMA vs. analytical for a sphere of diameter 256λ within $\pm 5^o$ of the main lobe.

associated with storing them directly at the uppermost levels. This resulted in approximately 76% of the *matvec* time being spent in V-list evaluation as opposed to 36% in the 256λ case, hence the increase in runtime being larger than the increase in unknown count. The calculated RCS of the spheres are shown in Figs. 3.5 and 3.6 along with the analytical Mie series solution, with which the agreement is very good.

3.5.2.2 Non-uniform trees

Finally, we consider evaluation of RCS from geometries with non-uniform distributions using non-uniform trees. In these cases, the far-field integration rule is set to one point per patch to ease memory demands for storing X and W list operators. First, we demonstrate scattering from a small 20λ diameter sphere with a $\lambda/10$ discretization everywhere except for the $\theta = 0$ pole, where a much finer discretization is used, using 2,053,947 unknowns overall. Using a minimum box size of $1.95 \times 10^{-3}\lambda$ and $s_{max} = 35$ results in a 15-level tree with 12 unknowns per leaf box on average. The simulation parameters for MLFMA and ACE were $\chi = 1.5$ and $P = 4$. Using 28 processes and GMRES tolerance of 10^{-3}, the solution took 38 minutes at about 15 s per *matvec*. The calculated

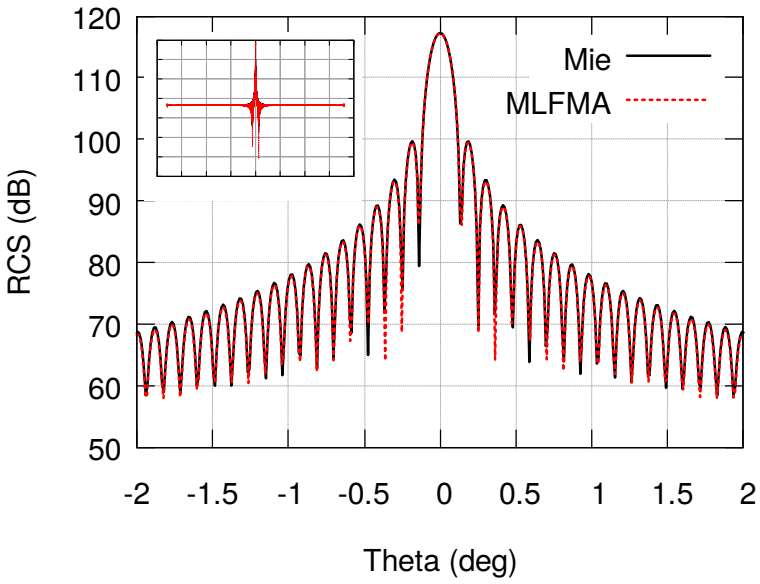


Figure 3.6: Comparison of bistatic RCS calculated by parallel MLFMA vs. analytical for a sphere of diameter 512λ within $\pm 2^o$ of the main lobe.

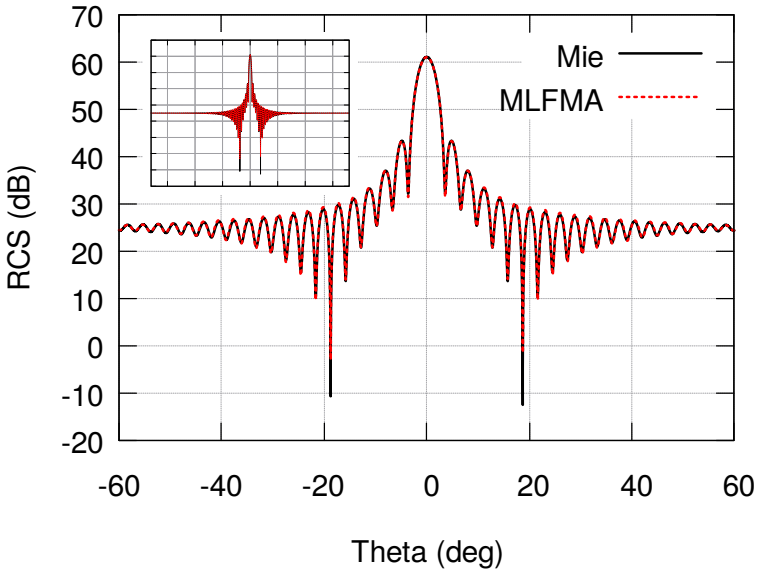


Figure 3.7: Comparison of bistatic RCS calculated by parallel MLFMA vs. analytical for the 20λ sphere with densely discretized pole.

RCS is shown in Fig. 3.7 and compared with the Mie series solution, with which it shows excellent agreement.

Next, we consider electrically-large objects with sharp corners or other features with regions of high density of unknowns. Our first example is an arrowhead-shaped geometry measuring $470\lambda \times 154\lambda \times 79\lambda$, discretized using 77,257,728 unknowns. A 14-level non-uniform tree is used

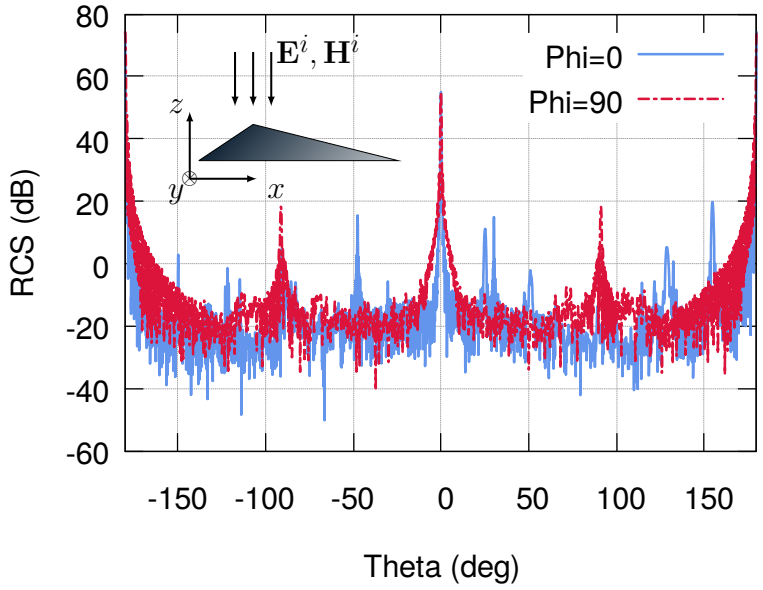
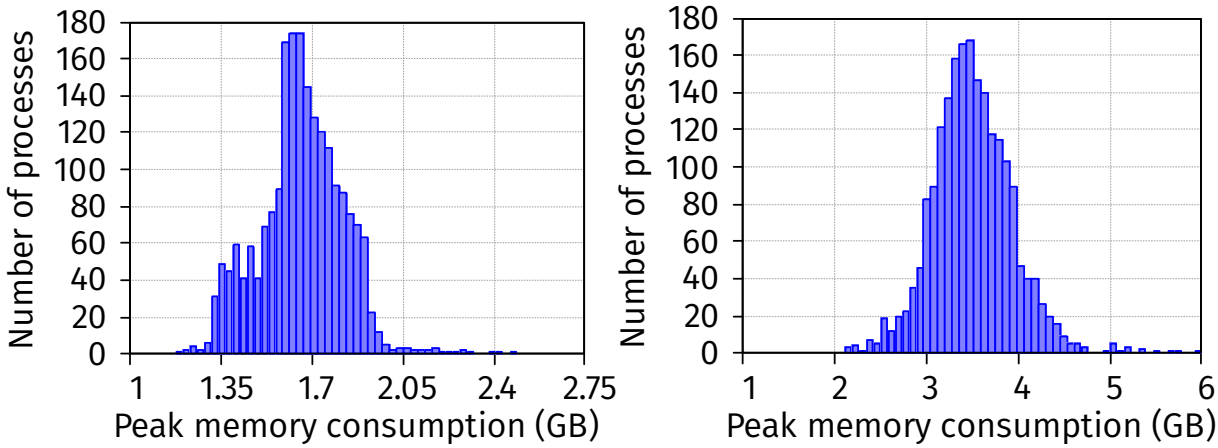


Figure 3.8: RCS calculated with parallel MLFMA for 470λ arrowhead geometry.



(a) Arrowhead (b) Airplane

Figure 3.9: Memory utilization histograms for (a) 470λ arrowhead and (b) 755λ airplane geometry.

with minimum leaf box size of 0.0574λ and $s_{max} = 35$, resulting in 10 unknowns per leaf, on average. The RCS is shown in Fig. 3.8, along with an illustration of the geometry. The incident electric field is given by $\mathbf{E}^i = \hat{x}e^{jkz}$. Solution with tolerance 5×10^{-3} took 19.3 minutes on 2048 processes, at 23.6 s per *matvec*. A histogram of peak memory utilization on each process is shown in Fig. 3.9.

Our last example concerns scattering from the 755λ airplane geometry depicted inset in Fig. 3.10. A $\lambda/10$ discretization of the surface yields 285,425,664 unknowns. The incident field is

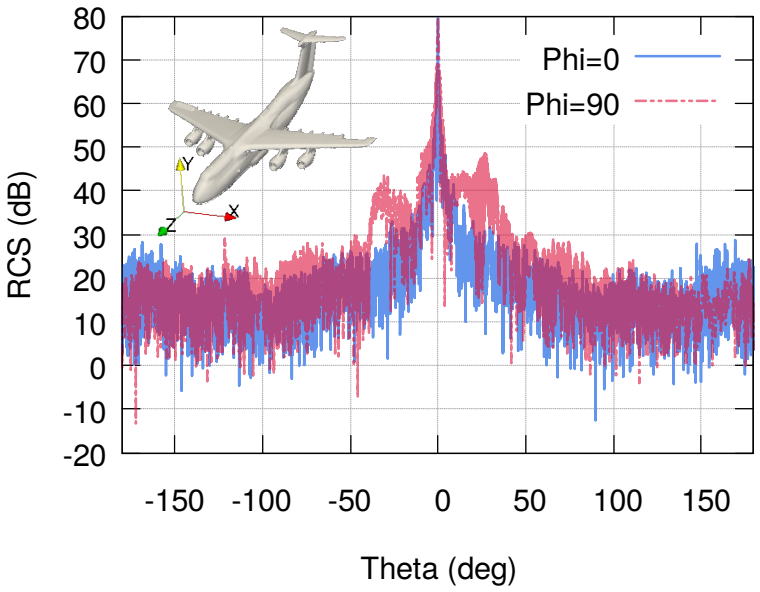


Figure 3.10: RCS calculated with parallel MLFMA for 755λ airplane geometry.

given as $\mathbf{E}^i = \hat{x}e^{-jkz}$. Using a minimum leaf size of 0.0231λ, the non-uniform tree spans 16 levels and is merged with $s_{max} = 35$, increasing the average number of unknowns per box from 1 to 18 and distributing leaves across the bottom four tree levels. ACE ($P = 3$) is used for boxes smaller than $\lambda/4$, and uniform MLFMA is used at the three uppermost levels. The densest box at the finest level contains 40 unknowns. Using GMRES with a tolerance of 5×10^{-3}, the solution on 2048 processes took 3.17 hours, using 1 minute 8 seconds per *matvec*. Memory utilization for this computation is depicted in Fig. 3.9. The RCS in the $\phi = 0^o, 90^o$ cuts is shown in Fig. 3.10.

3.6 Conclusion

We have presented an extremely wide-band parallel MLFMA with fine-grained control over the error in field evaluation and methods for reducing both storage and computational cost while sacrificing nothing in terms of accuracy. We introduced rigorous operators for computing interactions at any level on a non-uniform octree which is adapted in parallel to fit to the geometry. We also introduced a rigorous transition from spherical harmonics-based to Fourier-based inter/anterpolation of multipole expansions to optimize storage and solution time. An array of numerical examples demonstrate the accuracy and efficiency of the algorithm, and several scattering examples demon-

strated the ability of the solver to accurately solve problems on large, complicated objects with non-uniform spatial distributions of unknowns.

CHAPTER 4

PARALLELIZATION TECHNIQUES FOR THE NON-UNIFORM WIDEBAND MLFMA

4.1 Introduction

In this chapter, we detail an efficient parallelization strategy for the non-uniform wideband MLFMA presented in the previous chapter. The algorithm is based on that presented in [48], which we will show suffers from several serious computational bottlenecks when applied to progressively larger problems. This work clearly identifies the bottlenecks and their causes, and offers algorithmic remedies for overcoming them.

4.2 Problem Statement

Consider a collection of N point sources $u_n \in \mathbb{C}$ located at points $\mathbf{r}_n \in \mathbb{R}^3$, $n = 1, \ldots, N$. The Helmholtz potential $\Phi(\mathbf{r})$ at some observation point $\mathbf{r}$ is then given by

$$\Phi(\mathbf{r}) \doteq \sum_{n=1}^{N} g(\mathbf{r} - \mathbf{r}_n) u_n, \tag{4.1}$$

where the Green's function g for the Helmholtz equation is given by

$$g(\mathbf{r}) = \frac{e^{-ik|\mathbf{r}|}}{4\pi|\mathbf{r}|}, \tag{4.2}$$

where $k = 2\pi/\lambda$ denotes the wavenumber in rad/m and λ denotes the wavelength in meters. Sums of the form (4.1) often arise in the discretization of integral equations in electromagnetics and acoustics. Typically, the evaluation of $\Phi(\mathbf{r}_m), m = 1, \ldots, N$ is required, taking the form of a classical N-body problem with kernel g. As such, the $O(N^2)$ cost of direct (exact) evaluation of these sums becomes prohibitively expensive for large N. Instead, one often employs the multilevel fast multipole algorithm (MLFMA) [66] to approximate these quantities in $O(N \log N)$ time with the ability to control the accuracy of the approximation to arbitrary precision.

Similar in structure to the fast multipole method for the Laplace potential [34], the MLFMA employs a hierarchical tree data structure for rapidly computing interactions between clusters of

particles via alternate representations called multipole and local expansions. First, the computational domain is recursively subdivided into cubes (boxes) and each particle is mapped to the box in which it resides. The hierarchy of this recursive subdivision procedure is modeled as a tree (see Fig. 4.1), specifically an *octree* in which each subdivision results in the creation of eight boxes of half the diameter. The evaluation of (4.1) using the MLFMA is broken up into several stages of computation:

1. **Particle-to-particle (P2P)**: For each pair of spatially adjacent leaf (childless) boxes, evaluate interactions between all contained particles.

2. **Charge-to-multipole (C2M)**: Multipole expansions are created for each leaf box from their constituent particles.

3. **Multipole-to-multipole (M2M)**: Multipole expansions of non-leaf boxes are created by combining multipole expansions of their children.

4. **Multipole-to-local (M2L)**: The local expansion for each box is created by converting multipole expansions of boxes in the far field (defined momentarily) into local expansions and summing the results. The far field of a box b is defined as the set of children of b's parent's adjacent neighbors which are not adjacent to b.

5. **Local-to-local (L2L)**: The local expansion of each box b is updated with information from its ancestors in a top-down manner so that the influence of all particles in non-adjacent boxes is present in the final local expansion of b.

6. **Local-to-observer (L2O)**: For each box, the local expansion is used to calculate the potential at the location of each particle in that box.

Large values of N are most often encountered in problems that are either densely discretized to capture fine geometric features or electrically-large, i.e. the principal dimension of the geometry is hundreds or thousands of wavelengths long. Multiscale problems, i.e. problems featuring both of these characteristics, are very difficult to solve and require a robust computational approach for

the efficient evaluation of (4.1). The key to such an approach is to understand the behavior of the kernel g over a hierarchy of length scales, and to then design seamless transitions between methods exploiting this behavior in each regime. Realistic problems in modern computer-aided design and analysis are indeed multiscale, requiring discretizations with N in the millions to potentially billions. In such large problems, operations on the octree at the coarsest levels can become very expensive, underscoring the need for an efficient parallel algorithm.

4.2.1 Summary of non-uniform wideband MLFMA

In this section, we briefly summarize at a high level the non-uniform wideband MLFMA presented in the previous chapter. To begin, we assume that there exists an octree structure with L levels enveloping the computational domain. Level 1 refers to the *root* of the octree, i.e. the box containing the entire computational domain, and level L is the level of finest refinement. The box diameter at level l is denoted $D(l)$, and $D(L)$ is specified (in units of wavelengths) as an input parameter to the algorithm. The root box diameter is $2^{L-1}D(L)$. The octree is non-uniform, i.e. leaf (childless) nodes may exist at any level of the tree, and each leaf contains approximately the same number of particles per box up to a maximum of s_{max}.

4.2.1.1 Low-frequency regime

At small length scales relative to a wavelength, e.g. $k|\mathbf{r}| < \pi/2$ or $|\mathbf{r}| < \lambda/4$, the Helmholtz kernel g does not exhibit oscillatory behavior and is instead quite smooth. Away from the singularity, g is well-approximated over electrically short distances by a low-order polynomial expansion. The accelerated Cartesian expansion (ACE) method exploits this fact to evaluate the potential in these regimes in $\mathcal{O}(N)$ time. Within a domain Ω_o sufficiently far from a cluster of sources Ω_s, the potential Φ due to the sources within Ω_s is represented using an order-P Taylor series about the center of Ω_o. As length scales increase, the ability to accurately compute the potential via ACE diminishes and one must transition to MLFMA, as defined in [73].

4.2.1.2 High-frequency regime

For boxes of diameter larger than $\lambda/4$ the usual MLFMA is employed. Multipole and local expansions are functions defined on the unit sphere $(\theta, \phi) \in [0, \pi] \times [0, 2\pi]$ and sampled at locations $(\theta_i, \phi_j), i = 1, \ldots, N_\theta, j = 1, \ldots, N_\phi$. The sampling rates for a given box at some level l are governed by its bandwidth $K(l)$, defined by $K(l) = \lfloor \chi \sqrt{3} k D(l) \rfloor + 1$, where $\chi \geq 1.0$ is a parameter that to a large extent controls the accuracy of the MLFMA.

The MLFMA requires interpolation and anterpolation (decimation) of the sampled multipole and local expansions in the M2M and L2L stages, respectively. Three approaches to this task dominate the MLFMA landscape: i) spherical harmonics transforms, ii) fast Fourier transforms, and iii) local interpolation. Methods i) and ii) are exact and thus preferable to method iii), which requires significant oversampling and a properly band-limited interpolation function to achieve relative accuracy beyond a digit or two. The downside to the exact methods is their higher computational complexity; spherical harmonics and Fourier transforms require $O(K(l)^3)$ and $O(K(l)^2 \log K(l))$ operations whereas local interpolation is $O(K(l)^2)$, albeit with a large constant proportional to the square of the one-dimensional oversampling rate.

The method based on spherical harmonics transforms, while computationally expensive as $K \to \infty$, facilitates optimal (minimal) sampling rates on the unit sphere, or $N_\theta(l) = K(l)+1, N_\phi(l) = 2K(l) + 1$, and is used for several levels immediately above the ACE regime. Samples in θ are located at Gauss-Legendre nodes, and samples in ϕ are uniformly spaced. While the optimal sampling rate helps to minimize costs associated with M2L, the $O(K(l)^3)$ storage and complexity become untenable in the upper reaches of the tree. Instead, above some level l_T, a switch is made to the Fourier-based method which has no memory overhead [60, 39]. For these boxes, the samples are spaced uniformly in both variables, and $N_\theta(l) = 2K(l) + 1, N_\phi(l) = 2K(l) + 2$. The storage and M2L cost are doubled per node in this regime, so one should seek a value of l_T with the best trade-off between memory and computational time.

4.2.1.3 Interactions in the adaptive algorithm

Using a non-uniform tree complicates matters somewhat. Using the classic FMM terminology [34], the list of box pairs involved in P2P calculations is called the U-list, and the list of box pairs involved in M2L calculations is called the V-list. In the adaptive algorithm, there are two more lists which tabulate interactions between boxes of different size. The X-list tabulates interactions wherein the source box is smaller than the observer box, and the W-list is its transpose. Interactions in the X-list are carried out by directly computing the potential due to particles in the source box at each particle location in the observer box using the source box's multipole expansion; W-list interactions involve computing local expansions directly from particles in the source box. Obviously, observer boxes in the X-list (sources in the W-list) must be leaf boxes.

4.3 Parallelization of the MLFMA

4.3.1 Tree construction and setup

Let N_p denote the number of processes used in the computation. We begin by distributing the N particles evenly across all processes and determining the diameter D_0 of the cube bounding the entire computational domain. The number of levels in the tree with finest box diameter $D(L)$ is then calculated as the smallest integer L such that $L \geq \log_2(D_0/D(L)) + 1$. Once the number of levels is known, every particle is assigned a Morton key [76]. A parallel bucket sort on the Morton keys is then used to approximately evenly distribute particles across processes at the granularity of leaves. This is done by selecting $N_p - 1$ Morton keys, or "splitters", which chop the Morton Z-curve into N_p contiguous segments. Leaves are uniquely assigned to processes using these splitters. On each process, all ancestor keys of the leaves up to the root are determined, and the leaves and ancestors are stored in post-order, comprising the *local subtree*.

The distributed tree resulting from this bottom-up partitioning scheme generally reflects the geometry's inherent work distribution and provides the flexibility of modifying the partitioning if the workload is imbalanced. As will be discussed, this is often the case for multiscale geometries,

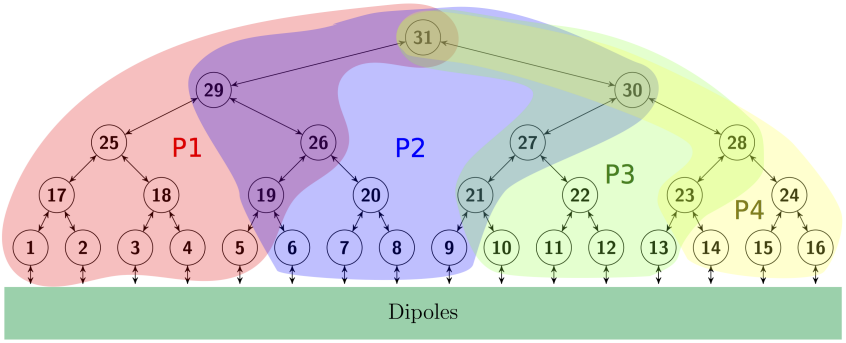


Figure 4.1: Illustration of distributed tree.

wherein the density of particles (and consequently the workload) varies significantly throughout the computational domain. This stands in contrast to the top-down partitioning scheme employed in the hierarchical partitioning (HiP) versions of the MLFMA [28, 51]. While the workload is balanced by design at the uppermost levels of the tree, the distribution of nodes at the lower levels is inherited from this rigid distribution of large nodes. As a result, the HiP methods perform well for relatively uniform particle distributions, but they may produce highly skewed trees and work distributions when applied to multiscale geometries.

4.3.2 Plural nodes

Once consequence of this partitioning strategy is the appearance of *plural nodes*, i.e. nodes which appear in the local subtrees of multiple processes. For a plural node b, the process with the highest rank is called the *resident process* and as such is designated the receiver for V-list operations, and consequently the root for broadcast operations prior to the L2L stage. The resident process' copy of b is called a *shared node*. The processes which own copies of b but are not the resident process are called *users* of b (denoted by the set $U(b)$), and their copies are called *duplicate nodes*. Within the local subtree of any process, its duplicate nodes are guaranteed to appear consecutively at the end of the post-order sequence, while shared nodes may in general appear anywhere in the sequence (except within a duplicate node chain) but tend to appear toward the beginning. As discussed later, these properties are helpful for overlapping communication with computation.

To mitigate the duplication of such nodes and their data, some distribution of the work associated with the node should be undertaken. In the low-frequency/ACE regime, the workload associated with a single plural node is considered negligible; however, in the upper-mid and high frequency regimes the workload may be substantial. The adaptive direction partitioning scheme for plural nodes was introduced in [48] to balance communication load in V-list evaluation for plural nodes in the MLFMA regime. Multipole expansions of plural nodes are sent to their resident processes, where they are summed and equidistributed back to all owners. This method was shown via numerical experiments to improve the load balance of the potential evaluation.

4.3.3 Adaptive tree

In the case of a spatially non-uniform distribution of particles it is computationally advantageous to prune regions of the tree where the number of particles per box is close to 1, as discussed in previous chapters. This process is called *merging* the tree. To begin, the number of particles contained within each non-leaf node in the local subtree is calculated in level-by-level fashion. At this stage, the plural nodes contain only the *local* particles counts, so a parallel sum is required to account for the particles contained within remote descendants.

The tree is then merged from the bottom up, level by level, using a voting mechanism. All tree nodes start out with an indeterminate vote. A maximum number s_{max} of particles per box is chosen. For each node at level L, the number of particles contained within the node is compared to s_{max}. If this number exceeds s_{max}, the node's vote is changed to 'veto' the merge, and its siblings follow suit. This vetoing influence is called *preclusion*. Otherwise, the box's vote is changed to 'merge' unless it has been precluded. As a result, all siblings must share the same vote. Here, the votes of duplicate nodes and siblings straddling process boundaries are synchronized across processes, with vetoes taking precedence. Moving to the next coarsest level $L-1$, this process is repeated level by level until level 3 is reached, where merging cannot take place. The result of this process is a pruned version of the distributed tree where leaves (childless boxes) can exist at any level and contain approximately s_{max} particles each.

4.3.4 Interaction lists

Once the tree is fully formed, the U, V, W, X-lists [32, 15] may be determined. First, define the *neighbors* of a box b as those boxes at the same level as b which share a face, edge, or vertex with b. For each node b in the local subtree, two sets of Morton keys are formed: i) the *near-field*, consisting of all possible neighbors ($3^3 = 27$ in total) of b, and ii) the *far-field*, comprising all children of the neighbors of b's parent which are not neighbors of b ($6^3 - 3^3 = 189$ in total). The different lists are computed as follows.

1. U**-list.** For each local node b, the keys in b's near-field are sent along with the Morton key representing b to their resident processes, as determined by the splitters. For each received key r generated by b, a top-down search of the local subtree is performed. If r is found, the interaction (b, r) is added to the U-list. If the search dead-ends on an ancestor A_r of r, and A_r is a leaf, then (b, A_r) is added to the U-list, and a flag is placed on this interaction. Otherwise, the interaction is discarded. For each flagged interaction, the reciprocal is added by sending the keys A_r, b to the resident process for b and adding (A_r, b) to its U-list. A return-to-sender approach suffices here, since leaves cannot be plural nodes.

2. V, W, X**-lists.** In the same manner as the near-field, far-field keys are exchanged and sought using top-down search. Consider the potential interaction (b, r). If r is found, then (b, r) is added to the V-list of r's resident process. If an ancestor A_r of r is found and it is both a leaf *and* not adjacent to b, then (b, A_r) is added to the X-list. Otherwise, the interaction is discarded. The W-list is determined by transposing the X-list, though a simple return-to-sender approach is not sufficient for this purpose. In W-list interactions, the node A_r serves as the source and b is the observer, so the interaction (A_r, b) must be sent to the resident process for b, which in general may not be the originator of the interaction (b, A_r).

4.3.5 Evaluation of the potential

Here, we describe the non-uniform version of the wideband MLFMA presented in [48] and identify several performance issues in the original implementation which will be addressed in the following sections. The parallel evaluation of (4.1) using the non-uniform wideband MLFMA comprises three stages: i) an upward pass through the tree, ii) translation of multipole expansions into local expansions, and iii) a downward pass of the tree to compute the potential at each observer.

First, we discuss the upward pass. On each process, the local subtree is traversed in post-order as multipole expansions are created via either C2M or M2M operations as described above. All operations are performed serially. When this is done, the multipole expansions of plural nodes in any local subtree are incomplete, as they are missing contributions from particles owned by remote processes. For plural ACE nodes, user processes send their multipole expansions to their resident processes, which sum the expansions and take full computational responsibility for their shared nodes until the L2L stage. For plural MLFMA nodes, resident processes similarly aggregate and sum multipole expansions from their users, but the results are distributed over all owning processes according to the adaptive direction partitioning scheme. This is referred to as the multipole update phase.

Next, we consider the translation (M2L) stage. A round-robin communication scheme uses packets of fixed size to exchange segments of multipole expansions from source processes to observer processes, where this data is used to carry out V- and X-list interactions and is then overwritten with the next segment, until all interactions are exhausted. Interactions where both source and observer boxes exist in a single process' local subtree are then carried out serially. Communication is overlapped with computation by making use of asynchronous MPI primitives. Next, particle strengths u_n are exchanged from sources to observers for the evaluation of W-list interactions.

The final stage is the downward pass. Because resident processes act as a "magnet" for all V, W-list interactions (all X-list observer boxes are leaves and thus cannot be plural nodes), only the resident process of a plural node possesses its complete local expansion. The complete expansions

are first sent to all users in what is called the local update phase, and the remaining work of the downward pass is then completely serial. The local subtree is traversed top-down in reverse post-order, updating local expansions via the L2L operations discussed above. Finally, the complete local expansions at leaf boxes are used to compute the potential observed at each particle via the L2O operation.

4.3.6 Bottlenecks

While the uniform version of this algorithm exhibited good performance for problems of modest size in [48], there are several performance bottlenecks which must be overcome if similar performance metrics are to be obtained for larger problems on larger clusters. First, the multipole and local update phases employ simple point-to-point communication, requiring communication proportional to $N_{user} N_\theta N_\phi$, where N_{user} is the number of users. Since each of $N_{user}, N_\theta, N_\phi$ at the uppermost levels of the tree grow with problem size and increasing process count, this cost can become prohibitive. As a remedy, we propose the use of MPI primitives which implement these communication patterns much more efficiently using tree-like algorithms. This topic is addressed in detail in the next section.

The second bottleneck arises in the serial portions of the upward and downward passes. Especially for electrically-large particle distributions, plural nodes at the uppermost levels of the tree contain a significant amount of data. Not only is the redundant serial computation of M2M and L2L operations for these nodes unnecessary, it also results in significant load imbalances caused by the fact that resident processes for these nodes typically have an additional node at the same level. These additional expensive calculations impede the progress of processes depending on these results. We discuss in the next section a method for parallelizing the redundant M2M and L2L operations to largely eliminate this bottleneck.

Finally, multiscale geometries often result in an imbalanced workload distribution. The initial distribution aims only to apportion approximately the same number of particles on each process, not accounting for the density of the distribution or the actual work to be done; as a result, processes that own regions of dense discretization have fewer nodes, and thus less work to be done, than

other processes. To alleviate this problem, an algorithm to approximately balance the workload is proposed in a later section.

4.4 Overcoming bottlenecks associated with plural nodes

4.4.1 Scheduling computations and communications for plural nodes

In order to make use of efficient MPI collective communication operations for plural nodes, it is necessary to create local communicators. It can be shown that at any level, the local subtree of any process has at most one duplicate node and at most one shared node. Therefore, on any process only two communicators are required per level. The use of blocking MPI calls used for communication in the algorithm described in the following section, however, requires a consistent execution schedule that is free of deadlocks. It suffices to determine such a schedule for each level independently.

An algorithm for scheduling a single level is presented in Algorithm 4, with an illustrative example in Table 4.1. We begin by assigning shared nodes into communication slot 1 and duplicate nodes into communication slot 2. The communication slot of a node b is given by $A(b)$. On process i, the shared node is denoted s_i and the duplicate node d_i. From the initial configuration, we proceed backward from the end of the process list and for each process in sequence, impose the communication slot of the current shared node upon its users, swapping slots where necessary. By the definition of resident process, each such imposition only affects processes of lower rank than the present process, so at step i the schedule is finalized for all processes of rank $j \geq i$. Once the schedule is determined for each level, one MPI communicator is created for each communication slot using `MPI_Comm_split` for a total of $2L$ communicators, well beneath the limits imposed by standard MPI implementations.

4.4.2 Parallelization of Fourier-based interpolation/anterpolation operators

While the adaptive direction partitioning scheme helps to balance the load within the V, W, X-list evaluations, interpolation and anterpolation operations in the M2M and L2L stages for plural

Algorithm 4: `SchedulePluralNodes`: Scheduling algorithm for communication and computation involving plural nodes

for $i \leftarrow N_p - 1, \ldots, 0$ **do**
  if $i ==$ rank **then**
    for $k \in U(s_i)$ **do**
      Send($A(s_i)$,k)
    end for
  else if rank $\in U(s_i)$ **then**
    Recv(a,i)
    $A(d_{\text{rank}}) \leftarrow a$
    $A(s_{\text{rank}}) \leftarrow (a \bmod 2) + 1$
  end if
end for

Table 4.1: Tabular illustration of Algorithm 4. Each row from top to bottom represents a step. Tildes denote the shared nodes on their resident processors. Boldface denotes that a node has been scheduled.

P_0	P_1	P_2	P_3	P_4	P_5	P_6
$(\cdot, a)$	$(\tilde{a}, b)$	$(\cdot, b)$	$(\tilde{b}, c)$	$(\tilde{c}, d)$	$(\cdot, d)$	$(\tilde{d}, \cdot)$
$(\cdot, a)$	$(\tilde{a}, b)$	$(\cdot, b)$	$(\tilde{b}, c)$	$(\mathbf{d}, \tilde{c})$	$(\mathbf{d}, \cdot)$	$(\tilde{\mathbf{d}}, \cdot)$
$(\cdot, a)$	$(\tilde{a}, b)$	$(\cdot, b)$	$(\tilde{b}, \mathbf{c})$	$(\mathbf{d}, \tilde{\mathbf{c}})$	$(\mathbf{d}, \cdot)$	$(\tilde{\mathbf{d}}, \cdot)$
$(\cdot, a)$	$(\mathbf{b}, \tilde{a})$	$(\mathbf{b}, \cdot)$	$(\tilde{\mathbf{b}}, \mathbf{c})$	$(\mathbf{d}, \tilde{\mathbf{c}})$	$(\mathbf{d}, \cdot)$	$(\tilde{\mathbf{d}}, \cdot)$
$(\cdot, \mathbf{a})$	$(\mathbf{b}, \tilde{\mathbf{a}})$	$(\mathbf{b}, \cdot)$	$(\tilde{\mathbf{b}}, \mathbf{c})$	$(\mathbf{d}, \tilde{\mathbf{c}})$	$(\mathbf{d}, \cdot)$	$(\tilde{\mathbf{d}}, \cdot)$

nodes present perhaps an even greater computational bottleneck at scale. For a fixed problem size, the number of plural nodes is proportional to the number of processes used in the computation. Attendant to this increase is an increase in duplicated work, particularly at the uppermost levels of the tree where i) the amount of work per node is the highest and ii) nodes are duplicated on the greatest number of processes. Consider the following. If a node b at level ℓ is a plural node which exists on n_b processes, then its parent $P(b)$ and all its ancestors up to the root must also be plural nodes existing on $n \geq n_b$ processes. It follows, then, that the serial operation $P(b) \leftarrow P(b) + \mathcal{SI}\, b$ is performed independently on each process its own part b_i, where $b = b_1 + \ldots + b_{n_b}$. As a consequence, the runtime for M2M and L2L is bounded from below by the time required to perform a serial interpolation or anterpolation operation between the two coarsest levels of the tree, independent of the number of processes used.

Obviously, this represents an enormous bottleneck to scalability. To break it, we parallelize

plural-to-plural Fourier interpolation and anterpolation operations over the processes which possess a copy of the child. In large-scale computations, the size of the nodes in the high-frequency regime dominate the rest, so we allow duplicate work to be done in the mid-frequency regime as it does not make an appreciable difference in comparison. During the upward pass, plural-to-plural M2M operations are deferred until after the rest of the M2M operations are complete. The parallel M2M operations are then performed level by level, beginning with the finest level and ascending the tree, according to the schedule outlined in the previous subsection. At the outset of the L2L stage, the parallel plural-to-plural L2L operations are carried out according to the same schedule from top to bottom, after which the serial L2L work is executed.

Multipole and local expansions are arranged in the form of a 2-D array with the most rapidly varying (row) index representing samples in ϕ and the other representing samples in θ. In the M2M stage, partial multipole expansions of plural nodes are evenly distributed across processes at the granularity of columns via a reduce-scatter operation. The parallel interpolation operation illustrated in Fig. 4.2 proceeds as follows: first, Fourier interpolation of the column (ϕ-varying) data is performed locally on each process via FFT, zero-padding, and inverse FFT (4.2b); this data is then folded according to the spherical symmetry condition described in [60] and transposed across the processes for contiguous access in the following step (4.2c); next, Fourier interpolation is performed on each column (θ-varying) of the folded and transposed data (4.2d). Finally, the data is unfolded back to the original unit sphere, shifted to the parent box, and distributed according to the adaptive direction partition scheme. The L2L stage proceeds in largely the same fashion, with the exception that the initial data distribution is achieved using a broadcast operation so that L2L operations from plural nodes to interior nodes can be performed serially.

4.5 Load balancing

4.5.1 MLFMA vs. static FMM

Load balancing for the MLFMA is significantly more complicated than for static FMMs. Broadly speaking, load balancing the FMM for the Laplace kernel is simple in that work is concentrated

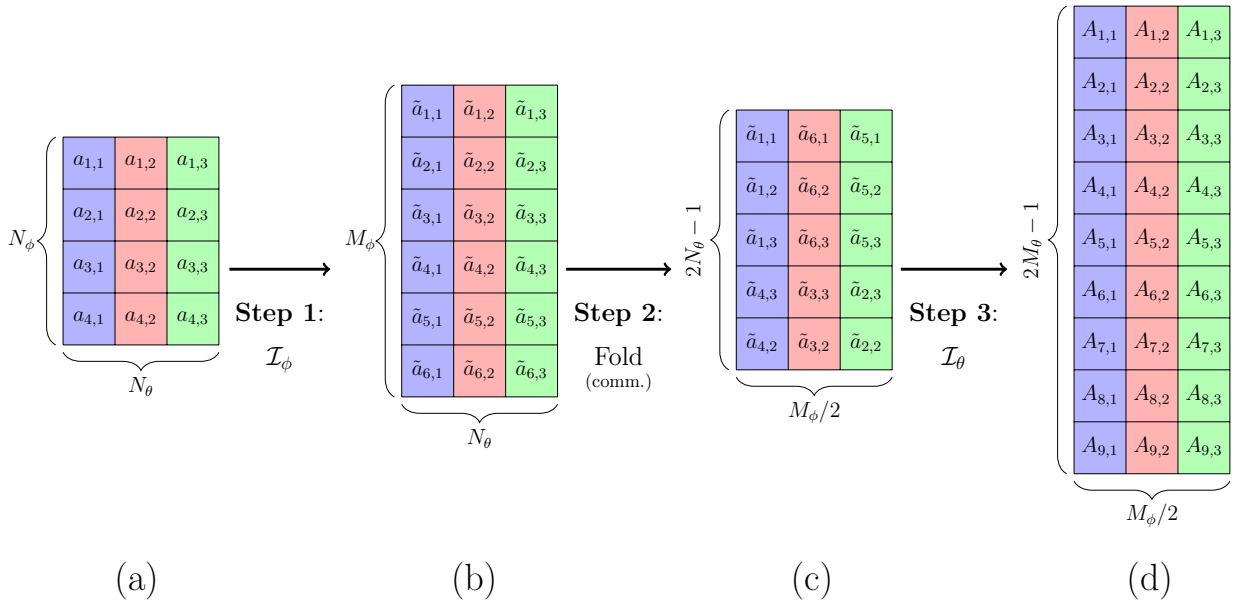


Figure 4.2: Graphical illustration of the transposition and folding operation within the parallel FFT interpolation of radiation pattern a to A for $N_\theta = 3, N_\phi = 4, M_\theta = 5, M_\phi = 6$.

at the leaves as work per level falls off rapidly above the leaf level. Sufficient load balance can be achieved by considering the work model for the leaves only. Obviously, the problem is complicated by non-uniform distributions of particles, but the fact remains that the work per node is constant no matter the level.

In the adaptive wideband MLFMA, the work associated with each node depends on the level and regime. In the low-frequency regime, the work per node is constant irrespective of level, as in the static FMM. Here, V-list and M2M/L2L operations dominate the cost. In the mid- and high-frequency regimes, the work for a node at level l is proportional to $(K(l))^3$ or $(K(l))^2 \log K(l)$, respectively. In these regimes, M2M and L2L dominate the complexity estimate. Consequently, and especially for surface distributions, the work per level is *constant* and must therefore be accounted for in the work profile used for load balancing.

4.5.2 Parallel load balancing strategy for adaptive wideband MLFMA

Whether or not the adaptive tree is employed, the basic strategy of equipartitioning the particles over processes fails to give good load balance for spatially non-uniform distributions. A more

effective re-partitioning of the tree to balance the computational load is then in order. If a new, more optimal set of splitters is determined, the original partitioning algorithm can re-partition the tree using these splitters.

The splitters are determined as follows. A cost model is established based on timings of dummy M2M, L2L, and U, V, W, X-list operations. For each node in the tree, each of these operations is counted, multiplied by the appropriate cost, and summed to obtain an intrinsic node cost. Next, the costs for plural nodes are summed in parallel to obtain a consistent account across resident and user processes. Then, for each node b, the number of leaf nodes contained within the subtree rooted at b is determined and synchronized in the same fashion as the particle count in the merging algorithm described above. This is denoted by $N_l(b)$.

Now, the cost per leaf is determined. First, define the *partial percolated cost* (PPC) of a node as a bucket for the cost of its ancestors scaled by the number of leaves beneath them, and initialize this to zero. Starting at the root node and progressing downward level by level, for each node b, the intrinsic cost for b is divided by $N_l(b)$ and added to the PPC of each of its children, along with b's PPC. This process goes on until the leaf level, where the cost for each is computed by the sum its intrinsic and partial percolated costs. This procedure is perhaps best explained via illustration, as in the exemplary Fig. 4.3. The purpose of the PPC scheme is to evenly distribute the cost of upper-level nodes to children in a manner that i) is not influenced by the initial partitioning, and ii) does not overwhelm the costs of lower-level nodes.

Lastly, a parallel prefix sum of the leaf costs in the Morton sequence is computed, and the total leaf-level cost C is broadcast from the highest-rank process. Denoting the number of processes used by N_p, splitters are determined to be the $N_p - 1$ leaf keys closest to each of mC/N_p, $m = 1, \ldots, N_p - 1$. All splitter keys are then shifted back to their leftmost descendants at level L for use in re-partitioning. The greatest limitation of the proposed algorithm in terms of load balancing ability is that the existence of plural nodes renders the true load balancing problem non-linear, as the associated costs are a function of the splitter locations. Ideally, this problem could be addressed via an iterative balancing and cost re-evaluation procedure, but in this context doing so would be

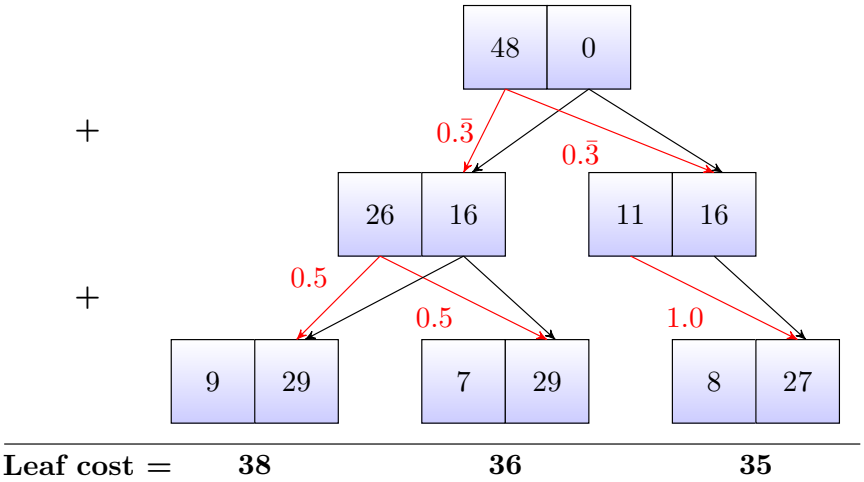


Figure 4.3: Illustration of the cost-percolation algorithm. Each pair of boxes represents a node in the tree; the number in the left box is the intrinsic cost, while the number in the right box is the partial percolated cost of all ancestors.

prohibitively expensive. Because the interaction list communication graph depends on the tree partitioning, it would need to be re-computed after each iteration to calculate the load.

4.6 Results

In this section, we present the results of numerical experiments demonstrating the gains in performance resulting from the parallelization techniques described in the previous sections. All results presented here were obtained on a cluster comprising nodes with two sockets each, populated by Intel Xeon E5-2698 v3 ("Haswell") processors with a clock speed of 2.3 GHz. Each node has 32 cores and 128 GB 2133MHz DDR4 RAM. The code is implemented in Fortran 90 using only MPI parallelization and compiled with Intel compilers. The FFTW library is used for all FFTs.

We will first examine the performance gains in the M2M and L2L stages from parallelizing the Fourier interpolation/anterpolation operators for plural nodes. We first consider a sphere of diameter 320λ discretized using 241,920,000 dipoles on the surface. The leaf box is chosen as $d_0 = 0.2\lambda$, yielding a 12-level tree. The transition level is chosen such that the finest five levels of computation use spherical sampling and the top five levels use uniform.

Timings for M2M and L2L for different numbers of processes N_p up to 2048 are given in Table 4.2. As with the grid geometry, performance gains using PFFT become increasingly evident

with increasing N_p, culminating in speedups of over 3× on 2048 processes. The scalability of the traversal operations thus improves dramatically. A more intimate perspective is offered by Figs. 4.4-4.5, which present per-process timings for M2M and L2L in both the original and PFFT-augmented algorithms. Unlike the grid geometry, a sphere does not conform directly to the octree structure so imbalances in these operations are to be expected. This is evident from the timings shown in Fig. 4.5a), which shows timings of individual components of M2M and L2L for the original algorithm. Here, a large amount of time is spent waiting in M2M at an `MPI_Waitall` as other processes finish their portion of the serial interpolations. L2L also suffers from a load imbalance resulting from uneven distribution of high-level serial anterpolations. These effects are highlighted by the extreme imbalance of composite serial interpolation and anterpolation (int/ant) timings. Parallelization of these operations, as shown in Fig. 4.5b), significantly reduces both the computation and wait times. The serial interpolation and anterpolation are reduced to about one second overall in the worst case, and the overall distribution is much more balanced. The difference in the shapes of the M2M and L2L timings is due to the fact that M2M has a synchronization barrier at the end, while L2L does not. We next consider a $512\lambda \times 512\lambda$ grid of particles in the $z = 0$ plane with a grid spacing of $\lambda/32$ with 268,435,456 points. The box size is chosen to be 0.25λ, resulting in a 12-level tree with 10 levels of computation. For the hybrid implementation, we empirically set the bottom 6 levels of the tree to employ the spherical scheme, while the top 4 levels of computation employ the uniform scheme. The oversampling parameter χ is chosen to be 1.

The regularity of the grid geometry helps to highlight load imbalances inherent to the original algorithm and to illustrate the benefits of the PFFT algorithm. A study of the timings for M2M and L2L for both the original and PFFT schemes is shown in Table 4.3. We observe bigger gains in performance relative to the original scheme as the number of processes N_p increases. This is due to the increasing number of plural nodes to be parallelized. PFFT also facilitates significant improvements in the M2M and L2L parallel efficiency (*strong scaling*), defined as

$$\text{Eff.}\left[N_q\right](\%) \doteq \frac{N_p T_{N_p}}{N_q T_{N_q}} \times 100\%, \tag{4.3}$$

Table 4.2: Comparison of original algorithm with PFFT for 320λ diameter sphere geometry.

	Orig (s)		**PFFT** (s)		**Speedups**		**Orig Eff. (%)**		**PFFT Eff. (%)**	
N_p	M2M	L2L	M2M	L2L	M2M	L2L	M2M	L2L	M2M	L2L
32	78.6	62.6	73.9	71.9	1.06	0.87	100	100	100	100
64	43.7	34.4	41.3	38.5	1.06	0.89	89.9	91.0	89.5	93.4
128	28.9	22.0	26.2	20.0	1.10	1.10	68.0	71.1	70.5	89.9
256	16.7	12.4	13.9	10.5	1.20	1.18	58.8	63.1	66.5	85.6
512	12.5	9.16	9.08	6.87	1.38	1.33	39.4	42.7	50.9	65.4
1024	18.6	14.7	6.38	4.78	2.92	3.08	13.2	13.3	36.2	47.0
2048	12.7	9.22	3.80	2.97	3.34	3.10	9.67	10.6	30.4	37.8

where N_p, T_{N_p} are the reference process count and computation time, and N_q, T_{N_q} are the process count and computation time for which the efficiency is being calculated. We now consider the case of a $1024\lambda \times 1024\lambda$ grid lying in the $z = 0$ plane with a regular spacing between points of $\lambda/32$ in each dimension. The total number of particles in this geometry is 1,073,741,824. The leaf box size is chosen as $\lambda/4$, resulting in a 13-level tree (with 11 levels of computation). The finest six levels of the tree employ spherical interpolation/anterpolation and the coarsest five levels employ uniform interpolation/anterpolation.

Figure 4.6 provides a comparison of tree traversal timings for 2048 processes both with and without the PFFT scheme. Specific timings for the run without the PFFT scheme are shown in Figure 4.7a), while those for the run with the PFFT scheme are given in Figure 4.7b). As with the previous grid example, the apparently well-balanced load in M2M for the former case is shown to be misleading, as most processes spend approximately 23 seconds in M2M waiting for nodes with larger than average amounts of serial interpolations to finish their serial computations. The load imbalance is also clear in the timings for L2L, which lacks an explicit synchronization step at the end.

The PFFT scheme, in contrast, results in much better load balance in both M2M and L2L stages, along with reduced run-times due to the parallelization of some previously serial computations. The time spent in M2M is reduced from 46.2 seconds for the original algorithm to 7.65 seconds with the PFFT scheme, a speedup of over 6×. Similarly, the maximum time spent in L2L decreases from 35.0 seconds to 5.44 seconds for a speedup of 4.6×.

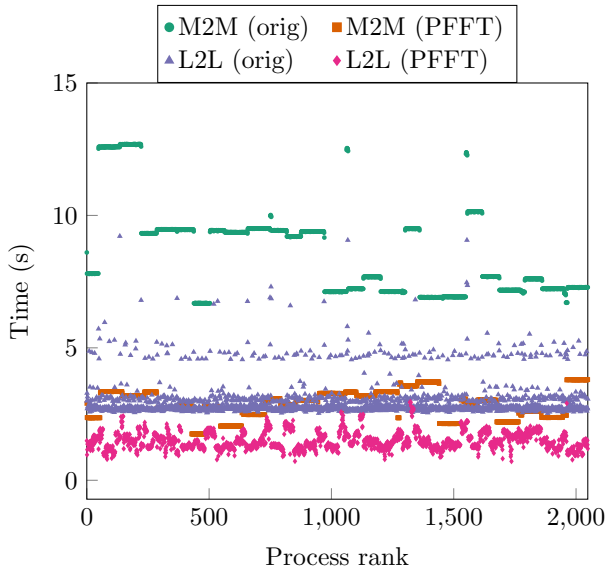
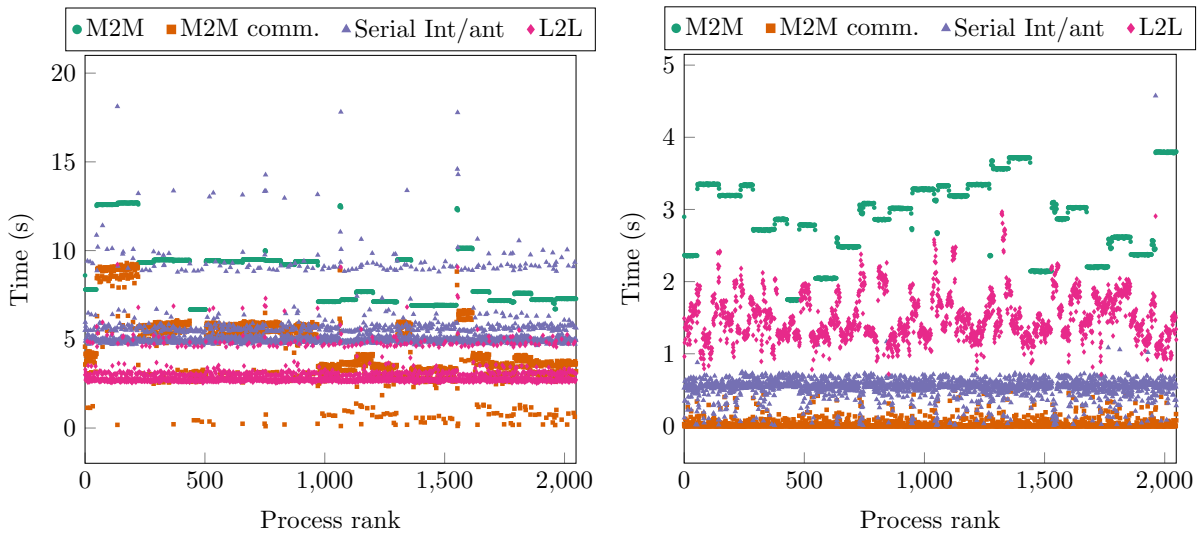


Figure 4.4: Timing comparison of M2M and L2L operations for 320λ diameter sphere on 2048 cores with and without PFFT.



(a) Without PFFT

(b) With PFFT

Figure 4.5: Timing breakdown for M2M/L2L with and without PFFT for the 320λ sphere.

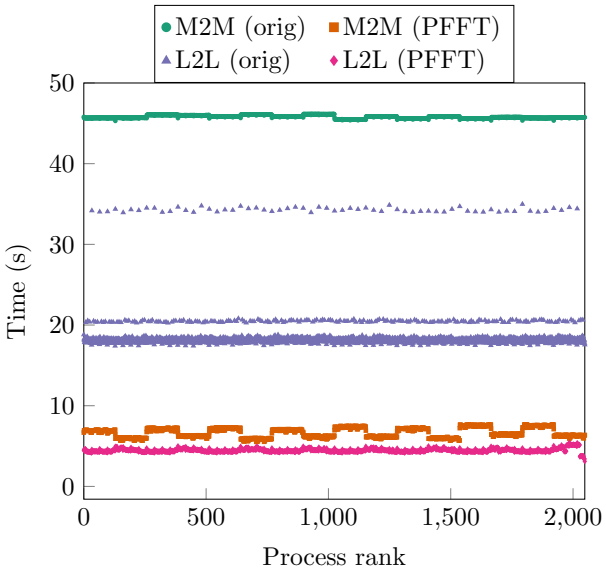
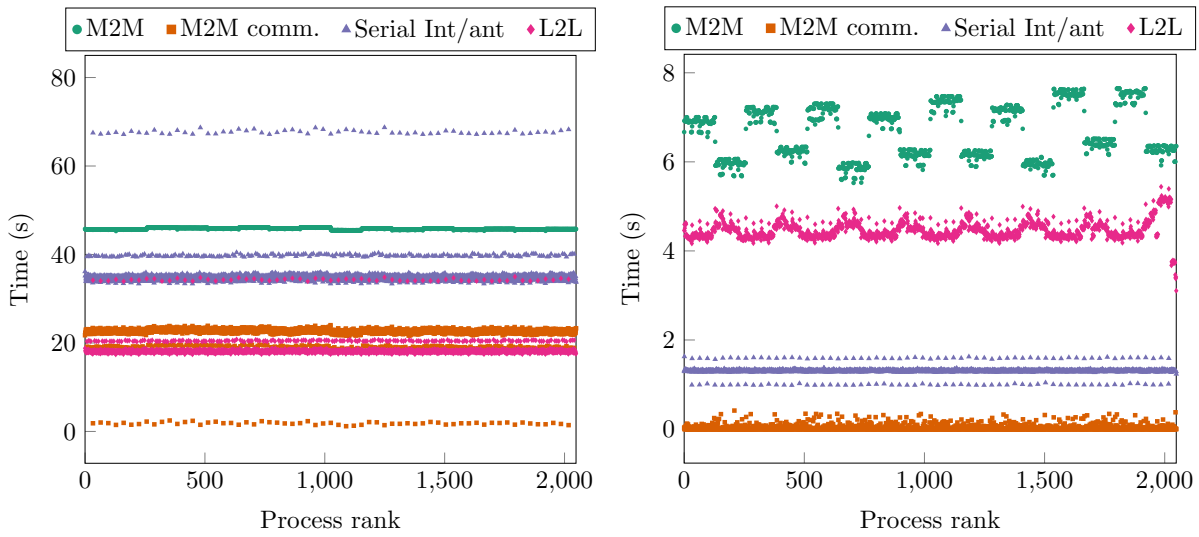


Figure 4.6: Timing comparison of M2M and L2L operations for 1024λ diameter grid on 2048 cores with and without PFFT.



(a) Without PFFT

(b) With PFFT

Figure 4.7: Timing breakdown for M2M/L2L with and without PFFT for the 1024λ grid.

Table 4.3: Comparison of PFFT with original algorithm for 512λ grid geometry.

	Orig (s)		**PFFT** (s)		**Speedups**		**Orig Eff.** (%)		**PFFT Eff.** (%)	
N_p	M2M	L2L	M2M	L2L	M2M	L2L	M2M	L2L	M2M	L2L
32	45.4	38.0	44.5	37.4	1.02	1.02	100	100	100	100
64	25.8	21.2	24.9	20.3	1.04	1.04	88.0	89.6	89.4	92.1
128	18.5	14.3	12.9	10.5	1.43	1.36	61.4	66.4	86.2	89.0
256	13.5	9.9	7.4	5.7	1.82	1.74	42.0	48.0	75.2	82.0
512	12.1	8.5	4.4	3.4	2.75	2.5	23.5	27.9	63.2	68.8
1024	11.4	7.7	3.0	2.3	3.80	3.35	12.4	15.4	46.4	50.8

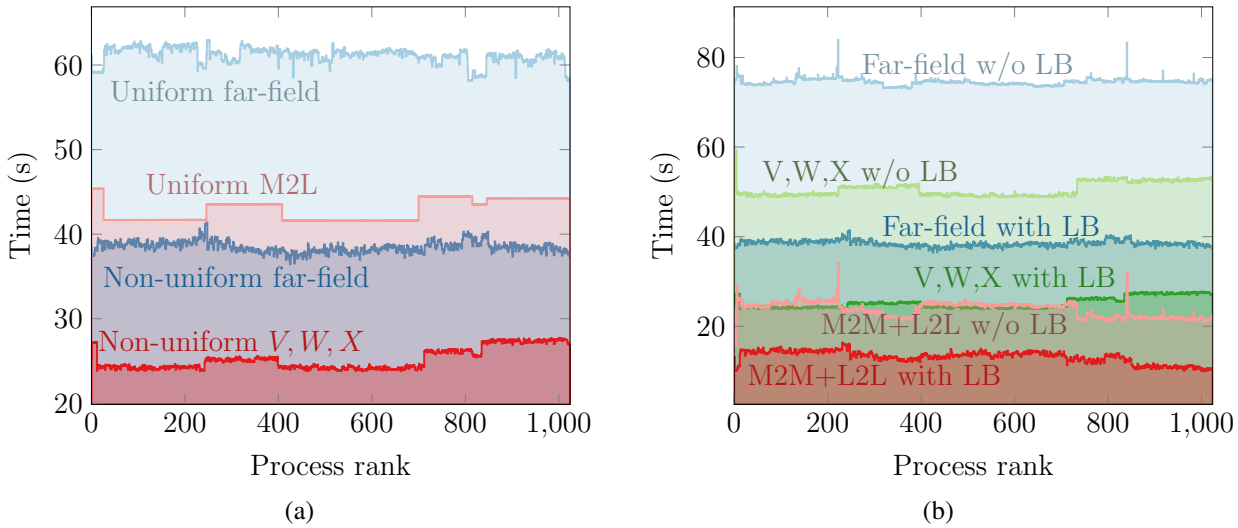


Figure 4.8: Per-process timings for far-field matvec stages for the airplane geometry (a) with uniform vs. non-uniform tree and (b) with and without load balancing (LB) for non-uniform tree on 1024 processes.

Finally, we examine the effects of merging the tree for non-uniform distributions and load balancing. We consider a geometry containing 175,764,666 points on the surface of an aircraft which fits into a bounding box of dimensions $693.5\lambda \times 200.2\lambda \times 754.8\lambda$. The minimum leaf box diameter was set to $\lambda/32$ resulting in a 16-level tree, and the densest box contains 56 points. The four coarsest levels of computation use uniform sampling, and the remaining seven MLFMA levels use spherical sampling. ACE is used for boxes of diameter smaller than $\lambda/4$. The non-uniform tree is merged with $s_{max} = 40$, resulting in leaves containing 20 points on average distributed over the bottom five levels of the tree, and an almost twenty-fold reduction in leaf boxes. For accuracy on the order of 10^{-3}, we set $\chi = 1.0, P = 3$ [73]. This supposition is backed up by randomly selecting

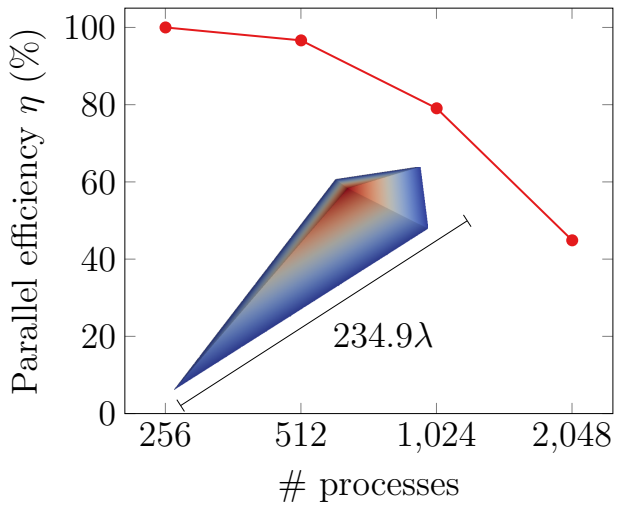


Figure 4.9: Parallel efficiency of the complete *matvec* for the 286M-point arrow geometry with reference to 256 processes. Colorization of geometry solely to illustrate depth.

a point on each process to ensure good spatial distribution, computing the exact observed field, and comparing with the computed value; the average relative error at these observers is 3.98×10^{-3}.

With this setup, we examine the trade-offs for non-uniform trees vs. uniform trees and the beneficial effects of load balancing for parallel matvecs. Fig. 4.8a shows the time taken per process for the far-field matvec using 1024 processes using a uniform tree and a non-uniform tree. The time drops from 61 seconds on average for the uniform tree to 38 seconds for the non-uniform tree. Load balancing is employed in both instances. Much of the speedup is achieved in the translation stages. The flatness of portions of the M2L timings for the uniform tree are caused by synchronization barriers. Fig. 4.8b details the effectiveness of load balancing, showing per-process timings for the tree traversal (M2M+L2L) and translation ($V, W, X-$lists) both with and without load balancing with exactly the same parameters. Without load balancing, the translation stage alone takes ten seconds longer than the entire matvec with load balancing.

Next, we consider a collection of 286,312,650 points distributed uniformly on a surface geometry with an arrow-like shape, yielding high variation in point density over the surface. The bounding box for the arrow measures $234.9\lambda \times 77\lambda \times 39.32\lambda$, and we choose the minimum box size to be $\lambda/64$, resulting in a 15-level tree. As always, ACE is used for boxes smaller than $\lambda/4$. The three coarsest MLFMA levels of computation employ uniform sampling, while the rest employ spherical

sampling. The tree is merged using $s_{max} = 25$, resulting in 9 points per leaf box on average.

The parallel efficiency of the complete *matvec* (both near- and far-field together) is given in Fig. 4.9 with reference to 256 processes. Memory limitations prevented us from running the code under the same memory-per-core conditions beneath 256 processes. Up to 1,024 processes the *matvec* exhibits good scaling, but drops precipitously for 2,048 processes. This appears to be associated with synchronization steps within the parallel interpolation/anterpolations.

4.7 Conclusion

In this work, we have proposed and demonstrated several remedies to fundamental bottlenecks in the parallel wideband MLFMA of [48] updated with the adaptive version of the same algorithm presented in [39]. We provided a parallel algorithmic prescription for constructing non-uniform trees and interaction lists and for load balancing these trees. Our methods enable significant improvements in runtime and scalability for problems up to 1024 wavelengths in diameter.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

The works presented in this thesis demonstrably advance the state of the art in parallelization of the ACE and MLFMA. These developments facilitate efficient simulations involving highly non-uniform distributions of particles.

A novel parallel non-uniform ACE algorithm for evaluating linear operators acting on arbitrary non-oscillatory potentials was shown to be scalable up to 16,384 processes with little loss in performance for even highly non-uniform surface distributions of billions of particles with up to 25 levels in the octree. The error convergence of this method was demonstrated for several different kernel functions. While good results were obtained, further investigation of reducing the cost of translation (M2M, M2L, L2L) from $O(P^6)$ should be pursued. These operators each possess a quasi-convolutional structure which may potentially be exploited to reduce the complexity to something like $O(P^3 \log P)$, as can be done for actual convolutions using fast Fourier transforms. The solution to this problem could significantly boost the appeal of this algorithm.

The bulk of this thesis deals with the non-uniform wideband MLFMA and its parallelization. The contributions presented accomplish several important goals: i) a key computational and memory bottleneck is broken, greatly increasing the maximum problem size; ii) a controllably-accurate, provably convergent, adaptive form of the algorithm is introduced, improving the runtime for multiscale distributions; iii) an efficient parallel algorithm from tree construction to potential evaluation is detailed, breaking two other important computational bottlenecks; and iv) a novel load balancing mechanism for multiscale distributions is presented. There are, however, still many open topics for research. First, investigating flexible direction partitioning schemes (i.e. not based on powers of 2 or 4 as in [28, 50]) and bandlimited local interpolation methods could facilitate analysis of extremely electrically large geometries. To that same end, development of error-controllable "windowed" translation operators in the high-frequency regime may significantly reduce both computation and communication for very large problems. Finally, extending the PEC algorithm

to formulations for penetrable objects such as JMCFIE [55] or PMCHWT [81] will require the development of a novel load balancing mechanism, as variations in subdomain size and material properties lead to severe inefficiencies when each subdomain is processed one at a time.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Emmanuel Agullo et al. "Task-based FMM for multicore architectures". In: *SIAM Journal on Scientific Computing* 36.1 (2014), pp. C66–C93.

[2] Christopher R Anderson. "An implementation of the fast multipole method without multipoles". In: *SIAM Journal on Scientific and Statistical Computing* 13.4 (1992), pp. 923–947.

[3] David L Andrews. "On the conveyance of angular momentum in electronic energy transfer". In: *Physical Chemistry Chemical Physics* 12.27 (2010), pp. 7409–7417.

[4] Andrew W Appel. "An efficient program for many-body simulation". In: *SIAM Journal on Scientific and Statistical Computing* 6.1 (1985), pp. 85–103.

[5] Jonatan Aronsson, Ian Jeffrey, and Vladimir Okhmatovski. "Generalization of the Barnes-Hut algorithm for the Helmholtz equation in three dimensions". In: *IEEE Antennas and Wireless Propagation Letters* 8 (2009), pp. 425–428.

[6] AD Baczewski and B Shanker. "An O (n) method for the rapid analysis of periodic problems using accelerated cartesian expansions (ace)". In: *Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE*. IEEE. 2010, pp. 1–4.

[7] Andrew D Baczewski, Daniel L Dault, and Balasubramaniam Shanker. "Accelerated Cartesian expansions for the rapid solution of periodic multiscale problems". In: *IEEE Transactions on Antennas and Propagation* 60.9 (2012), pp. 4281–4290.

[8] Josh Barnes and Piet Hut. "A hierarchical O (N log N) force-calculation algorithm". In: *nature* 324.6096 (1986), pp. 446–449.

[9] Austin R Benson et al. "A parallel directional fast multipole method". In: *SIAM Journal on Scientific Computing* 36.4 (2014), pp. C335–C352.

[10] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[11] John A Board et al. "Accelerated molecular dynamics simulation with the parallel fast multipole algorithm". In: *Chemical Physics Letters* 198.1-2 (1992), pp. 89–94.

[12] Ignace Bogaert et al. "NSPWMLFMA: A low frequency stable formulation of the MLFMA in three dimensions". In: *Antennas and Propagation Society International Symposium, 2008. AP-S 2008. IEEE*. IEEE. 2008, pp. 1–4.

[13] Alexander H Boschitsch, Marcia O Fenley, and Wilma K Olson. “A fast adaptive multipole algorithm for calculating screened Coulomb (Yukawa) interactions”. In: *Journal of Computational Physics* 151.1 (1999), pp. 212–241.

[14] Martin D Buhmann. *Radial basis functions: theory and implementations*. Vol. 12. Cambridge university press, 2003.

[15] J Carrier, Leslie Greengard, and Vladimir Rokhlin. “A fast adaptive multipole algorithm for particle simulations”. In: *SIAM journal on scientific and statistical computing* 9.4 (1988), pp. 669–686.

[16] Matt Challacombe, Chris White, and Martin Head-Gordon. “Periodic boundary conditions and the fast multipole method”. In: *The Journal of chemical physics* 107.23 (1997), pp. 10131–10140.

[17] YH Chen, WC Chew, and S Zeroug. “Fast multipole method as an efficient solver for 2D elastic wave surface integral equations”. In: *Computational mechanics* 20.6 (1997), pp. 495–506.

[18] Hongwei Cheng, Leslie Greengard, and Vladimir Rokhlin. “A fast adaptive multipole algorithm in three dimensions”. In: *Journal of computational physics* 155.2 (1999), pp. 468–498.

[19] Hongwei Cheng et al. “A wideband fast multipole method for the Helmholtz equation in three dimensions”. In: *Journal of Computational Physics* 216.1 (2006), pp. 300–325.

[20] Weng Cho Chew et al. *Fast and efficient algorithms in computational electromagnetics*. Artech House, Inc., 2001.

[21] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. “The fast multipole method for the wave equation: A pedestrian prescription”. In: *IEEE Antennas and Propagation Magazine* 35.3 (1993), pp. 7–12.

[22] Eric Darve. “The fast multipole method I: error analysis and asymptotic complexity”. In: *SIAM Journal on Numerical Analysis* 38.1 (2000), pp. 98–128.

[23] Daniel Dault and B Shanker. “A mixed potential MLFMA for higher order moment methods with application to the generalized method of moments”. In: *IEEE Transactions on Antennas and Propagation* 64.2 (2016), pp. 650–662.

[24] Walter Dehnen. “A hierarchical O (N) force calculation algorithm”. In: *Journal of Computational Physics* 179.1 (2002), pp. 27–42.

[25] Jack Dongarra and Francis Sullivan. “Guest editors’ introduction: The top 10 algorithms”. In: *Computing in Science & Engineering* 2.1 (2000), pp. 22–23.

[26] Tommi Dufva and Jukka Sarvas. “Broadband MLFMA with plane wave expansions and optimal memory demand”. In: *IEEE Transactions on Antennas and Propagation* 57.3 (2009), pp. 742–753.

[27] Michael A Epton and Benjamin Dembart. “Multipole translation theory for the three-dimensional Laplace and Helmholtz equations”. In: *SIAM Journal on Scientific Computing* 16.4 (1995), pp. 865–897.

[28] ÖzgÜr Ergul and Levent Gurel. “Efficient parallelization of the multilevel fast multipole algorithm for the solution of large-scale scattering problems”. In: *IEEE Transactions on Antennas and Propagation* 56.8 (2008), pp. 2335–2345.

[29] X-J Fan, Nhan Phan-Thien, and Rong Zheng. “Completed double layer boundary element method for periodic suspensions”. In: *Zeitschrift für angewandte Mathematik und Physik ZAMP* 49.2 (1998), pp. 167–193.

[30] William Fong and Eric Darve. “The black-box fast multipole method”. In: *Journal of Computational Physics* 228.23 (2009), pp. 8712–8725.

[31] Yuhong Fu and Gregory J Rodin. “Fast solution method for three-dimensional Stokesian many-particle problems”. In: *International Journal for Numerical Methods in Biomedical Engineering* 16.2 (2000), pp. 145–149.

[32] Leslie Greengard. *The rapid evaluation of potential fields in particle systems*. MIT press, 1988.

[33] Leslie Greengard and William D Gropp. “A parallel version of the fast multipole method”. In: *Computers & Mathematics with Applications* 20.7 (1990), pp. 63–71.

[34] Leslie Greengard and Vladimir Rokhlin. “A fast algorithm for particle simulations”. In: *Journal of computational physics* 73.2 (1987), pp. 325–348.

[35] Leslie Greengard and Vladimir Rokhlin. “A new version of the fast multipole method for the Laplace equation in three dimensions”. In: *Acta numerica* 6 (1997), pp. 229–269.

[36] Leslie Greengard and John Strain. “The fast Gauss transform”. In: *SIAM Journal on Scientific and Statistical Computing* 12.1 (1991), pp. 79–94.

[37] Leslie Greengard et al. “Accelerating fast multipole methods for the Helmholtz equation at low frequencies”. In: *IEEE Computational Science and Engineering* 5.3 (1998), pp. 32–38.

[38] Leslie F Greengard and Jingfang Huang. “A new version of the fast multipole method for screened Coulomb interactions in three dimensions”. In: *Journal of Computational Physics* 180.2 (2002), pp. 642–658.

[39] Stephen Hughey et al. “Parallel Wideband MLFMA for Analysis of Electrically Large, Non-Uniform, Multiscale Structures”. In: ().

[40] Rüdiger Jakob-Chien and Bradley K Alpert. “A fast spherical filter with uniform resolution”. In: *Journal of Computational Physics* 136.2 (1997), pp. 580–584.

[41] Jian-Ming Jin. *Theory and computation of electromagnetic fields*. John Wiley & Sons, 2011.

[42] Ludvig af Klinteberg, Davoud Saffar Shamshirgar, and Anna-Karin Tornberg. “Fast Ewald summation for free-space Stokes potentials”. In: *Research in the Mathematical Sciences* 4.1 (2017), p. 1.

[43] Robert Krasny and Lei Wang. “Fast evaluation of multiquadric RBF sums by a Cartesian treecode”. In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2341–2355.

[44] Konstantin N Kudin and Gustavo E Scuseria. “A fast multipole algorithm for the efficient treatment of the Coulomb problem in electronic structure calculations of periodic systems with Gaussian orbitals”. In: *Chemical Physics Letters* 289.5 (1998), pp. 611–616.

[45] Ilya Lashuk et al. “A massively parallel adaptive fast multipole method on heterogeneous architectures”. In: *Communications of the ACM* 55.5 (2012), pp. 101–109.

[46] M Lu and E Michielssen. “A local filtering scheme for FMM/PWTD algorithms”. In: *Antennas and Propagation Society International Symposium, 2004. IEEE*. Vol. 4. IEEE. 2004, pp. 4523–4526.

[47] 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), p. 17.

[48] Vikram Melapudi et al. “A scalable parallel wideband MLFMA for efficient electromagnetic simulations on large scale clusters”. In: *IEEE Transactions on Antennas and Propagation* 59.7 (2011), pp. 2565–2577.

[49] Bart Michiels. “Parallel fast multipole methods for the simulation of extremely large electromagnetic scattering problems”. PhD thesis. Ghent University, 2013.

[50] Bart Michiels et al. “Full-wave simulations of electromagnetic scattering problems with billions of unknowns”. In: *IEEE Transactions on Antennas and Propagation* 63.2 (2015), pp. 796–799.

[51] Bart Michiels et al. “Weak scalability analysis of the distributed-memory parallel MLFMA”. In: *IEEE Transactions on Antennas and Propagation* 61.11 (2013), pp. 5567–5574.

[52] Naoshi Nishimura. “Fast multipole accelerated boundary integral equation methods”. In: *Applied Mechanics Reviews* 55.4 (2002), pp. 299–324.

[53] Joris Peeters. “Efficient simulation of 3D electromagnetic scattering problems using boundary integral equations”. PhD thesis. Ghent University, 2010.

[54] Andrew F Peterson et al. *Computational methods for electromagnetics*. IEEE press New York, 1998.

[55] Andrew J Poggio and Edmund K Miller. *Integral equation solutions of three-dimensional scattering problems*. MB Assoc., 1970.

[56] Douglas Potter, Joachim Stadel, and Romain Teyssier. “PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys”. In: *Computational Astrophysics and Cosmology* 4.1 (2017), p. 2.

[57] C Pozrikidis. “Computation of periodic Green’s functions of Stokes flow”. In: *Journal of engineering Mathematics* 30.1-2 (1996), pp. 79–96.

[58] Sadasiva Rao, D Wilton, and Allen Glisson. “Electromagnetic scattering by surfaces of arbitrary shape”. In: *IEEE Transactions on antennas and propagation* 30.3 (1982), pp. 409–418.

[59] Vladimir Rokhlin. “Rapid solution of integral equations of classical potential theory”. In: *Journal of computational physics* 60.2 (1985), pp. 187–207.

[60] Jukka Sarvas. “Performing interpolation and anterpolation entirely by fast Fourier transform in the 3-D multilevel fast multipole algorithm”. In: *SIAM Journal on Numerical Analysis* 41.6 (2003), pp. 2180–2196.

[61] B Shanker and H Huang. “Accelerated Cartesian expansions–A fast method for computing of potentials of the form R- ν for all real ν”. In: *Journal of Computational Physics* 226.1 (2007), pp. 732–753.

[62] Balasubramaniam Shanker et al. “Fast analysis of transient electromagnetic scattering phenomena using the multilevel plane wave time domain algorithm”. In: *IEEE Transactions on Antennas and Propagation* 51.3 (2003), pp. 628–641.

[63] Jaswinder Pal Singh et al. “A parallel adaptive fast multipole method”. In: *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*. ACM. 1993, pp. 54–65.

[64] Jaswinder Pal Singh et al. “Load balancing and data locality in adaptive hierarchical N-body methods: Barnes-Hut, fast multipole, and radiosity”. In: *Journal of Parallel and Distributed Computing* 27.2 (1995), pp. 118–141.

[65] Jiming Song and Weng Cho Chew. “Interpolation of translation matrix in MLFMA”. In: *Microwave and optical technology letters* 30.2 (2001), pp. 109–114.

[66] Jiming Song, Cai-Cheng Lu, and Weng Cho Chew. “Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects”. In: *IEEE Transactions on Antennas and Propagation* 45.10 (1997), pp. 1488–1493.

[67] JM Song and Weng Cho Chew. “Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering”. In: *Microwave and Optical Technology Letters* 10.1 (1995), pp. 14–19.

[68] Hari Sundar, Rahul S Sampath, and George Biros. “Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel”. In: *SIAM Journal on Scientific Computing* 30.5 (2008), pp. 2675–2708.

[69] Manouchehr Takrimi, Özgür Ergül, and Vakur B Ertürk. “A novel broadband multilevel fast multipole algorithm with incomplete-leaf tree structures for multiscale electromagnetic problems”. In: *IEEE Transactions on Antennas and Propagation* 64.6 (2016), pp. 2445–2456.

[70] Sanjay Velamparambil, Weng Cho Chew, and Jiming Song. “10 million unknowns: Is it that big?[Computational electromagnetics]”. In: *IEEE Antennas and Propagation Magazine* 45.2 (2003), pp. 43–58.

[71] Melapudi Vikram and Balasubramaniam Shanker. “An incomplete review of fast multipole methods-from static to wideband-as applied to problems in computational electromagnetics”. In: *APPLIED COMPUTATIONAL ELECTROMAGNETICS SOCIETY JOURNAL* 24.2 (2009), pp. 79–108.

[72] Melapudi Vikram and Balasubramaniam Shanker. “Fast evaluation of time domain fields in sub-wavelength source/observer distributions using accelerated Cartesian expansions (ACE)”. In: *Journal of Computational Physics* 227.2 (2007), pp. 1007–1023.

[73] Melapudi Vikram et al. “A novel wideband FMM for fast integral equation solution of multiscale problems in electromagnetics”. In: *IEEE Transactions on Antennas and Propagation* 57.7 (2009), pp. 2094–2104.

[74] Melapudi Vikram et al. “Accelerated Cartesian expansion (ACE) based framework for the rapid evaluation of diffusion, lossy wave, and Klein–Gordon potentials”. In: *Journal of Computational Physics* 229.24 (2010), pp. 9119–9134.

[75] Melapudi Vikram et al. “Parallel accelerated Cartesian expansions for particle dynamics simulations”. In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE. 2009, pp. 1–11.

[76] Michael S Warren and John K Salmon. “A parallel hashed oct-tree n-body algorithm”. In: *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*. ACM. 1993, pp. 12–21.

[77] Changjiang Yang et al. “Improved fast gauss transform and efficient kernel density estimation”. In: *null*. IEEE. 2003, p. 464.

[78] Samuel YK Yee. “Studies on Fourier series on spheres”. In: *Monthly Weather Review* 108.5 (1980), pp. 676–678.

[79] Lexing Ying, George Biros, and Denis Zorin. “A kernel-independent adaptive fast multipole algorithm in two and three dimensions”. In: *Journal of Computational Physics* 196.2 (2004), pp. 591–626.

[80] Lexing Ying et al. “A new parallel kernel-independent fast multipole method”. In: *Supercomputing, 2003 ACM/IEEE Conference*. IEEE. 2003, pp. 14–14.

[81] Pasi Yla-Oijala and Matti Taskinen. “Application of combined field integral equation for electromagnetic scattering by dielectric and composite objects”. In: *IEEE Transactions on Antennas and Propagation* 53.3 (2005), pp. 1168–1173.

[82] Feng Zhao. “An O (N) algorithm for three-dimensional N-body simulations”. In: (1987).

[83] Jun-Sheng Zhao and Weng Cho Chew. “Three-dimensional multilevel fast multipole algorithm from static to electrodynamic”. In: *Microwave and Optical Technology Letters* 26.1 (2000), pp. 43–48.