INTEGRAL-EQUATION-BASED FAST ALGORITHMS AND GRAPH-THEORETIC METHODS FOR LARGE-SCALE SIMULATIONS

Bo Zhang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics.

Chapel Hill
2010

Approved by:

Professor Jingfang Huang
Professor Michael L. Minion
Professor Laura Miller
Professor Nikos P. Pitsianis
Professor Xiaobai Sun

ABSTRACT

BO ZHANG: Integral-equation-based Fast Algorithms and Graph-theoretic Methods for Large-scale Simulations
(Under the direction of Professor Jingfang Huang)

In this dissertation, we extend Greengard and Rokhlin's seminal work on fast multipole method (FMM) in three aspects. First, we have implemented and released open-source new-version of FMM solvers for the Laplace, Yukawa, and low-frequency Helmholtz equations to further broaden and facilitate the applications of FMM in different scientific fields. Secondly, we propose a graph-theoretic parallelization scheme to map FMM onto modern parallel computer architectures. We have particularly established critical path analysis, exponential node growth condition for concurrency-breadth, and a spatio-temporal graph partition strategy. Thirdly, we introduce a new kernel-independent FMM based on Fourier series expansions and discuss how information can be collected, compressed, and transmitted through the tree structure for a wide class of kernels.

To my parents and my wife.

ACKNOWLEGEMENTS

First and foremost, I would like to express my greatest gratitude and appreciation to my advisor Jingfang Huang, without whom this dissertation would simply be impossible. Huang has generously helped me and supported me in various ways. It is my great honor to have him as my advisor.

I thank Xiaobai Sun and Nikos Pitsianis for broaden my knowledge in the field of computer science, for providing me with unconditional help, support and guidance on many topics of this dissertation in the past one and a half years. The benefit of their suggestions could hardly be exaggerated.

I am very grateful to Michael Minion and Laura Miller, for their intelligent advice and helps during my study at UNC.

I would like to thank the support from my family. To my parents, for raising me, educating me, and loving me. To my wife Jianyu, for believing in me, supporting me, and encouraging me in all these years.

Finally, I would like to thank all my friends (students, faculty, and administrators) for providing me with such a pleasant environment during my study at UNC.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In the last two decades, the scientific community has witnessed the broad influence and impact of the seminal work by Greengard and Rokhlin in 1987 [34] on particle simulations using the fast multipole method (FMM). Due to its arithmetic $O(N)$ or $O(N \log N)$ complexity, in linear or super linear proportional to the number N of interacting particles [34, 35], the FMM has accelerated or enabled many important large-scale calculations or simulations wherever directly applicable, in scientific and engineering studies. In particular, the FMM has inspired the study of integral equation method as a new and effective approach for numerical solutions or simulations of different types of physical, chemical or biochemical processes that are traditionally modeled as boundary value problems of partial differential equations (see, for example, [59, 60, 55, 53, 54]).

The fundamental observation in the FMM is that the numerical rank of the far-field ("well-separated") interactions is relatively low and hence can be "compressed" into p terms (depending on the required accuracy) of the so-called "multipole expansion". For the Coulomb interactions, assuming $\frac{R}{\rho_i} > c_0 > 1$ for "source" i carrying a charge q_i located at $(\rho_i, \alpha_i, \beta_i)$ in the spherical coordinate, the multipole expansion is given by

$$\Phi(R, \theta, \phi) = \sum_{i=1}^{N} q_i \cdot \frac{1}{|R - \rho_i|} \approx \sum_{n=0}^{p} \sum_{m=-n}^{m=n} M_n^m \frac{Y_n^m(\theta, \phi)}{R^{n+1}} \tag{1.1}$$

where the multipole coefficients are computed by

$$M_n^m = 8 \sum_{i=1}^{N} q_i \cdot Y_n^{-m}(\alpha_i, \beta_i). \tag{1.2}$$

The spherical harmonic function of order n and degree m is defined according to the formula in [2].

$$Y_n^m(\theta, \phi) = \sqrt{\frac{(2n+1)(n-|m|)!}{4\pi(n+|m|)!}} \cdot P_n^{|m|}(cos\theta) e^{im\phi}, \tag{1.3}$$

and P_n^m is the associated Legendre polynomial.

For arbitrary distributions of particles, a hierarchical oct-tree (in 3D) is generated so that each particle is associated with boxes at different levels. A divide-and-conquer strategy is applied to account for the far-field interactions at each level in the tree structure, by accumulating information from the multipole expansions in the interaction list to the "local expansions" given by

$$\Phi(R, \theta, \phi) = \sum_{i=1}^{N} q_i \cdot \frac{1}{|R - \rho_i|} \approx \sum_{n=0}^{p} \sum_{m=-n}^{m=n} L_n^m \cdot R^n Y_n^m(\theta, \phi) \tag{1.4}$$

where $\{L_n^m\}$ are the local expansion coefficients. The local expansion of a parent box which collects its far-field contributions can be transmitted to its children. As each particle only interacts with the box and nearby particles at the finest level, and information at higher levels is transferred using a combination of multipole and local expansions, the original FMM has asymptotically optimal complexity $O(N)$. However, because the multipole-to-local translation requires prohibitive $189p^4$ operations for each box, the huge prefactor makes the original FMM less competitive compared with the FFT-based methods. In 1997, a new-version FMM was introduced by Greengard and Rokhlin [35] for the Laplace equation, in which exponential expansions are introduced to diagonalize the multipole-to-local translation, and a "merge-and-shift" technique is used to further reduce the number of such translations. Compared with the original FMM, the complexity

was reduced from $189p^4$ to $40p^2 + 6p^3$ for each box. Numerical experiments show that the new-version FMM breaks even with direct calculation when the number of particles $N = 750$ for three digit accuracy, and $N = 1500$ for six digit accuracy for the Coulomb interactions. The details of the FMM will be discussed in Chpater 2.

The new-version FMM poses multiple challenges on software or hardware implementations, and presents a fundamental question on computer system support of large-scale simulation in terms of software and hardware design and development. Despite the challenges, a lot of efforts have been devoted to developing new-version FMM-based numerical tools, in order to meet the increasing and practical demand in computational sciences and engineering. In the following, we discuss three recent research efforts to demonstrate the power of the new-version FMM as well as the urgent need for further improvements on modern parallel computer architectures.

In [65], Huang et al. have applied an adaptive new-version FMM to solve the steady Stokes equations. This algorithm is the basis for the software package **FMMStokes**. The solver has been applied to the simulation of Stokes flow inside a micro-fluidic device with complex geometry. This device consists of two parallel circular plates, with 81 spouts placed on the top. The bottom plate is moving with axial symmetrical normal velocity. The inlet and spouts are specified by traction boundary conditions; zero-velocity boundary conditions are specified on other surfaces. In the simulation, the device is discretized into $279,046$ triangular elements. In the adaptive FMM, 18 trems in the multipole, local and exponential expansions were used. The CPU time requirement for the problem with more than 10^6 unknowns is about 3 hours, substantially more efficient than other existing solvers. In Figure 1.1, the contour plot of the surface normal traction is shown, which is good agreement with observed experimental results.

The Stokes solve in its current state, nonetheless, calls for further improvements. In particular, the parallelization of the algorithm seems non-trivial. Figure 1.2, taken from [65], shows the degradation in parallel performance, although it compares very favorably

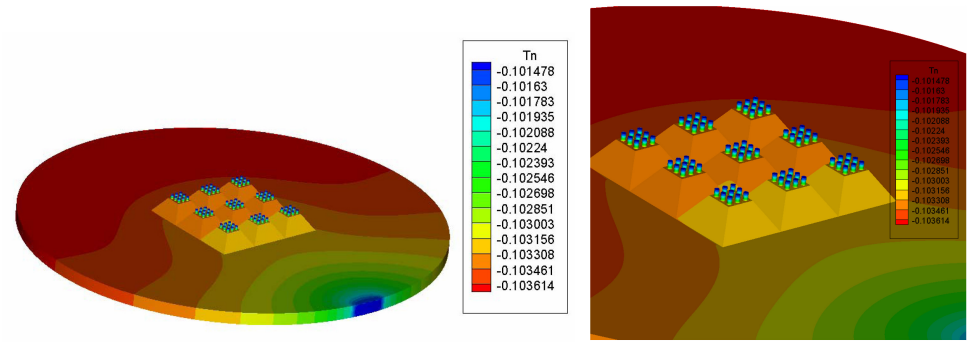


Figure 1.1: Normal stress distribution (left) and its zoomed view (right)

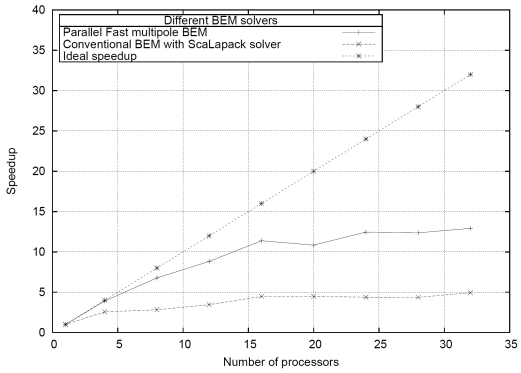


Figure 1.2: Parallel speedup vs. number of processors

ScaLapack based methods. In fact, this is a challenge brought forth by the adaptive FMM.

In [46], an Adaptive Fast Multipole Poisson-Boltzmann (AFMPB) solver was presented for the linearized Poisson-Boltzmann (LPB) equation which models the electrostatic interactions in many biomolecule systems. A second kind integral equation reformulation approach is successfully applied to single-molecule cases as well as to systems with more than one macro-molecule in the solver, and the Krylov-space-based iterative methods such as the generalized minimal residual (GMRES) or bi-conjugate gradients stabilized (BiCGStab) methods are therefore used effectively. Each iteration utilizes the new-version FMM. The arithmetic complexity of the solver is asymptotically optimal $O(N)$ both in CPU time and memory usage. Numerical experiments show that the solver

can compute the electrostatic field of a small-scale molecule system (about $100,000$ unknowns) in less than 10 seconds on today's desktop and reduce the time for very large molecule structure from days by other software tools to 20 minutes with AFMPB. In has also been tested on different biomolecule systems including the nicotinic acetylcholine receptor (nAChR), and interactions between protein Sso7d and DNA. The software has been released under open source license agreements, and there are already interests in incorporating this package into other commonly used tools for molecular dynamics simulations. AFMPB has made a significant and critical leap, in terms of arithmetic complexity and numerical stability, towards the simulations of electrostatics of large-scale systems in protein-protein interactions and nano particle assembly processes. Its potential is yet to be fully explored on modern and emerging multicore or multi-processors computing systems for simulating a very large molecular system such as in protein-protein interacitons or molecular binding, where millions of such evaluations are required. By estimation, we expect that the time for calculating the electrostatic field to be within one second in order to perform the dynamical simulation within a reasonable time frame. In other words, an order or two of magnitude in speed-up is expected.

In [79], a new-version FMM was applied to calculate the stress field of dislocation ensembles due to the long-range interaction of dislocations that are the primary carriers of plastic deformation in crystals. Their numerical results show that the new algorithm is very efficient and accurate, and can evaluate the stress field of a large number of dislocations. However, the presented algorithm is currently designed only for dislocation ensembles in isotropic media, where the interactions can be modeled by the biharmonic kernels. In order to incorporate the elastic anisotropy, new algorithms are required for the anisotropic Green's function commonly used in material science studies.

My research is motivated by the success as well as the urgent need of the new-version FMM for a wider class of problems on modern computer architectures. In particular, this thesis will focus on (1) the development and delivery of high quality open source

new-version FMM codes for the scientific computing community; (2) their parallelization on multicore machines using multi-threading techniques; and (3) generalized kernel independent FMM for a wider class of kernels.

The thesis is organized as follows. In Chapter 2, we first provide a brief overview of the FMM, including its advancements in the last two decades. We then use the screened Coulomb potential as an example to describe the oct-tree data structure in the algorithm and the approximation expansions and corresponding translation operators. Next, we present a pseudo-code for the adaptive algorithm and introduce the first part of my thesis work on developing and delivering an open source package *FMMSuite* of the new-version FMM for the Laplace, Yukawa, and low-frequency Helmholtz kernels. We also present some numerical results to demonstrate the performance of the software. In Chapter 3, we present the second part of my thesis work on the mathematical foundation and numerical implementation of the *FMMSuite* package on multicore computers by means of a graph-theoretic approach. We establish the exponential node growth condition for the concurrency-breadth and critical-path analysis and develop a spatio-temporal graph partition and mapping scheme with respect to the parallel computer architecture configuration and capacity. We have implemented the scheme using multi-threading programming library *Pthreads* and some preliminary results are presented. We are currently developing necessary documentations for the package before releasing it under open source agreements. In Chapter 4, we introduce a new kernel-independent FMM based on the Fourier expansion approximation of the kernel function. In particular, we have established the theoretical results for kernels with scaling property, for instance, the anisotropic kernel in dislocation dynamics. We first present the uniform algorithm, followed by discussions on strategies for further improvements and the introduction of the adaptive version. We present an prototype implementation of the algorithm in 2D in its uniform version in Matlab on which several numerical tests have been carried out. Finally, in Chapter 5, we summarize our existing results, and conclude this thesis with possible applications and

plans for future work.

Chapter 2

Fast Multipole Method

In the numerical simulation of many physical processes, the rapid evaluation of the pairwise interactions of N particles is required. Examples include the dynamics of charged particle clusters governed by the Newton's law of motion in the electrostatic field, and the solution of equations for the gravitational interactions in the study of astrophysics. Given N particles each carrying a charge q_i (or mass m_i) at location $\boldsymbol{x}_i = (x_i, y_i, z_i)$, the electrostatic (or gravitational) potential field is described by

$$\Phi(\boldsymbol{x}_j) = \sum_{\substack{i=1 \\ i \neq j}}^{N} \frac{q_i}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}, \tag{2.1}$$

and the corresponding force field can be computed by means of $\boldsymbol{F} = -\nabla\Phi$. In this formula, the Coulomb interaction $\frac{1}{r}$ is assumed where r is the distance between two particles. Other interactions include the screened Coulomb potential $\frac{e^{-kr}}{r}$ ($k \in \mathcal{R}^+$), the Lennard-Jones potential, and the hydrodynamics interactions described by the Oseen tensor. Equation (2.1) also arises in the integral equation methods when a convolution $\int G(x,y)q(y)\,dy$ of the Green's function (kernel) $G(x,y)$ and a given density $q(y)$ is discretized using quadrature rules. Equation (2.1) can be equivalently represented as a matrix-vector multiplication, with the matrix having zeros on the diagonal and $\{\frac{1}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}\}$ on the off-diagonals and the vector being $\{q_i\}$ for $i, j = 1, \ldots, N$. When a direct method is applied to the summation (or the corresponding matrix-vector multipli-

cation) in Equation (2.1), $O(N^2)$ operations are required, which becomes prohibitive for large-scale problems even on modern supercomputers. Indeed, the advance in computer architectures requires innovative numerical algorithms. In particular, the asymptotically optimal $O(N)$ methods for the special structured matrix vector operations are in urgent need in science and engineering applications.

There have been numerous research efforts to develop $O(N)$ or $O(N \log N)$ algorithms for the summation in Equation (2.1), including the Fast Fourier Transform (FFT) based algorithms (e.g., Particle Mesh (PM) , Particle-Particle Particle-Mesh (P3M) [38], Particle Mesh Ewald (PME) [21], pre-corrected FFT [57]), multi-wavelet based schemes [37], multigrid multi-level methods [15], and the multipole expansion techniques [5, 6, 34, 35]. In this thesis, we focus on the multipole expansion techniques first studied by Appel [5] and Barnes and Hut [6]. In their "tree-code" algorithm, the complexity is reduced to $O(N \log N)$ by using a low order spherical harmonics expansion to represent the "far-field" of a cluster of particles and a downward (or upward) pass to transmit this "multipole" expansion to the "interaction" list. In 1987, by introducing the additional local expansion and using both the upward and downward passes, Greengard and Rokhlin invented the asymptotically optimal $O(N)$ fast multipole method (FMM) and applied it to many-body problems [34]. The FMM was elected as one of the top ten algorithms for the twentieth century and it has been successfully applied in many science and engineering fields such as computational electromagnetic [59, 60, 32, 18, 48, 49, 81, 82], molecular dynamics [14, 31, 45, 47, 40], computational fluid and solide mechanics [33, 26, 63, 69, 73, 72, 74], etc.

However, numerical comparison of the $O(N)$ FMM with the FFT-based algorithms and the tree-code shows that the original version of FMM is less efficient for problem sizes of current interest (e.g., several millions of particles), due to the huge prefactor in the $O(N)$ resulting from the large number of boxes in the interaction list together with the expensive "multipole-to-local" translation operator. To further improve its

efficiency, Greengard and Rokhlin published a new-version FMM [35] for the Coulomb potential after ten years of hard work. In the new-version FMM, an intermediate "plane-wave" representation is introduced to diagonalize the most expensive "multipole-to-local" translation operator, and a "merge-and-shift" technique is applied to reduce the number of such translations. As a result, the new-version FMM can be orders of magnitude faster compared with its original version, especially in 3D. Specifically, the 3D new-version FMM breaks even with direct interaction (estimation of the prefactor in $O(N)$) at approximately $N = 750$ for three digits accuracy and $N = 1500$ for six digits accuracy, compared with tens of thousands for the original version. The details of the original and new version of FMM will be discussed in subsequent sections, and we refer interested readers to [39] for the new version of FMM for the Laplace equation in 2D, to [71] for an improved technique in 1D, to [24] for a new-version FMM accelerated Poisson solver in 2D, and to [20] for the adaptive implementation details for the Laplace equation in 3D.

The new-version FMM has also been generalized to other kernels, including the screened Coulomb potential, the Helmholtz Green's function, and the biharmonic kernel. In particular, in [32], a new-version FMM was developed for the low-frequency Helmholtz equation in 3D by separating the Helmholtz Green's function into the evanescent and propagating parts and introducing the exponential expansions. This technique was later combined with a high-frequency FMM code based on a different diagonal translation operator as described in [60], resulting in the so-called "wideband" FMM [18]. We want to mention that the diagonal technique in the low-frequency regime is very different from that in the high-frequency regime: the complexity of the low-frequency diagonal scheme is $O(N^2)$ when used in the high-frequency regime while the diagonal scheme for high-frequency regime becomes unstable when applied to low-frequency problems. Detailed discussions of these techniques and their coupling are available in [18] and the references therein. More recent advances on the FMM technique include the kernel-independent FMMs that will be further discussed in Chapter 4, the time-domain FMM

for the Maxwell equations in [52], the elastic wave propagations [76, 77, 78], and the more recent fast direct solvers using the low separation rank properties in the FMM algorithms [19, 50].

The new-version FMMs for the commonly used Coulomb, screened Coulomb, and Helmholtz kernels are considered well-studied subjects. In this chapter, we focus on the screened Coulomb interaction $\Phi = \frac{e^{-\lambda r}}{r}$ and discuss two fundamental building blocks for the new-version FMM: (a) In Section 2.1, we describe the adaptive oct-tree structure of the new-version FMM for transmitting information; and (b) In Section 2.2, we summarize the classical approximation theory techniques and translation operators to collect, send, and receive information in the oct-tree based data structure. In Section 2.3, we present a pseudo-code to explain the implementation details of the new-version FMM algorithm. The screened Coulomb potential is also referred to as the Yukawa potential in high energy physics, or the Debye-Hückel potential in biophysics studies. The corresponding partial differential equation is the so-called linearized Poisson-Boltzmann equation which describes the electrostatic interactions in various biologically and physically important charged systems [23, 13, 64, 44, 43, 42, 27]. However, to the best of our knowledge, existing open source implementations of the new-version FMM for the Yukawa potential are scarce, perhaps due to the sophisticated theory and complicated programming. There is an urgent need for optimized FMM algorithms from many science and engineering applications. Inspired by this, in Section 2.4, we present our recently developed open source package **FMM-Yukawa** [40], and illustrate its performance and complexity by several numerical experiments. The code can be downloaded from [1], a website which has been developing by us to host educational and research resources for the scientific community.

2.1 Spatial Domain Adaptive Tree Structure

Unlike the algebraic structure based FFT algorithm, the FMM algorithm uses a data structure based on the spatial domain adaptive oct-tree for data communication. In this section, we describe how to construct the adaptive oct-tree and the corresponding data structure. Interested readers are also referred to [20] where the adaptive tree and data structure are discussed for the new-version FMM for the Laplace equation (Coulomb potential).

Given N particles, the new-version FMM starts with constructing an adaptive oct-tree consisting of a hierarchy of boxes. The root box, which is referred to as refinement level 0, is the smallest Cartesian box containing all N particles. Starting from level 0, we recursively obtain level $l+1$ through subdivision of the boxes at level l with more than s particles into eight boxes of equal size, where s is some pre-specified integer parameter. At each level of refinement, a table of non-empty boxes is maintained, so that once an empty box is encountered, its existence is forgotten and it is completely ignored by the subsequent process.

In this structure, a box is called a *parent* box if it contains more than s particles. Otherwise, it is referred to as a *childless* box or *leaf* box. A box c is said to be the *child* of box b if box c is obtained by a single subdivision of box b. On the other hand, such a box b is the *parent* of box c. Boxes resulting from the subdivision of a parent box are referred to as *siblings*. The *colleagues* of a box b consist of the boxes at the same level of b sharing at least one point with b, including box b itself. Apparently, a given box can have up to 27 colleagues in three dimensions.

One key observation in the FMM algorithm is the so-called "low separation rank" property for the well-separated boxes in the oct-tree structure. We say two sets $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_i\}$ are *well-separated* if there exists points $\boldsymbol{x}_0$ and $\boldsymbol{y}_0 \in \mathcal{R}^3$ and a real number $r > 0$ such that

$$
\begin{aligned}
&|\boldsymbol{x}_i - \boldsymbol{x}_0| < r \qquad \forall i = 1, \ldots, m \\
&|\boldsymbol{y}_j - \boldsymbol{y}_0| < r \qquad \forall j = 1, \ldots, n, \quad \text{and} \\
&|\boldsymbol{x}_0 - \boldsymbol{y}_0| > c \cdot r,
\end{aligned} \tag{2.2}
$$

where c is bounded below by some constant $c_0 > 2$. Two boxes B_1 and B_2 of the same size in the oct-tree structure are said to be *well-separated* if they are at least one box of the same size apart, i.e., if any sets of points $\{\boldsymbol{x}_i\} \subset B_1$ and $\{\boldsymbol{y}_i\} \subset B_2$ are well-separated (with $c_0 = \frac{4}{\sqrt{3}}$). As will be discussed in the next section, "information" in two well-separated boxes can be compressed either analytically using special basis functions (e.g., spherical harmonics for Laplace kernels) or numerically using singular value decomposition (SVD) before being sent out.

Well-separated boxes can also appear at different levels in the oct-tree structure. Assume that a box b can inherit information from its parent c while traversing down the tree structure, which contains the information from all well-separated boxes of c. Then we can define the *interaction list* of b as the union of boxes at the same level of b that are well-separated from b, but whose parents are not well-separated from b's parent c. Clearly, the information received from b's parent and the interaction list is equivalent to the information from all well-separated boxes of b.

For a given box b in the adaptive oct-tree structure, it may also need to "communicate" with boxes of different sizes. In Figure 2.1, we associate b with five lists of other boxes, determined by their positions with respect to b. We store list and pointers for parent-child relations for each box in the oct-tree structure.

In the following, we give the detailed definition for each list.

- List 1 of a box b will be denoted by U_b; it is empty if b is a parent box. If b is childless, U_b consists of b and all childless boxes adjacent to b.

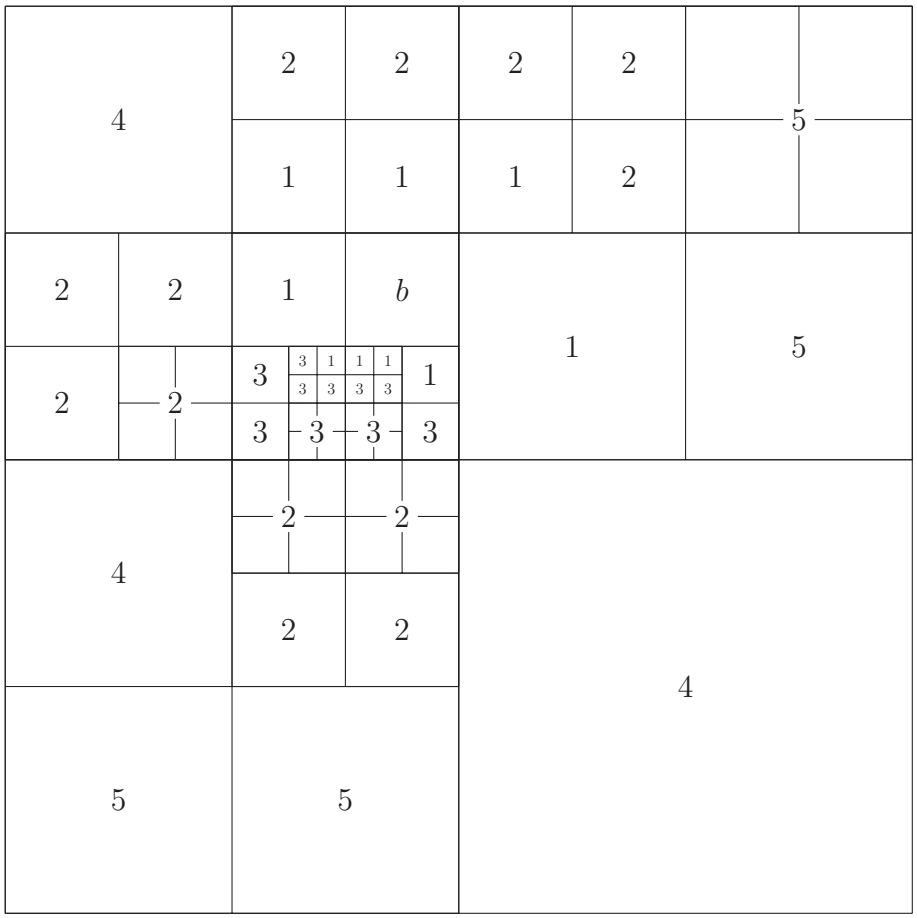


Figure 2.1: Box (b) and its associated lists 1 to 5

- List 2 of a box b will be denoted by V_b and is formed by all the children of the colleagues of b's parent that are well separated from b. List 2 is also referred to as the *interaction* list.

- List 3 of a box b will be denoted by W_b. W_b is empty if b is a parent box, and consists of all descendants of b's colleagues whose parents are adjacent to b, but who are not adjacent to b themselves, if b is a childless box. Note b is separated from each box w in W_b by a distance greater than or equal to the length of the side of w.

- List 4 of a box b will be denoted by X_b and is formed by all boxes c such that $b \in W_c$. Note that all boxes in List 4 are childless and larger than b.

- List 5 of a box b will be denoted by Y_b and consists of all boxes that are well separated from b's parent.

In the new-version FMM, plane-wave expansions are introduced to diagonalize the multipole-to-local translation from box b to its interaction list boxes to reduce the huge prefactor in $O(N)$. As the plane-wave expansions have directions, each box b is associated with six such expansions, each emanating from one face of the cube. Consequently, the interaction list for each box is further subdivided into six lists, associated with the six coordinate directions ($+z$, $-z$, $+y$, $-y$, $+x$, $-x$) in the three dimensional coordinate system. We will refer to the $+z$-direction as *up*, the $-z$-direction as *down*, the $+y$-direction as *north*, the $-y$-direction as *south*, the $+x$-direction as *east*, and the $-x$ direction as *west*.

The *Uplist* is demonstrated in Figure 2.2, and the definition for each list is given below:

1. The *Uplist* for a box b consists of those elements of the interaction list which lie above b and are separated by at least one box in the $+z$ direction (Figure 2.2).

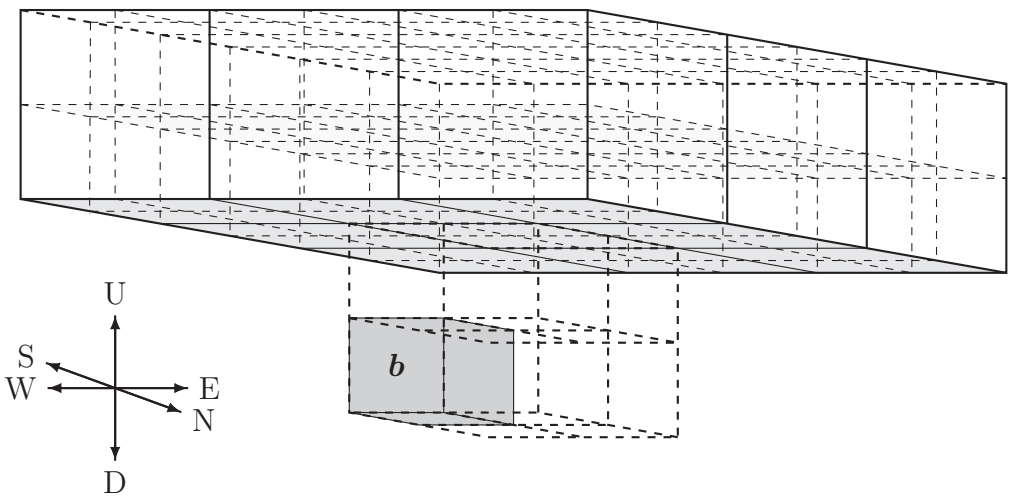


Figure 2.2: The *Uplist* for the box (b)

2. The *Downlist* for a box b consists of those elements of the interaction list which lie below b and are separated by at least one box in the $-z$ direction.

3. The *Northlist* for a box b consists of those elements of the interaction list which lie north of b, are separated by at least one box in the $+y$-direction, and are not contained in the Up or Down lists.

4. The *Southlist* for a box b consists of those elements of the interaction list which lie south of b, are separated by at least one box in the $-y$-direction, and are not contained in the Up or Down lists.

5. The *Eastlist* for a box b consists of those elements of the interaction list which lie east of b, are separated by at least one box in the $+x$-direction, and are not contained in the Up, Down, North, or South lists.

6. The *Westlist* for a box b consists of those elements of the interaction list which lie west of b, are separated by at least one box in the $-x$-direction, and are not contained in the Up, Down, North, or South lists.

Finally in this section, we want to mention that finding the optimal oct-tree and the corresponding data structure for the new-version FMM is still an open problem. In

[54], White et al. described another variant of the oct-tree and data structure used in FMM. In their scheme, a neighbor of a given cube is defined as any cube which shares a corner with the given cube (nearest neighbor) or shares a corner with the nearest neighbor (second-nearest neighbor). The interaction list of a cube is the set of cubes that are either the second-nearest neighbors of a given cube's parent or are children of the given cube's parent's nearest neighbors excluding the nearest or second-nearest neighbors of the given cube. The advantage of this approach is that fewer terms can be used in the multipole and local expansion since the " interacting" boxes are further apart. However, this will significantly change the implementation and performance of the plane-wave expansions in the new-version FMM. It is unclear which data structure is superior. As will be discussed in the next chapter, one of my future research projects is to generate a graph with nodes representing different interactions of the boxes and edges the associated costs, and study the "optimal" connecting tree on parallel computer architectures.

2.2 Approximations and Translations

Another important concept in the FMM algorithm is the extraction, storage, and transmission of the data (or "information") based on the oct-tree structure. In this section, using Yukawa potential as an example, we introduce the analytical spherical harmonics expansions and the plane-wave approximation, and discuss the translation operators which transmit the "information" from one box to another. Specifically, given N charges $q_1, q_2, \ldots, q_N$ at the location $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ in $\mathcal{R}^3$, we consider the pairwise interactions

$$\Phi(\boldsymbol{x_j}) = \sum_{\substack{i=1 \\ i \neq j}}^{N} q_i \cdot \frac{e^{-\lambda \|\boldsymbol{x}_j - \boldsymbol{x}_i\|}}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}, \qquad \lambda \in \mathcal{R}^+, \tag{2.3}$$

and discuss analytical formulas and numerical techniques for data management on the tree structure. As these formulas and theorems are well documented in the literature, we

neglect their proofs in this section, and refer interested readers to [31] for further details.

First, we study how to collect and compress the "information" from particles inside a box c using the multipole expansion as described in the following theorem.

Theorem 2.1 (Multipole Expansion ($\mathcal{T}_{SM}$ operator))**.** *Suppose that box c centered at the origin contains N sources of strengths $q_1, q_2, \ldots, q_N$ located at points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ with spherical coordinates $(\rho_1, \alpha_1, \beta_1), (\rho_2, \alpha_2, \beta_2), \ldots, (\rho_N, \alpha_N, \beta_N)$, respectively. Then the potential for any point $\boldsymbol{x} = (r, \theta, \phi)$ outside box c is given by a multipole expansion*

$$\begin{aligned}\Phi(\boldsymbol{x}) &= \sum_{i=1}^{N} q_i \cdot \frac{e^{-\lambda\|\boldsymbol{x}-\boldsymbol{x}_i\|}}{\|\boldsymbol{x}-\boldsymbol{x}_i\|} = \frac{2\lambda}{\pi}\sum_{i=1}^{N} q_i \cdot k_0(\lambda\|\boldsymbol{x}-\boldsymbol{x}_i\|) \\ &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n} M_n^m k_n(\lambda r)\cdot Y_n^m(\theta,\phi), \end{aligned} \tag{2.4}$$

where the multipole coefficients are

$$M_n^m = 8\lambda \sum_{i=1}^{N} q_i \cdot i_n(\lambda\rho_i)\cdot Y_n^{-m}(\alpha_i,\beta_i). \tag{2.5}$$

Furthermore,

$$\left|\Phi(\boldsymbol{x}) - \sum_{n=0}^{p}\sum_{m=-n}^{n} M_n^m k_n(\lambda r)\cdot Y_n^m(\theta,\phi)\right| = O\left(\frac{a}{r}\right)^p, \tag{2.6}$$

where a is the radius of the smallest sphere enclosing box c.

In the formulas, Y_n^m is the spherical harmonics of degree n and order m defined by

$$Y_n^m(\theta,\phi) = \sqrt{\frac{2n+1}{4\pi}}\sqrt{\frac{(n-|m|)!}{(n+|m|)!}}\cdot P_n^{|m|}(\cos\theta)e^{im\phi}, \tag{2.7}$$

where the associated Legendre functions P_n^m are defined using the usual Legendre poly-

nomial $P_n(x)$ by Rodrigues' formula

$$P_n^m(x) = (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_n(x), \tag{2.8}$$

and the modified spherical Bessel and modified spherical Hankel functions $i_n(r)$, $k_n(r)$ are defined in terms of the usual Bessel function $J_\nu(z)$ via

$$I_\nu(r) = i^{-\nu}J_\nu(ir) \quad (i=\sqrt{-1}), \quad K_\nu(r) = \frac{\pi}{2\sin\nu\pi}[I_{-\nu}(r) - I_\nu(r)]$$
$$i_n(r) = \sqrt{\frac{\pi}{2r}}I_{n+1/2}(r), \qquad k_n(r) = \sqrt{\frac{\pi}{2r}}K_{n+1/2}(r).$$

Notice that we have $\frac{a}{r} < 1$ when two boxes are well-separated, hence the expansion decays rapidly as p increases.

Next, we describe how to store the collected far-field "source" particle information in a local expansion, which is valid for all the "target" particles in the box which is well-separated from the box containing the source particles.

Theorem 2.2 (Local Expansion ($\mathcal{T}_{SL}$ operator))**.** *Suppose that N sources of strengths $q_1, q_2, \ldots, q_N$ are located at the points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ in $\mathcal{R}^3$ with spherical coordinates $(\rho_1, \alpha_1, \beta_1)$, $(\rho_2, \alpha_2, \beta_2), \ldots, (\rho_N, \alpha_N, \beta_N)$, respectively. Suppose further that all the points $\boldsymbol{x}_1$, $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ are located outside the sphere S_a of radius a centered at the origin. Then, for any point $\boldsymbol{x} \in S_a$ with coordinates (r, θ, ϕ), the potential $\Phi(\boldsymbol{x})$ generated by the sources $q_1, q_2, \ldots, q_M$ is described by the local expansion*

$$\Phi(\boldsymbol{x}) = \sum_{j=0}^{\infty}\sum_{k=-j}^{j} L_j^k i_j(\lambda r)\cdot Y_j^k(\theta,\phi), \tag{2.9}$$

where

$$L_j^k = 8\lambda\sum_{l=1}^{N} q_l k_j(\lambda\rho_l)\cdot Y_j^{-k}(\alpha_l,\beta_l). \tag{2.10}$$

Furthermore,

$$\left|\Phi(\boldsymbol{x}) - \sum_{j=0}^{p}\sum_{k=-j}^{j} L_j^k i_j(\lambda r)\cdot Y_j^k(\theta,\phi)\right| = O\left(\frac{r}{a}\right)^p. \tag{2.11}$$

In the tree-code algorithm, for each box at a certain level in the tree structure, its multipole expansion is formed by collecting information from the particles inside the box. As a result, at least $O(N)$ operations are required for each level to account for communication with N particles. Notice that the tree has approximately $\log N$ levels, hence the complexity of tree-code algorithm is at least $O(N \log N)$. In order to reduce the cost in forming multipole expansions for all boxes, in the original FMM [34], a divide-and-conquer strategy is applied. For each parent box, instead of communicating with the particles directly, its multipole expansion is derived by shifting and merging its children's multipole expansions (which already contain the collected and compressed particle information) using the following multipole-to-multipole ($\mathcal{T}_{MM}$) translation operator:

Theorem 2.3 ($\mathcal{T}_{MM}$ translation)**.** *Consider a box b centered at the origin and its child c centered at $\boldsymbol{x}_0 = (\rho, \alpha, \beta)$, and a point $\boldsymbol{x} = (r, \theta, \phi)$ in the far-field boxes of b. Assume the potential due to charges in box c is given by the multipole expansion*

$$\Phi(\boldsymbol{x}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} O_n^m k_n(\lambda r')\cdot Y_n^m(\theta',\phi'), \tag{2.12}$$

where (r', θ', ϕ') are the spherical coordinates of the vector $\boldsymbol{x} - \boldsymbol{x}_0$. Then, the field can also be described by a shifted multipole expansion:

$$\Phi(\boldsymbol{x}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} M_n^m k_n(\lambda r)\cdot Y_n^m(\theta,\phi). \tag{2.13}$$

The linear operator mapping the old multipole coefficients $\{O_n^m\}$ to the new multipole coefficients $\{M_n^m\}$ is denoted by $\mathcal{T}_{MM}$.

After forming all the multipole expansions in an upward sweep, a downward sweep

from refinement level 0 to the finest level is performed to form each box's local expansion which contains the far-field particle contributions. For a box c at level l, a translation is first carried out to shift its parent b's local expansion (which contains particle information from b's far-field) to an equivalent local expansion about the center of box c, via the local-to-local ($\mathcal{T}_{LL}$) operator as follows:

Theorem 2.4 ($\mathcal{T}_{LL}$ translation)**.** *Consider a point $\boldsymbol{x} = (r, \theta, \phi)$ in box c and the local expansion of c's parent box b*

$$\Phi(\boldsymbol{x}) = \sum_{n=0}^{p} \sum_{m=-n}^{n} L_n^m i_n(\lambda r) \cdot Y_n^m(\theta, \phi). \tag{2.14}$$

Assume box c is centered at $\boldsymbol{x}_0 = (\rho, \alpha, \beta)$, then the local expansion of c is given by

$$\Phi(\boldsymbol{x}) = \sum_{n=0}^{p} \sum_{m=-n}^{n} N_n^m i_n(\lambda r') \cdot Y_n^m(\theta', \phi'), \tag{2.15}$$

where (r', θ', ϕ') are the spherical coordinates of the vector $\boldsymbol{x} - \boldsymbol{x}_0$. The linear operator mapping the old local coefficients $\{L_n^m\}$ to the new local coefficients $\{N_n^m\}$ is denoted by $\mathcal{T}_{LL}$.

In the second step of the downward pass, box c receives contributions from particles which are located inside a region formed by subtracting its parent b's far-field from its own far-field. Boxes in this region are members of the aforementioned list 2 or the interaction list of the given box c. Instead of communicating with the particles in this region directly, a multipole-to-local translation ($\mathcal{T}_{ML}$) is performed to convert the multipole expansions of each interaction list box into a local expansion which is then merged into c's local expansion. The $\mathcal{T}_{ML}$ translation operator is defined as follows:

Theorem 2.5 ($\mathcal{T}_{ML}$ translation)**.** *Suppose box b centered at $\boldsymbol{x}_0 = (\rho, \alpha, \beta)$ is an interaction list box of c, and its multipole expansion is given by the same formula in Equation (2.12). Then for any point $\boldsymbol{x} = (r, \theta, \phi)$ in c centered at the origin, the potential due*

to charges in b can be described by a local expansion

$$\Phi(\boldsymbol{x}) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k i_j(\lambda r) \cdot Y_j^k(\theta, \phi). \tag{2.16}$$

The linear operator mapping the multipole coefficients $\{M_n^m\}$ *to the local coefficients* $\{L_n^m\}$ *is denoted by* $\mathcal{T}_{ML}$.

At the end of the downward sweep, we evaluate local expansion at every particle location in each childless box, and then compute the near-field interactions directly.

Recall that in three dimensions, the interaction list of a given box can contain up to 189 boxes. Additionally, the operation complexity of the $\mathcal{T}_{ML}$ operator is $O(p^4)$ assuming p^2 terms are used in each expansion. Consequently, this original multipole-to-local translation requires prohibitive $189p^4$ operations for each box, which makes the original FMM less attractive to large-scale problems. Therefore, we next introduce the plane-wave expansions analogous to the work of Greengard and Rokhlin in [35] to diagonalize the $\mathcal{T}_{ML}$ operator. In three dimensions, six different plane-wave expansions are introduced for each face of the box. We use the upward (or $+z$) direction to illustrate the idea and omit the details of other directions which can be processed in a similar manner. For boxes in the uplist of a box (see Figure 2.2), its multipole expansion is first translated into an exponential expansion by the multipole-to-exponential ($\mathcal{T}_{ME}$) operator as follows:

Theorem 2.6 ($\mathcal{T}_{ME}$ translation)**.** *For a point* $\boldsymbol{x}$ *in the up direction with spherical coordinates* (r, θ, ϕ)*, assume the potential* $\phi(\boldsymbol{x})$ *is approximated by the truncated multipole expansion (due to charges in box c centered at the origin) with error bound* ϵ

$$\left| \Phi(\boldsymbol{x}) - \sum_{n=0}^{p} \sum_{m=-n}^{n} O_n^m \cdot Y_n^m(\theta, \phi) k_n(\lambda r) \right| < \epsilon, \tag{2.17}$$

then with error bound $O(\epsilon)$, $\Phi(\boldsymbol{x})$ can be approximated by

$$\lambda \sum_{k=1}^{s(\epsilon)} \sum_{j=1}^{M_k} W(k,j) \cdot e^{-(u_k+\lambda)z} \cdot e^{i\sqrt{u_k+2u_k\lambda}\cdot(x\cos\alpha_{j,k}+y\sin\alpha_{j,k})}, \tag{2.18}$$

where (x, y, z) are the Cartesian coordinates of $\boldsymbol{x}$, the coefficients $W(k, j)$ are given by

$$\frac{\pi w_k}{2\lambda M_k} \sum_{m=-p}^{p} i^{|m|} e^{im\alpha_{j,k}} \sum_{n=|m|}^{p} O_n^m \sqrt{\frac{2n+1}{4\pi}} \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}\left(\frac{\lambda+u_k}{\lambda}\right), \tag{2.19}$$

for $k = 1, \ldots, s(\epsilon)$, $j = 1, \ldots, M_k$. The linear mapping from $\{O_n^m\}$ to $\{W(k, j)\}$ is referred to as the $\mathcal{T}_{ME}$ translation operator.

In Equation (2.18), the weights w_k and nodes u_k are computed according to the generalized Gaussian quadrature as discussed in [70]. Once the multipole expansion is converted into the exponential expansion about the center of box c (the origin), shifting it to a new center $\boldsymbol{x}_0 = (x_0, y_0, z_0)$ of an interaction list box of c can be done diagonally, as described by the exponential-to-exponential ($\mathcal{T}_{EE}$) operator as follows:

Theorem 2.7 ($\mathcal{T}_{EE}$ translation)**.** *For the exponential expansion in Equation (2.18) of box c centered at the origin and a box in the interaction list centered at $\boldsymbol{x}_0 = (x_0, y_0, z_0)$, the shifted exponential expansion for any point (x, y, z) is given by*

$$\sum_{k=1}^{s(\epsilon)} \sum_{j=1}^{M_k} V(k,j) \cdot e^{-(u_k+\lambda)(z-z_0)} e^{i\sqrt{u_k^2+2u_k\lambda}\cdot((x-x_0)\cos\alpha_{j,k}+(y-y_0)\sin\alpha_{j,k})}, \tag{2.20}$$

where

$$V(k,j) = W(k,j) \cdot e^{-(u_k+\lambda)z_0} \cdot e^{i\sqrt{u_k^2+2u_k\lambda}\cdot(x_0\cos\alpha_{j,k}+y_0\sin\alpha_{j,k})}, \tag{2.21}$$

for $k = 1, \ldots, s(\epsilon)$, $j = 1, \ldots, M_k$. The linear operator mapping the coefficients $\{W(k, j)\}$ to the coefficients $\{V(k, j)\}$ is denoted by $\mathcal{T}_{EE}$.

Once a box has collected all the exponential expansions from its interaction list boxes,

the exponential expansions can be translated to a local expansion about the box center using the exponential-to-local ($\mathcal{T}_{EL}$) operator as follows:

Theorem 2.8 ($\mathcal{T}_{EL}$ translation)**.** *Suppose that the potential for $\boldsymbol{x} = (x, y, z)$ is given by Equation (2.18), then there exists an integer p, such that*

$$\left| \Phi(\boldsymbol{x}) - \sum_{n=0}^{p} \sum_{m=-n}^{n} L_n^m i_n(\lambda r) Y_n^m(\theta, \phi) \right| = O(\epsilon), \tag{2.22}$$

where (r, θ, ϕ) are the spherical coordinates of $\boldsymbol{x}$ with respect to the box center and

$$L_n^m = (-1)^n i^{|m|} \sqrt{4\pi} \sqrt{2n+1} \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} \sum_{k=1}^{s(\epsilon)} P_n^{|m|} \left(\frac{u_k + \lambda}{\lambda} \right) \sum_{j=1}^{M_k} W(k, j) e^{im\alpha_{j,k}} \tag{2.23}$$

for $n = 0, \ldots, p$, $m = -n, \ldots, n$. The linear operator converting the coefficients $\{W(k, j)\}$ into the coefficients $\{L_n^m\}$ is denoted by $\mathcal{T}_{EL}$.

From the complexity perspective, by using the decomposition $\mathcal{T}_{ML} = \mathcal{T}_{EL} \circ \mathcal{T}_{EE} \circ \mathcal{T}_{ME}$, the previous prohibitive $189p^4$ operation counts are now reduced to approximately $2p^4 + 189p^2$, which potentially can be further reduced to roughly $6p^3 + 40p^2$. To achieve that, we shall first apply the point-and-shoot technique and rotate the multipole expansion so that $\mathcal{T}_{ME}$ operator can be performed along the z-axis ($3p^3$ operations, more details in [35]). Next, the exponential expansions of box c and its siblings shall be merged before being sent out to their common interaction list boxes, which reduces the operations from $189p^2$ to $40p^2$. Lastly, we shall translate each box's exponential expansion into a local expansion and then rotate it back to the original coordinate system using the point-and-shoot technique (another $3p^3$ operations).

2.3 Algorithm Structure of Adaptive FMM

In this section, we present a pseudo-code to explain the algorithmic structure of the adaptive new-version FMM. To simplify our discussions, we introduce the following notations. The computational domain is denoted by B_0 and the set of all nonempty boxes at refinement level l is denoted by B_l. For each box b, its associated five lists are denoted by U_b, V_b, W_b, X_b, Y_b, respectively. V_b is further subdivided into $Uplist(b)$, $Downlist(b)$, $Northlist(b)$, $Southlist(b)$, $Eastlist(b)$, and $Westlist(b)$ to utilize the plane-wave expansions. Also, each box b is associated with the following fourteen expansions:

- A multipole expansion Φ_b of the form (2.4) representing the potential generated by charges inside b; it is valid in $\mathcal{R}^3 \setminus \{U_b \cup W_b\}$.
- A local expansion Ψ_b of the form (2.9) representing the potential generated by all charges outside $U_b \cup W_b$; it is valid inside box b.
- Six outgoing exponential expansions W_b^{Up}, W_b^{Down}, W_b^{North}, W_b^{South}, W_b^{East}, and W_b^{West} of the form (2.18), representing the potential generated by all charges located inside b and valid in $Uplist(b)$, $Downlist(b)$, $Northlist(b)$, $Southlist(b)$, $Eastlist(b)$, and $Westlist(b)$, respectively.
- Six incoming exponential expansions V_b^{Up}, V_b^{Down}, V_b^{North}, V_b^{South}, V_b^{East}, and V_b^{West} of the form (2.18), representing the potential inside b generated by all charges in $Downlist(b)$, $Uplist(b)$, $Southlist(b)$, $Northlist(b)$, $Westlist(b)$, and $Eastlist(b)$, respectively.

Pseudo-Code: Adaptive FMM Algorithm

Initialization

Choose precision ϵ and the order of the multipole expansions p. Choose the maximum number s of charges allowed in a childless box.

Generating Oct-Tree Structure

Step 1

for $l = 0, 1, 2, \ldots$

for each box $b \in B_l$

if b contains more than s particles **then**

Divide b into eight child boxes. Ignore empty children and

add the nonempty child boxes to B_{l+1}.

endif

end

end

Comment [Denote the maximum refinement level obtained by l_{max} and the total number of boxes created by $nboxes$.]

for each box b_i, $i = 1, 2, \ldots, nboxes$

Create Lists U_{b_i}, V_{b_i}, W_{b_i}, X_{b_i}.

Split V_{b_i} into Up, $Down$, $North$, $South$, $East$, $West$ lists.

end

Upward Pass

Step 2

for $l = l_{max}, \ldots, -1, 0$

for each box b in B_l

if b is childless **then**

Use Theorem 2.1 to form multipole expansion Φ_b.

else

Use operator $\mathcal{T}_{MM}$ to merge multipole expansions from its children into Φ_b.

endif

end

end

Downward Pass

Step 3

Comment [For each box b, add to its local expansion the contribution due to particles in X_b.]

for each box b_i, $i = 1, \ldots, nboxes$

for each box $c \in X_{b_i}$

if the number of particles $b_i \leq p^2$ **then**

Comment [The number of particles in b_i is small. It is faster to use direct calculation than to generate the contribution to the local expansion Ψ_{b_i} due to charges in c; act accordingly.]

Calculate potential field at each particle point in b_i directly from particles in c.

else

Comment [The number of particles in b_i is large. It is faster to generate the contribution to the local expansion Ψ_{b_i} due to particles in c than to use direct calculation; act accordingly.]

Generate a local expansion at b_i's center due to particles in c using Theorem 2.2, and add to Ψ_{b_i}

endif

end

end

Step 4

Comment [For each box b on level l with $l = 2, 3, \ldots, l_{max}$ and each direction $Dir = Up, Down, North, South, East, West$, create from box b's multipole expansion the outgoing exponential W_b^{Dir} in direction Dir, using Theorem 2.6. Translate W_b^{Dir} to the center of each box $c \in Dirlist(b)$ using Theorem 2.7 and add the translated expansions to its incoming exponential expansion V_c^{Dir}. Lastly, convert V_c^{Dir} into a local expansion using Theorem 2.8 and add it to Ψ_c.]

for $l = 2, 3, \ldots, l_{max}$

  for $Dir = Up, Down, North, South, East, West$

    for each box $b \in B_l$

      Convert Φ_b into W_b^{Dir} by Theorem 2.6.

      for each box $c \in Dirlist(b)$

        Translate W_b^{Dir} to the center of box c using Theorem 2.7.

        Add the translated expansion to V_c^{Dir}.

      end

    end

    for each box $c \in B_l$

      Convert V_c^{Dir} into a local expansion using Theorem 2.8,

      and add it to Ψ_c.

    end

  end

end

Step 5

Comment [For each parent box b, shift the center of its local expansion to its children.]

for each box b_i, $i = 1, 2, \ldots, nboxes$

if b_i is a parent box **then**

Shift the local expansion Ψ_{b_i} to the centers of its children using the operator $\mathcal{T}_{LL}$, and add the translated expansions to children's local expansions.

endif

end

Evaluation of Potentials

Step 6

Comment [Evaluate the local expansion at leaf nodes.]

for each box $b_i = 1, 2, \ldots, nboxes$

if b_i is childless **then**

Calculate the potential at each charge in b_i from the local expansion Ψ_{b_i}.

endif

end

Step 7

Comment [For each childless box b, evaluate the potential due to particles in W_b.]

for each box b_i, $i = 1, 2, \ldots, nboxes$

if b_i is childless **then**

for each box $c \in W_{b_i}$

if the number of charges in $c \leq p^2$ **then**

Comment [The number of charges in c is small. It is faster to use direct calculation than to evaluate the multipole expansion Φ_c.]

Calculate the potential at each charge b_i directly from particles in c.

else

Comment [The number of charges in c is large. It is faster to evaluate the expansion Φ_c than to use direct calculation.]

Calculate the potential at each charge in b_i

from multipole expansion Φ_c

endif

end

endif

end

Step 8

Comment [Local direct interactions.]

for each box b_i, $i = 1, 2, \ldots, nboxes$

if b_i is childless **then**

Calculate the potential at each charge in b_i directly

due to all charges in U_{b_i}

endif

end

2.4 FMM-Yukawa in FMMsuite

Perhaps due to its complexity both in mathematical theory and programming, to the best of our knowledge, we are unaware of any previous open source implementations of the new-version FMM. In order to meet the urgent needs from the scientific computing community, we have developed an open source package called **FMMSuite**, which includes the new-version FMM for the Laplace, Yukawa, and low-frequency Helmholtz equations. The codes are available for download from our website *http://fastmultipole.org*

under GPL 2.0 license agreement. The website also hosts many educational and research resources on integral equation methods and fast algorithms.

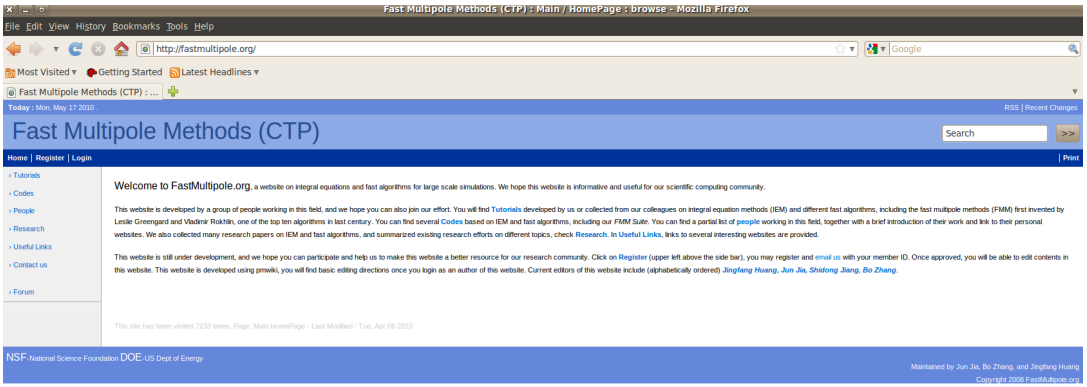


Figure 2.3: Screenshot of Fast Multipole Methods website

In this section, we present the details of the FMM-Yukawa solver in the FMMSuite package, including the installation instructions, a sample driver file, and several numerical experiments to demonstrate the performance of the solver. The manual for the FMM-Yukawa solver can also be found in [40].

2.4.1 FMM-Yukawa Installation Instructions

After the package is downloaded and extracted to local computer, the user will find the following directories:

- doc: contains license and readme.txt
- src: source files and makefile

The package has been successfully complied using the Intel® complier for Linux and GNU©F95 complier. The complied executable program is called "fmm" by default which can be changed by modifying the "Makefile". One function which may be machine dependent is the subroutine for getting the current CPU clock information for timing purposes. The users should check their computer platform and modify "second.f" accordingly.

The main driver for FMM-Yukawa is called "adapyukdriver.f". The main function is FMMYUK-A in the file "fmmadapyuk.f", which calls the subroutine D3MSTRCR to generate the adaptive tree structure, and YADAPFMM for calculating the force field and potential. There are three important parameters defined in the header file "parmayuk.h". NBOX is the maximum number of particles (s) allowed in a childless box; NTERMS is the number of terms in first summation of the multipole and local expansions; NLAMBS is the number of terms in the first summation of the exponential expansion. Currently only three- and six- digits accuracies are allowed, and more options will be added in future updates. Input variables include the screening factor BETA, the number of charges NATOMS, charge locations ZAT(3,NATOMS), and the charge CHARGE(NATOMS) carried by each particle. FMM-YUKAWA will calculate and output the potential POT(NATOMS) and the field FIELD(3,NATOMS).

2.4.2 A Sample Driver File

In this section, we provide a sample driver file to explain how FMM-YUKAWA can interact with other existing codes.

A driver file for FMM-Yukawa

```
      IMPLICIT NONE
      INTEGER *4 NATOMS,IER
      PARAMETER (NATOMS=1000000)
      REAL *8 BETA
      REAL *8 ZAT(3,NATOMS),CHARGE(NATOMS)
      REAL *8 POT(NATOMS),FIELD(3,NATOMS)
c
c—set up parameters.
c
```

```
      BETA=0.1D0
c
c—generate charges and their locations.
c
      CALL DUMMY(NATOMS,ZAT,CHARGE)
c
c—call fmm to calculate the potential and field.
c
      CALL FMMYUK_A(BETA,NATOMS,ZAT,CHARGE,POT,FIELD,IER)
c
      STOP
      END
```

2.4.3 Test Run Description

In this section, we present two numerical examples to show the efficiency and accuracy of the FMM-Yukawa package. In the experiment, we use a machine with Intel® T9400 processor at 2.53 GHz clock rate and 3GB memory. The results are summarized in Tables 2.1 – 2.4. In the table, N is the number of particles for which calculations have been preformed; s is the maximum number of particles allowed for a childless box (for each N, we run the program with multiple choices of s and the one yielding the optimal timing is reported here); l_{max} is the maximum number of refinement of the tree structure; *nboxes* is the total number of boxes created; T_{fmm} and T_{dir} refer to the timing results for FMM algorithm and direct calculation, respectively. In the table, all timings are given in seconds and the storage is measured in megabytes.

In the tests, we consider two distributions of the particles, namely, uniformly distributed inside a unit box $[-0.5, 0.5]^3$ and on a spherical surface centered at the origin of radius 0.5. For each distribution, we run the program with three- and six- digit accuracy. The screening factor is set to be 0.1 for all tests. In each table, T_{fmm} includes both the precomputation time

and the actual FMM calculation time. T_{dir} is estimated by extrapolating the CPU time for evaluating the potentials at 400 particle locations directly since calculating the entire system in this fashion would require prohibitive amounts of CPU time without providing much useful information. Similarly, the accuracy of the algorithm is calculated at those locations according to the formula

$$E = \sqrt{\frac{\sum_{i=1}^{400} \left| \Phi(\boldsymbol{x_i}) - \tilde{\Phi}(\boldsymbol{x_i}) \right|^2}{\sum_{i=1}^{400} |\Phi(\boldsymbol{x_i})|^2}}, \tag{2.24}$$

where $\Phi(\boldsymbol{x}_i)$ is the result obtained from direct calculation and $\tilde{\Phi}(\boldsymbol{x}_i)$ is the result obtained from FMM algorithm.

For each table, the timing results are further plotted as a function of N (see Figure 2.4). In the figure, we can observe that the actual CPU time required by the FMM-Yukawa package grows approximately linearly with N. Lastly, we want to mention that the numerical results suggest that the break-even point of the new version FMM is roughly 750 for three-digit and 1500 for six-digit accuracy, which is an estimate of the constant associated with $O(N)$ in the complexity analysis.

N	s	l_{max}	$nboxes$	p	S_{exp}	Storage	T_{fmm}	T_{dir}	Error
750	40	2	41	9	67	1.20	0.07	0.08	$9.7{\cdot}10^{-5}$
1500	30	3	165	9	67	1.90	0.15	0.32	$1.7{\cdot}10^{-4}$
5000	30	4	649	9	67	4.60	0.40	3.4	$1.9{\cdot}10^{-4}$
10000	40	4	681	9	67	5.08	0.77	14	$2.6{\cdot}10^{-4}$
20000	30	5	4729	9	67	26.44	2.25	54	$3.6{\cdot}10^{-4}$
50000	30	5	4776	9	67	28.52	4.29	337	$3.5{\cdot}10^{-4}$
100000	40	5	4777	9	67	31.57	11.02	1349	$4.4{\cdot}10^{-4}$
200000	40	6	33506	9	67	184.97	18.82	5480	$4.6{\cdot}10^{-4}$
500000	40	6	37363	9	67	223.05	47.32	33900	$4.6{\cdot}10^{-4}$

Table 2.1: Timing results for FMM-Yukawa for 3-digit accuracy with charges uniformly distributed inside the cube $[-0.5, 0.5]^3$

N	s	l_{max}	$nboxes$	p	S_{exp}	Storage	T_{fmm}	T_{dir}	Error
750	70	2	41	18	311	3.05	0.16	0.08	$5.9{\cdot}10^{-8}$
1500	50	3	81	18	311	3.92	0.32	0.32	$6.8{\cdot}10^{-8}$
5000	50	4	197	18	311	6.52	0.91	3.4	$1.1{\cdot}10^{-7}$
10000	50	4	681	18	311	16.76	1.71	14	$1.5{\cdot}10^{-7}$
20000	80	4	681	18	311	17.37	3.86	54	$1.8{\cdot}10^{-7}$
50000	80	5	4776	18	311	103.31	10.65	337	$2.3{\cdot}10^{-7}$
100000	80	5	4777	18	311	106.38	18.47	1349	$2.8{\cdot}10^{-7}$
200000	80	6	4945	18	311	115.94	46.27	5480	$2.5{\cdot}10^{-7}$
500000	60	6	37363	18	311	800.06	101.45	33862	$3.9{\cdot}10^{-7}$

Table 2.2: Timing results for FMM-Yukawa for 6-digit accuracy with charges uniformly distributed inside the cube $[-0.5, 0.5]^3$

N	s	l_{max}	$nboxes$	p	S_{exp}	Storage	T_{fmm}	T_{dir}	Error
750	40	3	74	9	67	1.40	0.09	0.08	$1.2{\cdot}10^{-4}$
1500	40	4	104	9	67	1.60	0.15	0.32	$8.8{\cdot}10^{-5}$
5000	50	5	391	9	67	3.29	0.36	3.4	$1.3{\cdot}10^{-4}$
10000	50	6	639	9	67	4.87	0.84	14	$1.2{\cdot}10^{-4}$
20000	40	7	1702	9	67	10.92	2.01	54	$1.9{\cdot}10^{-4}$
50000	30	9	5899	9	67	34.27	4.18	337	$1.9{\cdot}10^{-4}$
100000	30	10	10685	9	67	55.50	8.25	1349	$2.3{\cdot}10^{-4}$
200000	30	11	23044	9	67	131.33	16.13	5480	$2.2{\cdot}10^{-4}$
500000	30	12	57950	9	67	328.60	38.99	33862	$2.5{\cdot}10^{-4}$

Table 2.3: Timing results for FMM-Yukawa for 3-digit accuracy with charges distributed on the surface of a sphere

N	s	l_{max}	$nboxes$	p	S_{exp}	Storage	T_{fmm}	T_{dir}	Error
750	70	2	65	18	311	3.55	0.17	0.08	$1.0{\cdot}10^{-7}$
1500	60	3	100	18	311	4.31	0.31	0.32	$7.1{\cdot}10^{-8}$
5000	60	5	323	18	311	9.11	0.93	3.4	$0.5{\cdot}10^{-8}$
10000	50	6	639	18	311	15.90	1.79	14	$8.7{\cdot}10^{-8}$
20000	40	7	1702	18	311	38.34	4.17	54	$1.0{\cdot}10^{-7}$
50000	80	7	2079	18	311	47.92	9.39	337	$1.6{\cdot}10^{-7}$
100000	60	9	5976	18	311	131.01	18.32	1349	$1.8{\cdot}10^{-7}$
200000	60	10	10987	18	311	240.03	37.81	5480	$1.7{\cdot}10^{-7}$
500000	50	12	32715	18	311	704.60	89.79	33862	$1.8{\cdot}10^{-7}$

Table 2.4: Timing results for FMM-Yukawa for 6-digit accuracy with charges distributed on the surface of a sphere

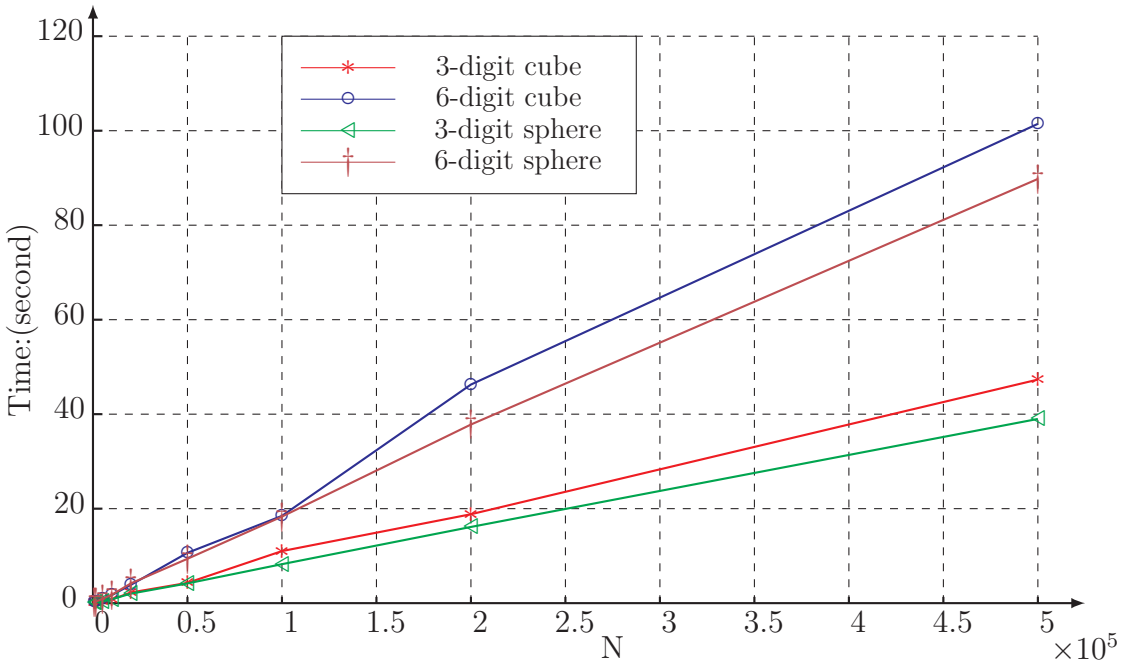


Figure 2.4: Linear relationship between CPU time and N

Chapter 3

Parallelization of Fast Multipole Method on Multicore Architectures

In this chapter, we focus on the parallelization of FMM to further accelerate the performance of the algorithm. Particularly, we will discuss available options in architecture platforms and develop parallelization strategies. It is well known that parallelization is one of the crucial factors left for further acceleration of the FMM algorithm since the complexity of the sequential FMM has already been pushed to its own limit with the introduction of plane-wave expansion. Although the new-version FMM breaks even with direct calculation at $N = 750$ for 3-digit accuracy and $N = 1500$ for 6-digit accuracy, which is a remarkable accomplishment, it still consumes prohibitively large amount of time in many large-scale simulations (hundreds of millions unknowns) of biomolecular, physical and chemical systems. There is an urgent need to further accelerate FMM algorithm by another two or three orders to enable those important research at affordable costs. Therefore, we propose a parallel new-version adaptive FMM based on graph-theoretic approach in this chapter.

This chapter is organized as follows. In Section 3.1, we provide a brief overview of the development of FMM in the last two decades, in terms of the algorithm employed, particle distribution, architecture topology, and the application contexts. In Section 3.2, we discuss the absolute critical path, maximum concurrency potential of the algorithm, and the impact of architecture constraints on the parallelization. In Section 3.3, we develop a parallelization strategy following a graph-theoretic approach based on the spatio-temporal partition of the FMM interaction graph. In Section 3.4, we give an implementation of the parallelization scheme

and report several numerical results to demonstrate the efficiency of the scheme.

3.1 Precursors of Parallel FMM

In this section, we give a brief overview of previous studies and efforts in developing parallel FMM. They differ in terms of the algorithm employed, particle distribution, architecture topology, and the application contexts.

The parallelization on FMM starts with the uniform version first. The first study comes from Greengard and Gropp in 1990 on 2D FMM [30]. They showed that with $O(N)$ processors, the overall complexity is $O(\log N)$. They also presented numerical results on a shared memory machine, the Encore Multimax 320. In 1991, Zhao and Johnsson developed a parallel version of the 3D uniform FMM using high performance Fortran for the Connection Machine system model CM-2 [80]. In [41], Board et al. implemented a parallel version of 3D FMM on a network of workstations using the coordination language Linda [17] and incorporated it into the molecular dynamics program MD of Windemuth and Schulten [68]. On this distributed memory system, the authors employed a master-worker scheme. A master process running on one processor separates the computational work into tasks and sends out one task per processor to the network. The remaining tasks are then assigned to the processors as they finish their previously assigned tasks.

Historically, the first successful parallel hierarchical N-body method for non-uniform particle distribution on distributed memory systems was obtained by Warren and Salmon [66]. The key ideas in this paper were the orthogonal recursive bisection (ORB) and locally essential tree (LET). The idea in ORB partitioning is to recursively divide the computational domain space into two subspaces with equal costs, until there is one subspace per processor. To provide load balancing, the cost of a subspace is defined as the sum of the profiled costs of all particles in the subspace. The idea in LET stems from the following observation. Every body sees only a fraction of the complete tree. The distant parts are seen at a coarser level of detail while the nearby sections are seen all the way down to the leaves. As nearby bodies see similar trees, a LET is then defined to be the union of all the trees seen by all the bodies in an ORB domain. It is the

data that will be required to carry on the computation in the domain. Once the LET is obtained in an ORB domain, one can proceed exactly as in the sequential case, allowing employing highly tuned sequential assembly language without regard to communication, synchronization or other parallel issues. However, ORB and LET introduce several new data structures, including a binary ORB tree that is distinct from the FMM tree. Additionally, ORB and LET are nontrivial to implement and debug, and have significant runtime overhead particularly as the number of processors increases [61]. As a result, other domain decomposition methods based on hashed tree and space-filling curves have been introduced to achieve better efficiency on the distributed memory systems.

The hashed oct-tree structure along with space-filling curves was first introduced by Warren and Salmon [67]. In this paper, instead of using pointers, the topology of the tree is implicitly represented by mapping the cell spatial locations and levels into keys, which are then translated into memory locations via a hash table. This scheme provides a uniform addressing mechanism to retrieve data across multiple processors. It also produces a good domain decomposition by cutting the one dimensional list of sorted body key ordinates into N_p (number of processors) equal pieces, weighted by the amount of work corresponding to each body. Two particular mappings were studied in the paper, namely, Morton ordering and Peano-Hilbert ordering. Although the latter provides a better decomposition theoretically, it does not lead to improved performance in practice.

In [61], the costzone decomposition has been shown to be more efficient than ORB on common address space architecture. In costzones, the tree is conceptually laid out in a two-dimensional plane, with a cell's children laid out from left to right in increasing order of child number. The cost of every particle is stored with the particle. Every internal cell holds the sum of the costs of all particles that are contained within it. These cell costs are computed during the upward pass. The total cost in the domain is divided among processors so that every processor has a contiguous, equal zone of costs. Which cost zone a particle belongs to is conceptually determined by the total cost up to that particle in an inorder traversal of the tree. In the costzones algorithm, processors descend the tree in parallel, picking up the particles that belong to their cost zone. To make sure that contiguity in the planarized tree always

corresponds to contiguity in space, an ordering strategy has been developed. In this solution, the order in which children are numbered is not the same for all cells. The orderings of a cell C's children are determined by: (a) the ordering of C's siblings; and (b) which child of its parent C is in that ordering.

A single board of GPUs (graphics processing unit) can be viewed as shared memory at the API level. The model for GPU computing is to use CPU and GPU together in a heterogeneous computing model. The sequential part of the application runs on the CPU and the computationally-intensive part runs on the GPU. In 2007, Nyland et al. provided an CUDA (computed unified device architecture) implementation on GPUs computing the pairwise interactions of N bodies using direct interaction method [56]. In 2008, Gumerov and Duraiswami [36] mapped the original FMM for Laplace kernel onto the GPU architectures. In the current state, due to bandwidth concerns, the problem size to be handled is approximately in the order of 10^6 particles. Furthermore, as GPU used to perform single precision floating point arithmetic, high-accuracy requirement can decrease the parallel speedup by a factor of up to 10 times [29].

Recently, there is an ongoing paradigm shift in parallel computing due to the developments both in hardware (multicore processor) and software (multi-threading library). Compared with traditional platform, applying multi-threading technique on multicore machine has several advantages: (a) it provides large on-chip memory; (b) it provides transparent cache memory access management; and (c) it provides flexible task distribution and scheduling among threads. In the remaining of this chapter, we establish a detailed parallelization scheme of FMM on this platform.

3.2 Analytical Parallel FMM

3.2.1 The Absolute Critical Path

In this section, we analyze the absolute critical path for evaluating the pairwise interactions of N particles under the parallel random access machine (PRAM) architecture. The critical

path analysis is fundamental in implementing parallel algorithms since it identifies the longest chain of dependent calculations. PRAM architecture is a shared memory abstract machine used by parallel algorithm designers to estimate the time complexity of the algorithm. It neglects issues such as synchronization and communication, but provides any (problem size-dependent) number of processors. In the analysis, we assume that interaction between particles is described by a function in the form of $\mathcal{K}(r)$ and the particle distribution can be either uniform or adaptive. The result of the analysis is summarized in Table 3.1, and will be elaborated throughout this section.

	Direct	Uniform FMM	Adaptive FMM
Sequential	$O(N^2)$	$O(N)$	$O(N)$
Ideal Parallel	$O(\log N)$	$O(\log N)$	$O(\log N)^*$

Table 3.1: Complexity analysis of parallel particle simulation

In the table, results for the sequential computing are well-known and therefore we focus on ideal parallel by which we mean parallelization under the PRAM architecture. Although the results are $O(\log N)$ for each case at first glance, the causes are different. For direct algorithm, it boils down to how fast one can add N numbers together. Using divide-and-conquer, the complexity will be $O(\log N)$. For uniform FMM, the depth of the oct-tree will be $O(\log N)$ and therefore with sufficient resources, the work at each level can be completed within one time unit, yielding the complexity result $O(\log N)$. For non-uniform particle distribution, *exponential node growth* condition must be satisfied in order to achieve the $O(\log N)$ complexity result.

Theorem 3.1. *In order to have $O(\log N)$ complexity for parallel adaptive FMM, it is necessary and sufficient for the data distribution to meet the exponential node growth condition: $\exists\ \alpha$, $1 < \alpha \leq 2^d$, such that*

$$\#\ nodes(l_{max} - l + 1) \geq \alpha \cdot \#\ nodes(l_{max} - l),\ 1 \leq l \leq l_{max}, \tag{3.1}$$

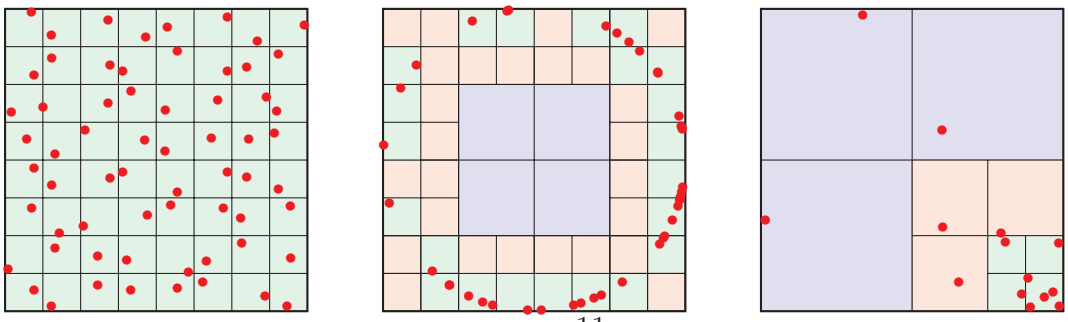
except for $\bar{l}$ levels. In (3.1), $\bar{l}$ is dependent of N, d is the dimensionality of the problem, and l_{max} is the maximum refinement level of the oct-tree.

Proof. *First, it is obvious that at any level, the number of non-empty boxes is at most N.*

Second, there exists an l^* such that the condition in 3.1 holds for all $l^* \leq l \leq l_{max}$. We further assume that there are T non-empty boxes at level l^*. Then, recursively, at level l where $l \geq l^*$, there are at least $T\alpha^{l-l^*}$ non-empty boxes.

The upper and lower bounds make the depth of the tree $O(\log N)$.

We consider several concrete examples as shown in Figure 3.1. In the figure, three levels of refinement have been carried out for each case and the corresponding values of α are computed. In Figure 3.1 (a), the particle sampling is uniform, and $\alpha = 4$ reaches the upper bound. As a result, the depth of the tree is the shortest. In Figure 3.1 (b), α value is in the range of $(1, 4)$ and therefore the depth of the tree will be $O(\log N)$. However, since α does not reach the upper bound, the depth will be longer than that in case (a). In Figure 3.1 (c), $\alpha = 1$ reaches the lower bounds. In this worst case, we can expect the depth of the tree to be $O(N)$. A second examination suggests that the usual definition for the computational domain is not suitable in this case. It should be considered as two clusters of particles in order to reduce the length of the critical path. More precisely, suppose that starting from level l^*, the value of α falls into $(1, 2^d]$ for all the levels $l \geq l^*$, then all particles contained in the box to be refined at level l^* should be considered as one cluster while the rest particles form the other cluster.



(a) $\alpha = 4$ (b) $\alpha = \frac{11}{6}$ (c) $\alpha = 1$

Figure 3.1: Exponential node growth rate

3.2.2 Maximum Concurrency under Dependency Constraints

In this section, we analyze the maximum concurrency potential of FMM subject to dependence constraints under PRAM architecture according to Bernstein's conditions [7]. In this process, we assume that data distribution satisfies the exponential node growth condition.

Given two program fragments P_i and P_j, Bernstein's conditions describe when the two are independent and can be executed in parallel. For P_i, let I_i be all the input variables and O_i the output variables, and likewise for P_j. P_i and P_j are independent if they satisfy

$$I_j \cap O_i = \emptyset \tag{3.2}$$

$$I_i \cap O_j = \emptyset \tag{3.3}$$

$$O_i \cap O_j = \emptyset \tag{3.4}$$

Violation of the first condition introduces a flow dependency, corresponding to the first statement producing a result used by the second statement. The second condition represents an anti-dependency, when the first statement overwrites a variable needed by the second expression. The third and final condition represents an output dependency: when two statements write to the same location, the final result must come from the logically last executed statement.

According to Bernstein's condition, we examine the dependence of the operators involved in each stage of FMM and produce the maximum concurrency graph as depicted in Figure 3.2. A detailed explanation is as follows.

First, far-field and near-field operations can start simultaneously from the beginning of the program since they have independent output sets (write to different memory locations). This means, generating multipole expansion at level l_{max} (far-field approximation) and processing list 1 for all childless boxes at each particle location (near-field direct interaction) are executed in parallel once the program begins.

Under PRAM architecture, there exists another operation that can start from the beginning

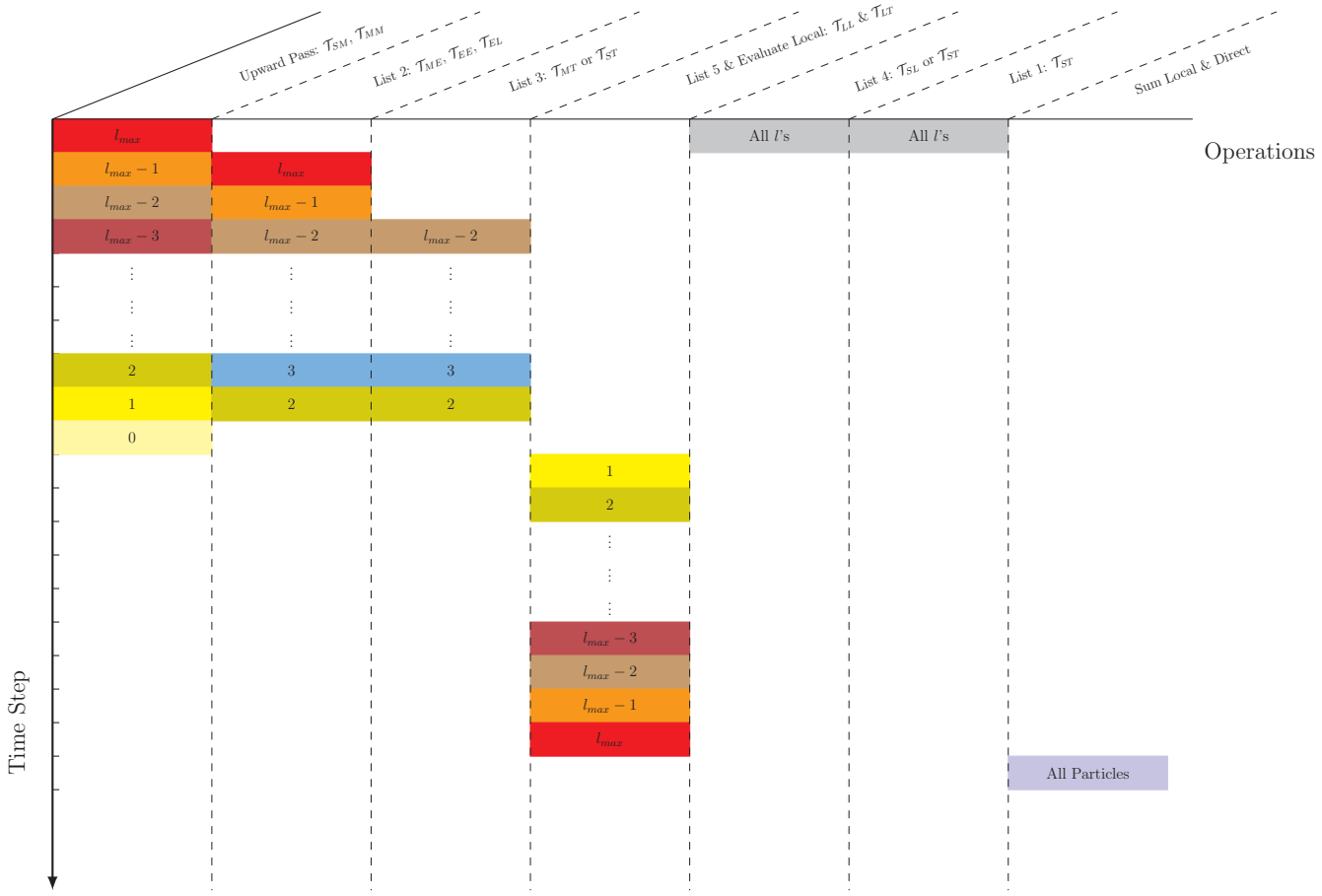


Figure 3.2: Potential concurrency under dependence constraints

of the program, namely, processing list 4. There exist two options to process list 4, either by generating local expansion from particle information using $\mathcal{T}_{SL}$ operator or by direct interaction using $\mathcal{T}_{ST}$ operator. The latter choice can induce potential writing conflicts with the operations related to list 1. However, under the PRAM assumption, it can be avoided by providing separate storages for the outputs. Doing so can also improve the accuracy of the numerical results. Recall when adding a set of numbers that differ by several orders of magnitude, it is more accurate to add the smaller values prior to the larger ones. Analogously, the results obtained from processing list 1 are larger in magnitude than those obtain from processing list 4 since the particles are further apart in the latter case. Using separate storages, we can calculate the contribution from each group accurately, hence yielding a better numerical result.

For boxes at level l, we only need their multipole expansions to process lists 2 and 3, which means these operations can start once the multipole expansions become available. Similar to processing list 4, there also exist two options to process list 3, either by direct interaction using $\mathcal{T}_{ST}$ operator or by evaluating multipole expansion using $\mathcal{T}_{MT}$ operator. Under PRAM assumption, the direct interaction operation will not induce any writing conflict. More precisely, with sufficient resources, generating multipole expansion at level l_{max}, and processing lists 1 and 4 can be completed in one time unit. Since there exists no box with a non-empty list 3 until level $l_{max} - 2$ (after two time units), there will be no writing conflict due to applying $\mathcal{T}_{ST}$ operator.

As the multipole expansion of a parent box depends on those of its children, the remaining part of the upward pass is processed sequentially, one level per time unit. Analogously, we process list 5 (shifting local expansion) and evaluate local expansion for particles in childless boxes one level per time unit.

In the last step, the results corresponding for near-field (direct interaction) and far-field (local expansion evaluation) are added together.

3.2.3 Architecture Constraints

In this section, we analyze the effects of architecture constraints on the parallelization of FMM by mainly considering two factors, i.e., limited computing resources and non-uniform memory access latency.

The limited computing resources have two effects. First, the length of the critical path of the algorithm will be elongated since certain task has to be split into more than one stages to complete. Second, separate storage is no longer the solution to resolve potential writing conflict due to insufficient resources. As a result, programmer must design spatial and temporal partition to introduce mutual exclusion to obtain correct result.

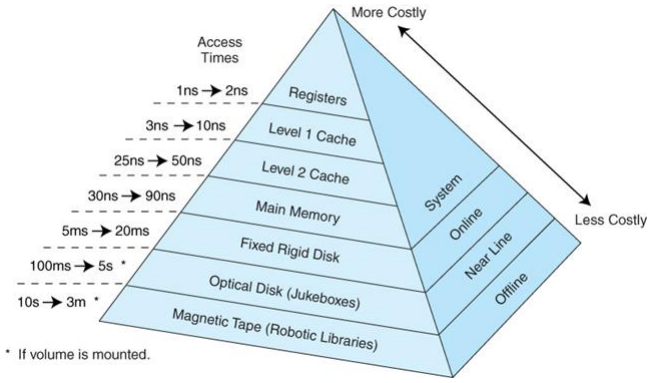


Figure 3.3: Memory hierarchy diagram

Non-uniform memory access latency is a more subtle issue, and has a huge impact on the performance of an application. Unlike the uniform memory access assumption for PRAM, a modern memory system is a hierarchy of storage devices with different capacities, costs, and access times, as shown in Figure 3.3 [1]. CPU registers hold the most frequently used data. Small, fast cache memories nearby the CPU act as staging areas for a subset of the data and instructions stored in the relatively slow main memory. The main memory stages data stored on large, slow disks, which in turn often serve as staging areas for data stored on the disks or tapes of other machines connected by networks. If the data requested by the program are stored in a CPU register, then they can be accessed in zero cycles during the execution of the instruction. If stored in a cache, it will take 1 to 30 cycles. If stored in main memory, the time

[1] http://tjliu.myweb.hinet/

becomes 50 to 200 cycles. In the situation the storage is in the disk, the access will require tens of millions of cycles. Consequently, it is crucial to understand how the system moves data up and down the memory hierarchy and write the application accordingly so that its data items are stored higher in the hierarchy where the CPU can access them more quickly.

To make a program efficient in memory access, one needs to consider three distances, i.e., *geometric distance*, *graph distance*, and *architecture distance*. We consider two objects in a physical system. The Euclidean distance between the two objects is said to be the geometric distance, which measures the distance in the actual physical system. For any problem, one needs to employ certain algorithm to solve it which introduces certain data structure. The logical distance in the data structure between the two objects is referred to as graph distance. Finally, when the algorithm is implemented on a computer, the architecture of the machine also affects the locations where different objects are stored. Clearly, closeness in geometric distance does not necessarily mean closeness in graph distance or closeness in architecture distance. Special case should be taken in the implementation of the algorithm to reduce memory access latency and improve the overall performance.

3.3 A Graph-Theoretic Parallelization Approach

Formally, the parallelization of FMM on the multicore architecture can be described as the following constrained minimization problem.

$$\begin{aligned} &\min_{\text{parallel scheduling}} \text{FMM\_time}(\{\text{data}\}, \{\text{translation-operators}\}) \\ &\text{subject to} \\ &\qquad \text{the temporal dependency constraints} \\ &\qquad \text{the limited degree of parallel processing} \\ &\qquad \text{the non-uniform latency in memory access} \end{aligned} \tag{3.5}$$

Currently, we are unaware of any $O(\log N)$ algorithm yielding the optimal solution to this

optimization problem. Therefore, in this section, we develop a graph-theoretic parallelization scheme trying to minimize the gap between the optimal solution and the practical solution. The key ideas in the parallelization strategy can be summarized as *spatio-temporal partition* and *split and merge scheduling*. They stem from the crucial observations that the workload of the adaptive FMM is not evenly distributed among all levels, and the majority is in the lower level of the tree. As a result, at a lower level, we should divide the level-specific work according to a partition of the tree, and assign one sub-domain to each available thread. At a higher level, the level-specific work should be merged with other tasks. For instance, the multipole-to-local translation at a given level l might not start as early as indicated in Figure 3.2. However, it does not need to follow the traditional approach which processes it in the downward pass. In other words, once the program reaches a certain upper level of the tree in the upward pass and have available resources to process it, the task can be executed. However, we emphasize that tasks on the critical path should always have the highest priority to finish first. Note that with only limited parallel computing resources, we need to design spatial partition to ensure mutual exclusion in writing in order to reduce synchronization overhead. As threads within the same process share system resources, we should also utilize the use of threads and thread management to reduce the synchronization scope as well.

3.3.1 Parallelization Scheme for Upward Pass

In the upward pass, the FMM sweeps the oct-tree from the finest level l_{max} to root level to generate multipole expansion for each box. In this process, the algorithm computes the multipole for a childless box from sources using $\mathcal{T}_{SM}$ operator and shift and merge the multipole of each child of a parent box using $\mathcal{T}_{MM}$ operator. The spatial data dependence further imposes a temporal constraint, i.e., $\mathcal{T}_{MM}$ operator cannot be applied on a parent box until it has been applied on each of its child boxes.

Due to the limited computing resources, spatial and temporal constraints, the upward pass is further divided into two stages. More precisely, this means that there exists a level in the oct-tree above which (including itself) the parallelization falls into the PRAM regime. Such a

level is referred to as *control level*. In other words, the critical path becomes elongated below the control level. A good parallelization scheme must be capable of mitigating this effect and hence we require the scheme to satisfy two criteria: (a) it should utilize as many threads as available to the program; and (b) it should distribute the workload in a way such that the number of threads being idle due to the aforementioned spatial and temporal constraints is minimal for a period as long as possible.

The accomplishment of the objectives is built on the following observations. First, in the adaptive version of FMM, not all childless boxes are on the finest level l_{max} and therefore the process need not necessarily start from level l_{max}. Secondly, there is no prior knowledge of the distribution of boxes at every level, and the traditional level-wise implementation is very likely to introduce artificial temporal constraints in parallelization.

In the first stage, a private thread is assigned to each box at the control level, whose task is to compute multipole expansions for the assigned box as well as all its descendants. We designed a recursive algorithm such that each thread could complete its work independently without following the level-wise restriction. As preparation, we set up counters for all parent boxes with initial value zero, which are used to record the number of times that $\mathcal{T}_{MM}$ operator has been applied on their children, respectively. Next, each thread starts from its assigned box, descends the tree, and stops until it reaches the first childless box. At this box, the thread will compute its multipole expansion from source information using $\mathcal{T}_{SM}$ operator followed by applying $\mathcal{T}_{MM}$ operator to shift the result to its parent. As all the operations associated with this box have been completed, the thread traverses up the tree to its parent box, updates the counter value and further compares it with the number of children of the parent box. If the results do not match, the thread descends the tree to the next unvisited child box. If this box is childless, the procedure just described is carried out, otherwise the algorithm is recursively invoked. If the results match, then the parent box has received all required information from its children and the thread will use $\mathcal{T}_{MM}$ operator to shift the multipole expansion to its own parent. For each thread, the first stage work is complete once the counter value of its initially assigned box equal to the number of its children. As an example, we consider the 1D adaptive tree shown in Figure 3.4. The control level is set to be level two and four threads are dispatched.

For thread 1, it will compute the multipole expansion for the boxes in the order of 16, 17, 8, 18, 28, 29, 19, 9, and 4.

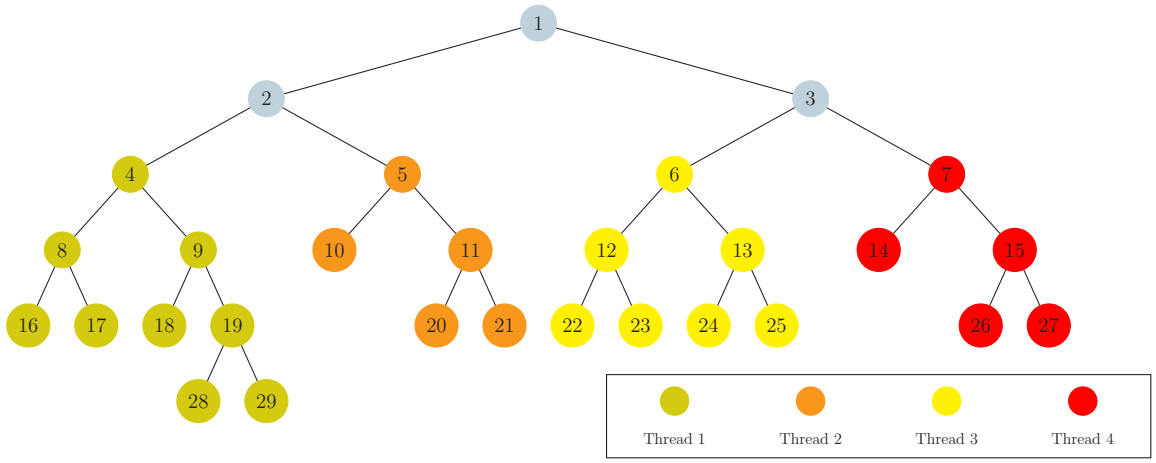


Figure 3.4: Upward pass parallelization scheme

In the second stage, each thread first shifts the multipole expansion of its initially assigned box to its own parent. After that, it attempts to shift the multipole expansion of its parent to its grandparent. However, this operation only succeeds if the aforementioned temporal constraint is satisfied, otherwise, the thread will exit and free system resources. We also notice that in this process, different threads may simultaneously update the counter value of a parent box. Therefore, we assign a mutex variable for each box above the control level. Each thread will first secure the mutex and then perform the operation. As an example, we consider the situation depicted in Figure 3.4. Both threads 1 and 2 will attempt to shift the multipole expansion of box 2 to box 1 and the work is executed by the thread that completes its first stage task later.

This recursive algorithm can effectively mitigates the elongation of the critical path below the control level and ultimately improve the parallelization efficiency, due to the following reasons. First, compared with the traditional level-wise approach, each box is visited the same times and the operation counts at each box remain unchanged too. Second, this algorithm is capable of keeping as many threads as possible working independently on their first stage tasks which contributes the majority workload of the upward pass.

3.3.2 Parallelization Scheme for Downward Pass

There are eight operators involved in the downward pass: $\mathcal{T}_{ME}$, $\mathcal{T}_{EE}$, and $\mathcal{T}_{EL}$ operators for processing the interaction list, $\mathcal{T}_{LL}$ operator for processing list 5, $\mathcal{T}_{SL}$, $\mathcal{T}_{MT}$, and $\mathcal{T}_{ST}$ operators for processing lists 1, 3, and 4, and $\mathcal{T}_{LT}$ operator for evaluating local expansions. According to their spatio-temporal properties (the input and output sets of the operator), these eight operators are further sorted into three categories and the parallelization scheme has three separate components accordingly. Operators $\mathcal{T}_{ME}$ and $\mathcal{T}_{EE}$ make up the first category; $\mathcal{T}_{LL}$ operator is in the second category; the rest operators belong to the last category. In the remaining of this section, we discuss the parallelization scheme for each category in more details.

There exist two options to process the interaction list. In particular, for each box being processed, it can considered either as a target box or a source box. As a target box, the algorithm converts the multipole expansion of each member of its interaction list into an exponential expansion and then into a local expansion about the target box's center. As the interaction list can have up to 189 members, the operation counts will be $6p^3 + 189p^2$. As a source box, the algorithm first converts the multipole expansion of the source box into an exponential expansion which is then sent to all the boxes that having the source box in their interaction lists. The operation counts in this manner can be reduced to $6p^3 + 40p^2$ by means of merge-and-shift technique, thanks to the fact that interaction lists of colleague boxes usually overlap. From the parallelization perspective, the first option itself inherits mutual exclusion while the other one has potential writing conflicts. In this thesis, we adopt the sender's view (consider each box as a source box) as its complexity is much smaller. Another reason is that the serial code available to the author is implemented with the merge-and-shift technique.

With the sender's option and merge-and-shift technique, it is crucial to introduce spatio-temporal partition to ensure mutual exclusion. We proceed as follows. Refinement level one is ignored since all boxes at this level have empty interaction lists. At refinement level two, no exclusion could be introduced since there are too few boxes to do so. However, we can either switch back to the receiver's option (consider each box as a target box) or simply executed the corresponding workload sequentially. Either choice will not affect the overall performance very

much. Mutual exclusion is introduced starting from refinement level three. As preparation, we associate a triplet of indices with each box, representing the order of this box in each direction assuming it were on a uniform mesh of its refinement level. These triplets are computed during the generation of the oct-tree structure. Since interaction lists have direction dependency due to the use of exponential expansions, the related workload is processed in the same manner. Particularly, in processing each direction, each thread should be assigned to work on a set of boxes whose corresponding index components in this direction form a set with at least four consecutive integers that does not overlap with the sets of other threads. The workload for each thread is complete in three stages.

We use an concrete example as shown in Figure 3.5 to illustrate the three-stage strategy. In this example, we are at refinement level three and the largest index in any direction is 8. As a result, two threads could be dispatched. Considering the y-direction, one thread will work on the boxes of the lower four rows while the other one will work on the rest. In the first stage, thread one will process boxes at rows two and three while thread two handles boxes at rows six and seven. In this process, boxes at rows four and five serve as the buffer zone as they are the upper and lower bound in box indices that will receive information from the boxes being processed. In the second stage, the first thread works on boxes at row one and the second thread processes boxes at row five. Lastly, boxes at rows four and eight are processed by each thread. Since threads may complete their assigned task at each stage at different paces, synchronization among threads is required at the end of each stage to ensure mutual exclusion. In other words, the operation of the next stage cannot start until all threads complete the task of current stage.

We would like to comment here that in the process of the interaction list in a given direction, the three-stage principle could be applied in the rest directions as well to achieve further acceleration. We consider the situation in Figure 3.5 and assume the algorithm is currently processing the y-direction using two threads, then the workload in each stage assigned to either thread could be completed in parallel by invoking two additional threads in the x-direction.

$\mathcal{T}_{LL}$ operator applies on parent boxes and it has a simple spatio-temporal constraint. Spatially, $\mathcal{T}_{LL}$ operator takes the local expansion of a parent box as input and generate local expansions of its children as output. There exists no writing conflicts when the operator is

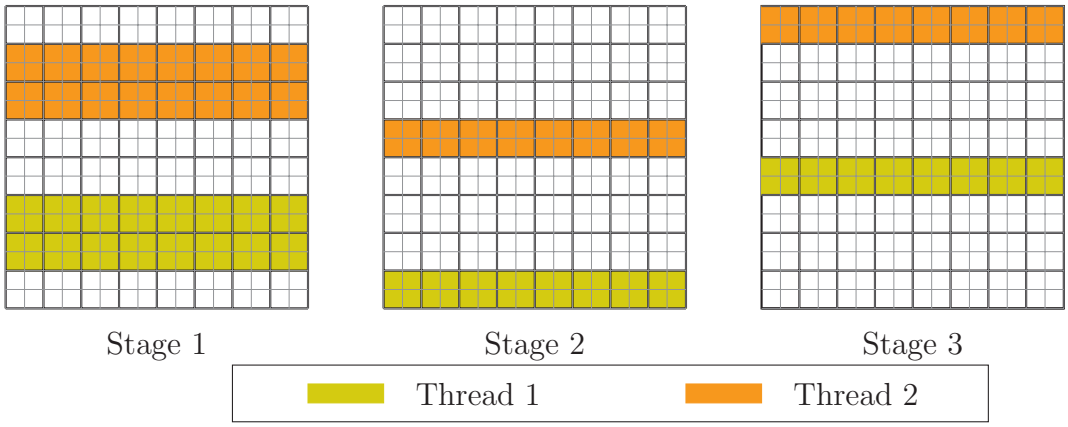


Figure 3.5: Parallelization scheme for $\mathcal{T}_{ME}$ & $\mathcal{T}_{EE}$ operators

applied on distinct parent boxes. Temporally, $\mathcal{T}_{LL}$ operator cannot be applied on a parent box unless the operator has already been applied to its own parent. As a result, the parallelization of $\mathcal{T}_{LL}$ operator is performed level-wise with more threads invoked at levels with more boxes.

The remaining operators are applied on childless boxes or boxes with non-empty list 4. For each operator, it can be applied on distinct boxes simultaneously since the output sets are independent. To achieve better load balance and therefore better parallelization efficiency, the workload corresponding to each operator is completed following a multiple-server-single-queue model. Specifically, we proceed as follows. We first allocate two arrays during the generation of the oct-tree structure to record the IDs of childless boxes and boxes with non-empty list 4. In other words, we first generate the "queue". We simulate multiple servers by the maximum number of threads available to the program. Initially, we assign one box to each thread and switch its state from idle to busy. For any thread, it will pick up the first box in the remaining queue once it finishes working on its previously assigned box. In this way, a thread will keep busy until it sees an empty queue and it will exit and free system resources. Notice in this process, multiple threads may check or update the current state of the workload queue simultaneously, and therefore we define a mutex variable to ensure the correct result.

3.4 Numerical Experiments

In this section, we demonstrate the efficiency of the parallelization scheme through some numerical experiments carried out on a Sun SunFire^{TM} X4600 server with 8 AMD Opteron^{TM} processor 885 at 2.6 GHz clock rate and 64 GB memory. The parallel implementation is based on and later compared with the serial code FMM-Yukawa, which computes the pairwise screened Coulomb interactions of N charges. The results are summarized in Tables 3.2, 3.4, and 3.5. In each table, timings are measured in second; the Binning column corresponds to the time consumed on the adaptive tree generation which is done sequentially in all tested cases. There are two points to remember in interpreting these times. One is that they were taken on a time sharing system; even though no other users were present, various daemons will consume some resources. The second is the effect of the choice of the maximum number of particles s allowed in a childless box. For problem size of the tested scale, up to 100 million particles, it is impractical to run the program with multiple choices of s for each N since it would require too much exclusive use of the server without producing much useful results. Based on previous studies (see Tables 2.1 – 2.4), we made an educated guess that $s = 80$ and used it for all the tests.

In the first set of tests, we study the relationship between the CPU time requirement and the number of threads available to the program. The result is given in Table 3.2. This experiment uses 10 million charges uniformly distributed inside the unit box $[-0.5, 0.5]^3$ and requires six digits accuracy. The data corresponding to one thread is obtained by running the sequential code FMM-Yukawa to avoid any thread management overhead. Recall that in different stages of FMM, our parallelization scheme has corresponding strategies. As a result, we further breaks down the timing results into five components, as shown in columns three to seven in Table 3.2, to see the efficiency of individual parts. The results in Table 3.2 is also plotted in Figure 3.6.

Several observations can be made from Table 3.2 as follows.

Firstly, we obtain nearly optimal 16 times speedup on the 16 core system in processing "Lists 1, 3, 4" and "Evaluate Local". This is mainly due to the mutual exclusion property intrinsic to the involved operators.

Secondly, the speedup for processing "Upward" and "List 5" is approximately 20% less than the optimal speedup, due to the temporal constraints imposed by the traversal of the oct-tree in those processes.

Thirdly, the speedup for "List 2" is the least. The parallel execution time is only 4 times faster on the 16 cores computing environment. Therefore, we take the $\mathcal{T}_{ME}$ operation in the up and down direction as an example to examine its causes by tracking workload, number of threads available, number of threads used, and execution time of each level. The results are given in Table 3.3. From this table, we can observe that we achieve the expected speedup at levels 3 and 4, a less satisfying result at levels 5 and 6, and no speedup at level 7. At level 7, there exists great parallelization potential in theory. However, the actual workload is extremely less than the theoretical estimate. As the program has no prior knowledge of this, we suffer significantly from the thread management overhead at this level. At levels 5 and 6, the current manually controlled three-stage strategy also seems to introduce too much synchronization overhead. Notice that with the merge-and-shift technique, the parallelization of $\mathcal{T}_{ME}$ operation is analogous to the parallelization of matrix-vector multiplication according to a column decomposition manner. Without this technique, the parallelization is analogous to a row decomposition scheme. The latter has intrinsic mutual exclusion while the former has fewer sequential operation counts. In this case, the operation counts of the former option is roughly a quarter of the latter. On the current implementation platform, if we did not apply the merge-and-shift technique and assume the speedup is optimal, then the expected execution time will roughly be the same as the current version with the merge-and-shift technique. We believe that a better result in terms of both parallelization efficiency and actual execution time should come from automatic tuning. This topic is currently under investigation and results will be reported in the future.

Fourthly, there are two phases of reduction in the program execution time. For instance, see the column for "Evaluate Local", from 1 thread to 16 threads, the execution time is approximately halved when the number of threads is doubled, due to the increase of cores invoked to execute the program. After this, the reduction is no longer linear. However, we achieve another roughly 20% improvement by increasing the number of threads from 16 to 128. This

is the result of balancing between the cost in swapping memory and switching among different threads. In practice, the code can hardly be optimal in the sense that all the data demanded by a piece of instructions is right in the register when the code is being executed. Therefore, the sequential code is associated with certain cost in swapping the memory hierarchy to get the desired data into the register. On the other hand, in a threaded code, when the desired data for one thread is unavailable, the program can invoke another thread to work while loading the data into the register. This choice is also associated with certain cost penalizing the switch between threads. Subsequently, when the number of threads increases in a reasonable range, it can somehow hide the memory reference latency and improve the overall performance.

In the next two sets of tests, we further examine the performance of the parallelization scheme on larger scale system with two types of particle distribution, namely, charges uniformly distributed inside the unit box and uniformly distributed on the surface of the sphere centered at the origin with radius 1. For each particle distribution, the computation is performed for both three and six digits accuracy. The maximum number of threads allowed is set to be 128. Although this choice may lead to less satisfying result in processing list 2, we realize that this operation only takes a small fraction of the sequential operation and using more threads indeed can increase parallelization efficiency of other time-consuming parts significantly. In the table, timing results are separated into the binning time and the actual algorithm executed time. TS corresponds to the sequential execution time of the algorithm obtained by FMM-Yukawa while TP refers to the parallel execution time of the algorithm with 128 threads. Subscripts are used to indicate the accuracy of the calculation. R_i is the ratio of TS_i to TP_i for $i = 3, 6$. The results in Tables 3.4 and 3.5 are also plotted as bar graphs in Figures 3.7 and 3.8. In each figure, parallel timing results are plotted on top of the sequential timing results.

# of Threads	Binning	Upward	List 2	Lists 1,3,4	List 5	Evaluate Local
1	26.719	72.462	132.089	1415.301	14.423	149.134
2	26.447	71.797	78.902	663.570	10.707	75.078
4	26.480	71.653	48.825	334.450	3.914	37.597
8	28.399	10.413	39.288	169.900	2.797	18.714
16	27.449	10.321	34.001	102.380	1.431	11.577
32	26.517	6.958	33.733	90.448	1.206	9.929
64	28.363	7.082	33.389	87.254	1.236	9.586
128	26.481	5.712	34.140	85.838	1.142	9.413
Speedup	—	12.686	3.869	15.333	12.630	15.843

Table 3.2: CPU execution time VS number of threads

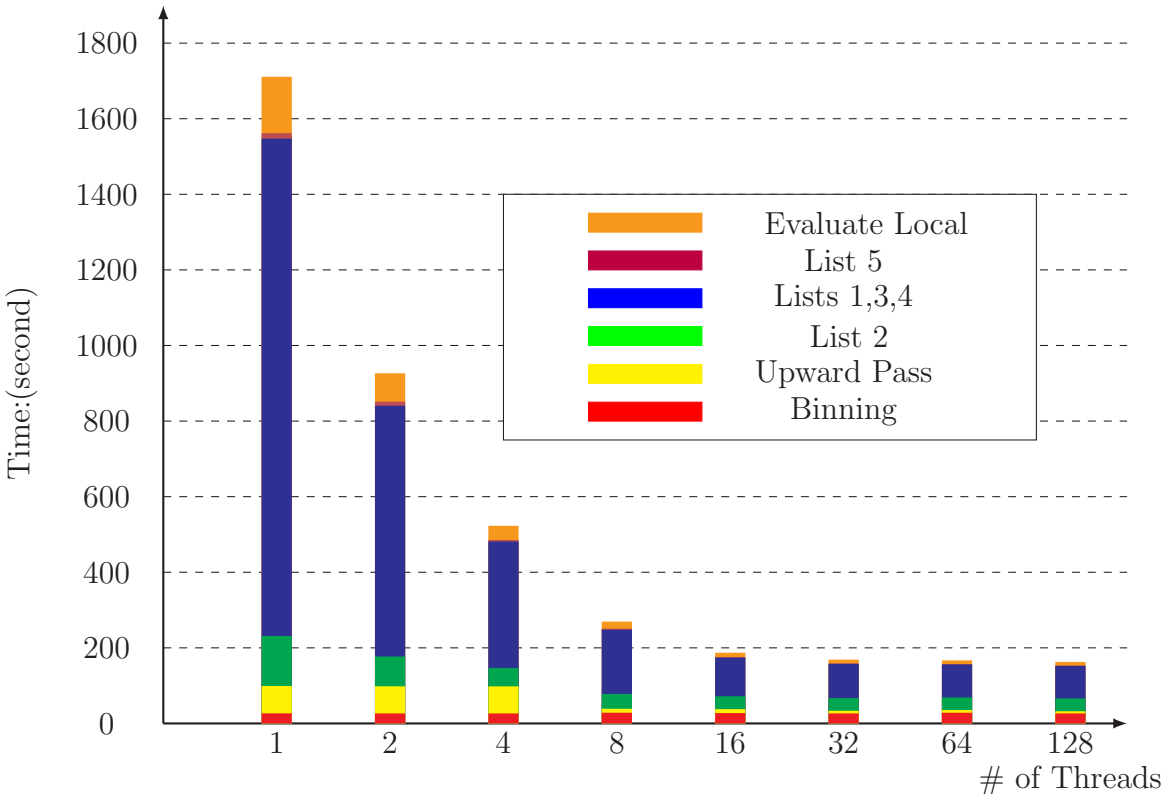


Figure 3.6: CPU execution time VS number of threads

worklev	(time, usedthread)							box2proc
0	(1,0.001)	(1,0.001)	(1,0.001)	(1,0.001)	(1,0.001)	(1,0.001)	(1,0.001)	8
1	(1,0.002)	(1,0.002)	(1,0.002)	(1,0.002)	(1,0.002)	(1,0.003)	(1,0.003)	32
2	(1,0.011)	(1,0.011)	(1,0.011)	(1,0.011)	(1,0.011)	(1,0.011)	(1,0.012)	128
3	(1,0.047)	(2,0.026)	(2,0.026)	(2,0.027)	(2,0.026)	(2,0.026)	(2,0.027)	512
4	(1,0.379)	(2,0.213)	(4,0.136)	(4,0.128)	(4,0.133)	(4,0.135)	(4,0.148)	4096
5	(1,3.257)	(2,1.820)	(4,1.154)	(8,0.759)	(8,0.771)	(8,0.789)	(8,0.751)	32768
6	(1,22.205)	(2,12.392)	(4,8.149)	(8,5.659)	(16,4.980)	(16,5.065)	(16,4.949)	207096
7	(1,0.029)	(2,0.034)	(4,0.066)	(8,0.140)	(16,0.292)	(32,0.513)	(32,0.533)	8
max threads	1	2	4	8	16	32	128	

Table 3.3: Timing results for $\mathcal{T}_{ME}$ operation at every level in Up direction

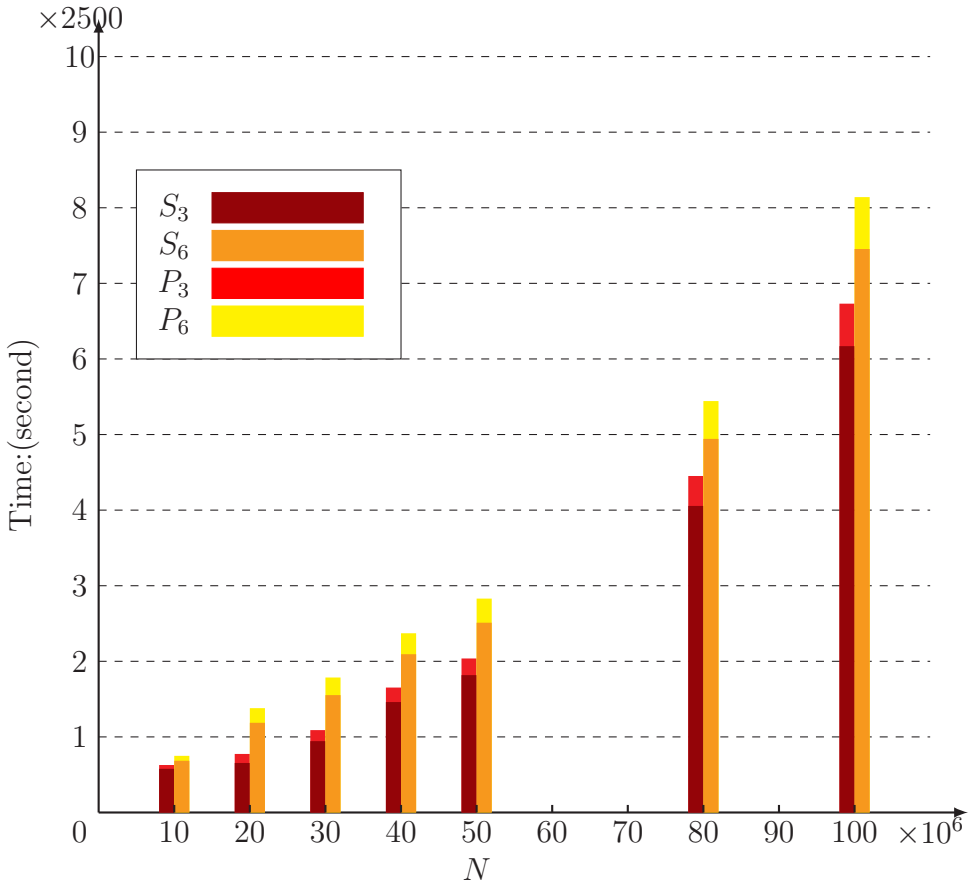


Figure 3.7: CPU execution time VS N for uniform sampling

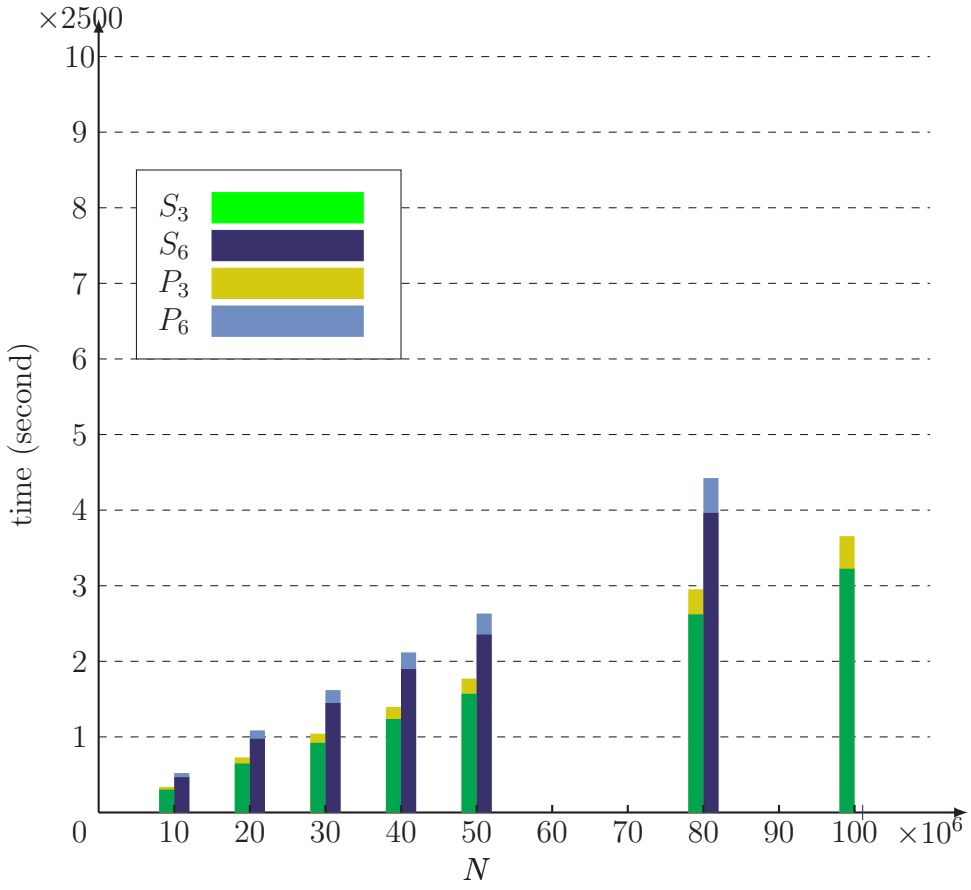


Figure 3.8: CPU execution time VS N for nonuniform sampling

$N(\times 10^6)$	Binning	TS_3	TP_3	R_3	TS_6	TP_6	R_6
10	27.080	1417.030	99.262	14.276	1683.867	135.646	12.413
20	141.950	1495.680	156.720	9.544	2824.748	341.893	8.262
30	167.791	2192.223	196.940	11.132	3714.574	414.153	8.969
40	226.606	3426.167	252.650	13.561	5008.203	466.325	10.740
50	253.935	4289.247	296.730	14.455	6022.582	543.600	11.079
80	268.883	9869.811	721.550	13.679	12086.537	982.520	12.302
100	308.770	15114.427	1096.305	13.787	18327.415	1407.343	13.022

Table 3.4: CPU execution time VS N for uniform sampling

$N(\times 10^6)$	Binning	TS_3	TP_3	R_3	TS_6	TP_6	R_6
10	21.610	737.450	67.196	14.276	1146.906	114.373	12.413
20	50.363	1571.514	149.141	9.544	2387.632	226.289	8.262
30	83.772	2220.290	218.043	11.132	3536.901	343.598	8.969
40	113.897	2977.343	284.198	13.561	4630.944	435.962	10.740
50	144.193	3783.726	356.934	14.455	5741.257	550.277	11.079
80	248.487	6305.099	579.618	13.679	9660.471	903.581	12.302
100	332.580	7730.302	743.888	13.787	—	—	—

Table 3.5: CPU execution time VS N for nonuniform sampling

Chapter 4

A Kernel Independent Fourier-series-based Fast Multipole Method

In this chapter we introduce a new kernel-independent Fourier-series-based FMM-like algorithm. Although the algorithm is applicable to a wide class of kernels, for expository purpose, we illustrate the key idea and establish the theory for kernels with the so-called *scaling property*. A kernel $K(r, \theta, \phi)$ given in the spherical coordinates is said to have scaling property if there exists a $\gamma \in \mathcal{R}$ such that

$$K(Cr, \theta, \phi) = C^{\gamma} K(r, \theta, \phi). \tag{4.1}$$

Kernels belonging to this category include Green's functions for Laplace equation in three dimensions, modified Laplace equation, Stokes equation, biharmonic equation, dislocation dynamics in material science, to name a few. For such kernels, we seek a truncated Fourier-series-based expansion in the form of

$$\sum_{j,k,l} \omega_{j,k,l} e^{i\alpha(j,k,l)(x,y,z)^T}, \quad \alpha \in \mathcal{R}. \tag{4.2}$$

Seeking the kernel expansion in this particular form has several benefits. First, once we fix the basis function of the expansion, it only requires the ability of evaluating kernel function to obtain the expansion coefficient. Therefore, the framework of FMM can be extended to many kernels whose expansions are either explicitly unavailable or are too expensive to calculate.

Secondly, it is shown in later sections that the multipole-to-local translation becomes diagonal. Thirdly, compared with the plane-wave expansion introduced in the new-version FMM, this expansion is valid in all directions, and therefore the multipole-to-local translation can be completed in only one loop. Fourthly, it allows all the translation operators constructed in a receiver-driven manner, which means the operators have intrinsic mutual exclusion property and great parallelization potential on modern multicore computers.

The organization of this chapter is outlined as follows. In Section 4.1, we give a brief review of previous developments in the kernel-independent FMM. In Section 4.2, we discuss how to find the truncated Fourier series expansion of the kernel function. In Section 4.3, we construct all the translation operators used in the new algorithm. In Section 4.4, we provide a pseudo-code to illustrate the algorithm structure and discuss several strategies to further reduce the algorithmic complexity. In Section 4.5, we present several numerical results.

4.1 Precursors of Kernel-Independent FMM

In this section, we give a brief overview of the development in the kernel-independent FMM in the last two decades. There are three building blocks in FMM, consisting of tree structure, kernel expansion, and translation operators. For all the approaches to be discussed, they don't change the tree structure. They are characterized by the basis used for kernel expansion and the corresponding translation operators.

A straightforward approach is to use Taylor expansions in the Cartesian coordinates, for instance, see [62, 58]. However, this approach is not suitable for high accuracy computation since it requires $O(p^d)$ expansion for pth-order accuracy which is quite expensive.

In a series of papers from Beylkin et al. [11, 12, 10, 8], the authors approximate the power function $r^{-\alpha}$ with a linear combination of Gaussians, where $\alpha > 0$. Doing so allows operations in d dimensions to use combinations of one-dimensional operations and therefore achieves computational complexity that formally scales linearly in d.

Gimbutas and Rokhlin derived a modification of FMM in 2D applicable to non-oscillatory kernels [28]. In their scheme, the Taylor and Laurent expansions are replaced with tensor

products of Legendre expansions which are subsequently compressed using singular value decomposition (SVD). The advantage of this technique is that using SVD guarantees an optimal compression in the sense of L^2 form, hence the number of terms in the multipole expansion is minimal for a given approximation error.

Another approach is based on using equivalent sources, which was first proposed by Anderson as he represented the far-field as the solution to an exterior Dirichlet problem on a ball surrounding the particles by means of the exact Green's function for the Laplacian [4]. However, Anderson's method requires the analytical form of the Green's function for each kernel, which may not be explicitly available in general. A successful implementation of the idea to general kernels comes from Ying et al. [75]. Their scheme only requires the existence of the Green's function and relies on kernel evaluation. The potential generated by sources inside a box is represented by a continuous distribution of an equivalent density on a surface enclosing the box, which is found by matching its potential to the potential of the original sources at a surface in the far-field. This scheme is applicable to second-order constant coefficient non-oscillatory elliptical partial differential equations.

In 2007, Martinsson and Rokhlin proposed a scheme based on the so-called "skeletonization" for 1D problems [51]. For two sets of particles, one as the source and the other one as target, this scheme approximates the interaction matrix by a low-rank matrix to within some precision, say rank k. They then choose a subset of k "proxy" sources to represent the source set and another subset of k target locations with the property that if the potential is known at these k points, it can be interpolated to all the remaining points.

A recent formulation of FMM for non-oscillatory kernels which are only known numerically was proposed by Fong and Darve [25]. This algorithm combines interpolation with SVD. Specifically, the far-field behavior of the kernel $K(x, y)$ is approximated by a Chebyshev interpolation scheme which results in a low-rank representation of the kernel. Then the multipole-to-local operator is to evaluate the field due to particles located at Chebyshev nodes, which is done using an SVD.

We mention that there exists another category of kernel-independent approach used in solving integral equations of the second kind. It is based on wavelet decomposition, combined

with a Galerkin scheme [9, 3]. The idea is to replace the exact Galerkin matrix through a matrix that is nearly sparse by setting all entries below a certain threshold equal to zero. Once the sparse form of the matrix is obtained, applying it to an arbitrary vector is an $O(N)$ procedure.

4.2 Kernel Approximation

In this section, we discuss how to find the truncated Fourier series expansion in the form of (4.2) for the kernel function. For expository purposes, we restrict our attention in 2D throughout the remaining of the chapter.

We first determine in which region the approximation should be accurate. This problem arises naturally due to the fact that kernels in general are not periodic themselves. Since the kernel approximation will be accurate in a finite region, we can only expect the multipole expansion to be valid in a finite region. Therefore, the region where kernel approximation is accurate should be determined in a way to guarantee that the multipole expansion of a box is accurate whenever it is needed. In the language of FMM, this means that the kernel approximation should guarantee that the multipole expansion of a box is accurate in its interaction list region since it will be translated into a local expansion in the downward pass. Consequently, in Figure 4.1 (a), we need the multipole expansion of box b accurate in the shaded green area. By scaling property, we can further assume without loss of generality that the length of the side of box b is 1. Then, for any particle contained in b, the distance between the particle and any box in the shaded green area is at least 1 and at most 4 in each direction. If the approximation of the kernel is accurate in the shaded region depicted in Figure 4.1 (b), then the contribution of each particle in b can be approximated accurately in the shaded green region in Figure 4.1 (a), and we can obtain a set of multipole coefficients of box b that are accurate in the boxes having b in their interaction lists, respectively. For the simplicity of the discussion, we introduce notation $\Omega^d_{a,b}$ to represent the region $[-a,a]^d \setminus [-b,b]^d$ where d is the dimensionality of the region. The approximation region for the kernel function will then be symbolically denoted by $\Omega^2_{1,4}$.

Given a kernel function $K(x,y)$, there exist two approaches to obtain a truncated Fourier series approximation accurate in $\Omega^2_{1,4}$. On one hand, we can directly extend $K(x,y)$ into a

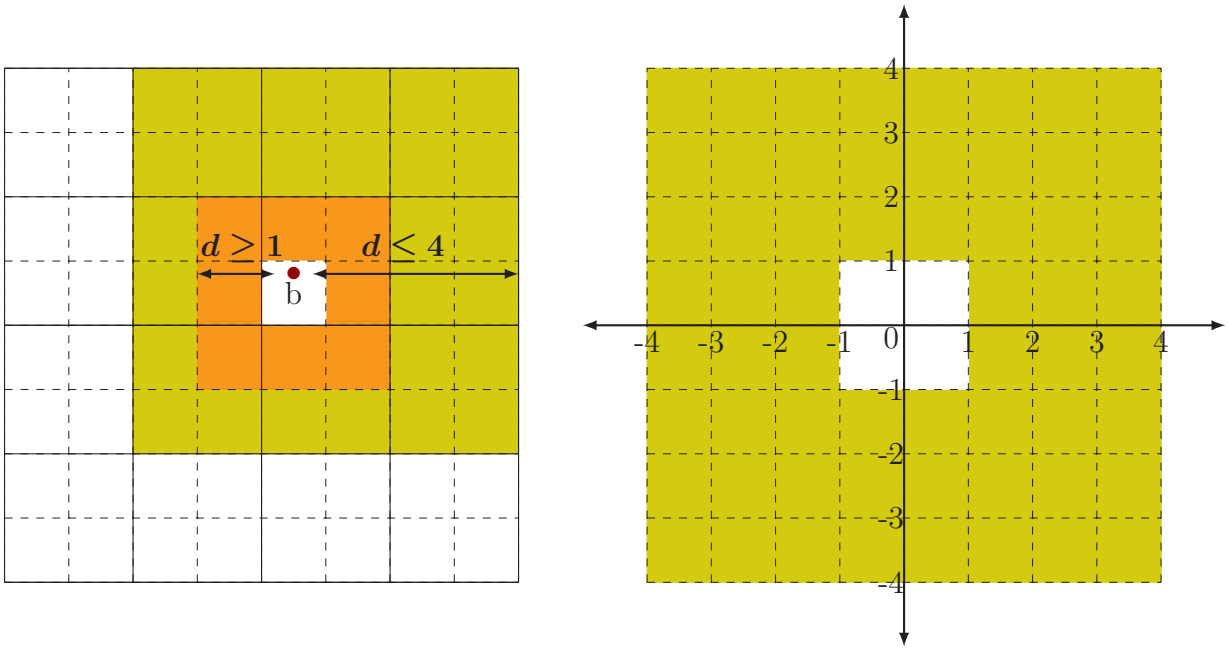


Figure 4.1: Kernel approximation region

periodic function with period 8. On the other hand, we can assume that in $\Omega^2_{1,4}$, $K(x,y)$ coincides with some unknown function $\tilde{K}(x,y)$ whose period L is larger than 8 and use the Fourier series expansion of $\tilde{K}(x,y)$. For the former choice, we realize that on the boundary of $\Omega^2_{1,4}$, the extended function might have discontinuity if $K(x,y)$ is an odd function or might only be C^1 if $K(x,y)$ is an even function. Consequently, the truncated series will converge to $K(x,y)$ very slowly which yields a large value of p. Such difficulty can be overcome by the latter choice since we are free to tweak $\tilde{K}(x,y)$ outside $\Omega^2_{1,4}$ such that it is sufficiently smooth and only a few terms will be needed to satisfy certain accuracy requirement. The existence of such function $\tilde{K}(x,y)$ can be established through the consideration of the Taylor series of $K(x,y)$ and use of appropriate smooth windowing functions around the boundary of $\Omega^2_{1,4}$.

In practice, we only need the existence of $\tilde{K}(x,y)$. We don't first construct $\tilde{K}(x,y)$ and then compute its Fourier coefficients to determine values for $\{\omega_{j,k}\}$. Instead, those coefficients are determined by P possibly non-equispaced sampling points in $\Omega^2_{1,4}$, say (x_p, y_p) with function value $K(x_p, y_p)$, $p = 1, \ldots, P$. Function $\tilde{K}(x,y)$ is determined from the coefficients $\{\omega_{j,k}\}$

implicitly.

Regarding the value of P, it can be either equal to or greater than the number of unknowns $(2p+1)^2$ to be determined. However, when $P = (2p+1)^2$, several numerical experiments indicate that regardless of the locations of the sampling points, the computed coefficients tend to differ by more than 15 orders of magnitude when high accuracy is required in the kernel approximation. This means, in double precision computing environment, the accuracy is contaminated the moment those coefficients are used. Due to this reason, we prefer the latter option, which we refer to as oversampling configuration.

Selecting the optimal sampling points remains an open question and is currently under investigation. In this section, we provide several rules of thumbs. The key idea can be summarized as *adaptive sampling*. In other words, we iteratively modify the distribution of the sampling points until the approximation error satisfies the accuracy requirement. In this process, we use two classes of sampling points, namely, uniform nodes and Chebyshev nodes. We will first use a simple 1D problem to illustrate the effect of using different nodes and then describe the algorithm in more details.

Let $f(x) = \dfrac{1}{x^2}$ be the kernel function and we seek an approximation in the form of $\sum_{-p\leq j\leq p} \omega_j e^{ij\alpha x}$ where $x \in \Omega^1_{1,4}$. We define the ratio of the number of sampling points to the number of unknowns as oversampling ratio. We fix $p = 10$ and $\alpha = \dfrac{2\pi}{9}$ and change the value of oversampling ratio. The numerical results are summarized in Table 4.1. From this table, we observe that: (a) with the same oversampling ratio, using Chebyshev nodes yields smaller maximum error and (b) the maximum error for Chebyshev nodes is less sensitive to the oversampling ratio than that of the uniform nodes. Therefore, we shall first use Chebyshev nodes to collect the overall behavior of the kernel function and then add uniform nodes in necessary regions in later iterations to meet the accuracy requirement.

Remark 4.1. *In Table 4.1 and throughout this chapter, the maximum errors are computed by comparison of the approximations and exact functions at sufficiently large numbers of points to insure complete resolution of all function features and accurate estimates of maximum errors.*

Formally, this process is described in the following algorithm:

Oversampling ratio	Uniform nodes	Chebyshev nodes
2	$9.5593 \cdot 10^{-4}$	$4.5897 \cdot 10^{-4}$
4	$6.5505 \cdot 10^{-4}$	$4.5897 \cdot 10^{-4}$
8	$6.5472 \cdot 10^{-4}$	$4.5897 \cdot 10^{-4}$

Table 4.1: Maximum approximation error for $\frac{1}{x^2}$: $p = 10$ and $\alpha = \frac{2\pi}{9}$

Adaptive Sampling Algorithm

Initialization:

Choose precision requirement ϵ and i_{max}.

for $p = p_1, p_2, \ldots$

for $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_{max}$

Set sampling points to be Chebyshev nodes, obtain approximation $\{\omega_{j,k}\}$.

for $i = 1, 2, \ldots, i_{max}$

Compute maximum approximation error.

if maximum error $< \epsilon$

Return p, α, and the corresponding $\{\omega_{j,k}\}$.

else

Identify region with larger approximation error.

Add uniform nodes to this region to obtain new approximation

endif

end

end

end

Remark 4.2. *The termination condition α_{max} for the α loop in the adaptive sampling algorithm is the following. At every iteration, we construct the function $\tilde{K}(x, y)$ from the computed coefficients $\{\omega_{j,k}\}$ and examine its graph in the region $[-L, L] \times [-L, L]$ where $L = \frac{2\pi}{\alpha}$. As indicated by numerical experiments, the coefficients tend to differ by large orders when the graph has concavity change in the examination region. Therefore, we terminate the α loop once such phenomenon is observed.*

4.3 Translation Operators

In this section, we construct translation operators for the Fourier-series-based kernel-independent FMM. The following theorems and lemma constitute the principal analytical tools of this chapter.

We first describe how to generate the multipole expansion from source information.

Theorem 4.1 (Multipole Expansion ($\mathcal{T}_{SM}$ operator))**.** *Suppose that M charges of strengths $q_1, q_2, \ldots, q_M$ are located at points $(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)$, all of which are contained inside a box A of length one and centered at (x_A, y_A). (x, y) is an arbitrary point belonging to the interaction list region of A. For any given $\epsilon > 0$, there exists a pth order multipole expansion which approximates the potential $\phi(x, y)$ due to all the charges inside A with a relative precision ϵ with respect to the potential induced by the total charge.*

Proof. *As the length of the side of A is one, the distance between any particle contained in A and any point belonging to the interaction list region of A is at least one and at most four in each direction. For the given $\epsilon > 0$, using the method discussed in Section 4.2, there exists an $\alpha \in (0, \frac{\pi}{4})$ and $p \in \mathcal{N}$ such that*

$$K(x - x_m, y - y_m)(1 + \theta_m) = \sum_{-p \le j,k \le p} \omega_{j,k}^{A} e^{i\alpha(j,k)(x - x_m, y - y_m)^T}, \quad 1 \le m \le M \tag{4.3}$$

where $(x - x_m, y - y_m) \in \Omega_{1,4}^2$ and $|\theta_m| < \epsilon$.

Define the multipole expansion of A as

$$\sum_{1 \le m \le M} \sum_{-p \le j,k \le p} \omega_{j,k}^{A} e^{i\alpha(j,k)(x - x_m, y - y_m)^T}.$$

Then,

$$\begin{aligned}
&\left| \sum_{1\le m\le M} q_m K(x-x_m, y-y_m) - \sum_{1\le m\le M} \sum_{-p\le j,k\le p} \omega^A_{j,k} e^{i\alpha(j,k)(x-x_m,y-y_m)^T} \right| \\
\le &\sum_{1\le m\le M} |q_m| \cdot \left| K(x-x_m, y-y_m) - \sum_{-p\le j,k\le p} \omega^A_{j,k} e^{i\alpha(j,k)(x-x_m,y-y_m)^T} \right| \\
= &\sum_{1\le m\le M} |q_m||\theta_m| K(x-x_m, y-y_m) \\
\le &\epsilon \sum_{1\le m\le M} |q_m| K(x-x_m, y-y_m)
\end{aligned}$$

The multipole expansion of A obtains a relative precision ϵ with respect to the potential due to the total charge in A.

Furthermore,

$$\begin{aligned}
&\sum_{1\le m\le M} q_m \sum_{-p\le j,k\le p} \omega^A_{j,k} e^{i\alpha(j,k)(x-x_m,y-y_m)^T} \\
= &\sum_{-p\le j,k\le p} \omega^A_{j,k} \left[\sum_{1\le m\le M} q_m e^{i\alpha(j,k)(x_A-x_m,y_A-y_m)^T} \right] e^{i\alpha(j,k)(x-x_A,y-y_A)^T} \\
\stackrel{def}{=} &\sum_{-p\le j,k\le p} \omega^A_{j,k} M^A_{j,k} e^{i\alpha(j,k)(x-x_A,y-y_A)^T} \qquad (4.4)
\end{aligned}$$

In the remaining discussion, we refer to Equation (4.4) as the multipole expansion about the center of box A and $\{\omega^A_{j,k} M^A_{j,k}\}$ as the coefficients of the multipole expansion. □

The construction of multipole to multipole ($\mathcal{T}_{MM}$) and local to local ($\mathcal{T}_{LL}$) operators requires the following lemma.

Lemma 4.1 (Half Period Expansion)**.** *Given $\alpha \in (0, \frac{\pi}{4})$ and $x \in [-1, 1]$, for any $\delta > 0$, there exists an $H > 0$ such that*

$$\left| e^{i\frac{\alpha}{2}x} - \sum_{-H\le h\le H} \tau_h e^{i\alpha h x} \right| < \delta \qquad (4.5)$$

Proof. *To start with, we observe that*

$$\cos\frac{\alpha}{2}x = \sum_{0\le n\le\infty}\frac{(-1)^n}{(2n)!}\left(\frac{\alpha}{2}\right)^{2n}x^{2n}$$

$$\sum_{0\le h\le H}C_h\cos h\alpha x = \sum_{0\le n\le\infty}\frac{(-1)^n}{(2n)!}\left(\sum_{0\le h\le H}C_h h^{2n}\right)\alpha^{2n}x^{2n}$$

There are totally $H+1$ unknowns of C_h and therefore we set

$$\sum_{0\le h\le H}C_h h^{2n} = \frac{1}{2^{2n}},\quad n=0,\ldots,H \tag{4.6}$$

Analogously, we have

$$\sin\frac{\alpha}{2}x = \sum_{0\le n\le\infty}\frac{(-1)^n}{(2n+1)!}\left(\frac{\alpha}{2}\right)^{2n+1}x^{2n+1}$$

$$\sum_{1\le h\le H}S_h\sin h\alpha x = \sum_{0\le n\le\infty}\frac{(-1)^n}{(2n+1)!}\left(\sum_{1\le h\le H}S_h h^{2n+1}\right)\alpha^{2n+1}x^{2n+1}.$$

We set

$$\sum_{1\le h\le H}S_h h^{2n+1} = \frac{1}{2^{2n+1}},\quad n=0,\ldots,H-1 \tag{4.7}$$

Using Euler formula, we have

$$\tau_0 = C_0,\ \tau_h = \frac{C_h+S_h}{2},\ \tau_{-h} = \frac{C_h-S_h}{2},\ 1\le h\le H \tag{4.8}$$

The truncation errors for various values of H in infinity norm corresponding to $\alpha=\frac{\pi}{4}$ are summarized in Table 4.2. □

We now describe how to shift the multipole expansion of a child box to the center of its parent.

Theorem 4.2 (Translation of a Multipole Expansion ($\mathcal{T}_{MM}$ operator))**.** *Suppose that (4.4) is the multipole expansion of box A of length one and centered at (x_A, y_A), which contains M*

H	Error
4	3.07405×10^{-4}
6	1.00136×10^{-6}
8	4.00287×10^{-7}
10	1.76879×10^{-8}
12	8.35399×10^{-10}
14	4.12392×10^{-11}
16	2.10243×10^{-12}

Table 4.2: Half period expansion error

charges of strengths $q_1, q_2, \ldots, q_M$ *located at points* $(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)$. *Box* B *of length two and centered at* (x_B, y_B) *is the parent box of* A. *There exists a linear transformation* $\mathcal{T}_{MM} : M^A_{j,k} \to M^B_{j,k}$ *such that for any point* (x, y) *belonging to the interaction list region of* B, *the potential* $\phi(x, y)$ *due to all charges contained in* A *is approximated by the multipole expansion*

$$\sum_{-p \le j,k \le p} \omega^B_{j,k} M^B_{j,k} e^{i\beta(j,k)(x-x_B, y-y_B)^T}, \quad \omega^B_{j,k} = 2^\gamma \omega^A_{j,k}, \;\; \beta = \frac{\alpha}{2} \tag{4.9}$$

with the same relative precision ϵ *defined in Theorem 4.1 with respect to the potential due to the total charge contained in* A.

Proof. *Applying the scaling property of the kernel to (4.3), we can derive*

$$\omega^B_{j,k} = 2^\gamma \omega^A_{j,k} \quad \textit{and} \quad \beta = \frac{\alpha}{2}$$

Analogously, we can generate a multipole expansion about the center of box B *due to the charges inside* A *using Theorem 4.1 and then obtain the claimed accuracy result.*

Now, we concentrate on constructing the linear transformation operator $\mathcal{T}_{MM}$.

First, by Lemma 4.1, we select an $H > 0$ *such that the difference between* $e^{i\frac{\alpha}{2}x}$ *and* $\sum_{-H \le h \le H} \tau_h e^{i\alpha x}$ *is smaller than* ϵ *for* $\alpha \in (0, \frac{\pi}{4})$ *and* $x \in [-1, 1]$ *(assume the machine precision is* ϵ*).*

Next, the $\mathcal{T}_{MM}$ *are constructed in three cases.*

Case I: j and k are both even. Specifically, $j = 2j_1$ and $k = 2k_1$.

$$\begin{aligned} M^B_{j,k} &= \sum_{1\le m\le M} q_m e^{i\beta(j,k)(x_B-x_m,y_B-y_m)^T} \\ &= e^{i\alpha(j_1,k_1)(x_B-x_A,y_B-y_A)^T} \sum_{1\le m\le M} q_m e^{i\alpha(j_1,k_1)(x_A-x_m,y_A-y_m)^T} \\ &= e^{i\alpha(j_1,k_1)(x_B-x_A,y_B-y_A)^T} M^A_{j_1,k_1} \end{aligned} \tag{4.10}$$

Case II: j is odd and k is even. Specifically, $j = 2j_1 + 1$ and $k = 2k_1$.

$$\begin{aligned} M^B_{j,k} &= \sum_{1\le m\le M} q_m e^{i\beta(j,k)(x_B-x_m,y_B-y_m)^T} \\ &= \sum_{1\le m\le M} q_m e^{i\alpha(j_1,k_1)(x_B-x_m,y_B-y_m)^T} e^{i\frac{\alpha}{2}(x_B-x_m)} \\ &= \sum_{1\le m\le M} q_m e^{i\alpha(j_1,k_1)(x_B-x_m,y_B-y_m)^T} \sum_{-H\le h\le H} \tau_h e^{i\alpha h(x_B-x_m)} \\ &= \sum_{-H\le h\le H} \tau_h \left[\sum_{1\le m\le M} q_m e^{i\alpha(j_1+h,k_1)(x_A-x_m,y_A-y_m)^T} \right] e^{i\alpha(j_1+h,k_1)(x_B-x_A,y_B-y_A)^T} \\ &= \sum_{-H\le h\le H} \tau_h M^A_{j_1+h,k_1} e^{i\alpha(j_1+h,k_1)(x_B-x_A,y_B-y_A)^T} \end{aligned} \tag{4.11}$$

Analogously, when $j = 2j_1$ and $k = 2k_1 + 1$, we have

$$M^B_{j,k} = \sum_{-H\le h\le H} \tau_h M^A_{j_1,k_1+h} e^{i\alpha(j_1,k_1+h)(x_B-x_A,y_B-y_A)^T} \tag{4.12}$$

Case III: j and k are both odd. Specifically, $j = 2j_1 + 1$ and $k = 2k_1 + 1$.

$$\begin{aligned}
M_{j,k}^B &= \sum_{1\le m\le M} q_m e^{i\beta(j,k)(x_B-x_m,y_B-y_m)^T} \\
&= \sum_{1\le m\le M} q_m e^{i\alpha(j_1,k_1)(x_B-x_m,y_B-y_m)^T} e^{i\frac{\alpha}{2}(x_B-x_m)} e^{i\frac{\alpha}{2}(y_B-y_m)} \\
&= \sum_{1\le m\le M} q_m e^{i\alpha(j_1,k_1)(x_B-x_m,y_B-y_m)^T} \sum_{-H\le h_1\le H} \tau_{h_1} e^{i\alpha h_1(x_B-x_m)} \sum_{-H\le h_2\le H} \tau_{h_2} e^{i\alpha h_2(y_B-y_m)} \\
&= \sum_{-H\le h_1,h_2\le H} \tau_{h_1}\tau_{h_2} \left[\sum_{1\le m\le M} q_m e^{i\alpha(j_1+h_1,k_1+h_2)(x_A-x_m,y_A-y_m)^T} \right] e^{i\alpha(j_1+h_1,k_1+h_2)(x_B-x_A,y_B-y_A)^T} \\
&= \sum_{-H\le h_1,h_2\le H} \tau_{h_1}\tau_{h_2} M_{j_1+h_1,k_1+h_2}^A e^{i\alpha(j_1+h_1,k_1+h_2)(x_B-x_A,y_B-y_A)^T} \qquad (4.13)
\end{aligned}$$

Lastly, we mention that in Equations (4.11) – (4.13), $j_1 + h$ and $k_1 + h$ range from $-H - \frac{p}{2}$ to $H + \frac{p}{2}$ respectively, assuming p is an even number. When we have $p \ge \frac{H}{2}$, all the multipole $\{M_{j,k}^A\}$ used in the formulas have been computed and the formulas are well defined. On the other hand, when $H > \frac{p}{2}$, some values of $\{M_{j,k}^A\}$ are missing. In this case, we let $\tilde{p} = 2H$ and associate each box with a $\tilde{p}^2$-term multipole expansion. For box A, we compute $\{M_{j,k}^A\}$, where $-\tilde{p} \le j, k \le \tilde{p}$. For the newly added $\{\omega_{j,k}\}$, their values are set to be zero. In this way, we obtain well defined formulas while maintaining the same precision result. □

Remark 4.3. *In Theorem 4.2, if the given $\alpha \in (0, \frac{\pi}{4})$ satisfies*

$$\left| e^{i\alpha x} - \sum_{-H\le h\le H} \tau_h e^{i2\alpha h x} \right| < \epsilon, \quad x \in [-1, 1],$$

then case III can be done in a simpler manner as follows:

$$\begin{aligned}
M_{j,k}^B &= \sum_{1\le m\le M} q_m e^{i\beta(j,k)(x_B-x_m,y_B-y_m)^T} \\
&= \sum_{1\le m\le M} q_m e^{i\alpha(j_1,k_1)(x_B-x_m,y_B-y_m)^T} e^{i\alpha\frac{x_B-x_m+y_B-y_m}{2}} \\
&= \sum_{1\le m\le M} q_m e^{i\alpha(j_1,k_1)(x_B-x_m,y_B-y_m)^T} \sum_{-H\le h\le H} \tau_h e^{i2\alpha h\frac{x_B-x_m+y_B-y_m}{2}} \\
&= \sum_{-H\le h\le H} \tau_h \left[\sum_{1\le m\le M} q_m e^{i\alpha(j_1+h,k_1+h)(x_A-x_m,y_A-y_m)^T}\right] e^{i\alpha(j_1+h,k_1+h)(x_B-x_A,y_B-y_A)^T} \\
&= \sum_{-H\le h\le H} \tau_h M_{j_1+h,k_1+h}^A e^{i\alpha(j_1+h,k_1+h)(x_B-x_A,y_B-y_A)^T}
\end{aligned}$$

We now describe how to translate the multipole expansion of box B into a local expansion of box C which has B in its interaction list.

Theorem 4.3 (Conversion of a Multipole Expansion into a Local Expansion ($\mathcal{T}_{ML}$ operator))**.** *Suppose that the multipole expansion of box B is given by (4.9). Box B belongs to the interaction list of box C centered at (x_C, y_C). Without loss of accuracy, the multipole expansion (4.9) can be converted into a local expansion about the center of box C.*

Proof. *The local expansion can be obtained by shifting the expansion center as follows.*

$$\begin{aligned}
&\sum_{-p\le j,k\le p} \omega_{j,k}^B M_{j,k}^B e^{i\beta(j,k)(x-x_B,y-y_B)^T} \\
&= \sum_{-p\le j,k\le p} \left[\omega_{j,k}^B M_{j,k}^B e^{i\beta(j,k)(x_C-x_B,y_C-y_B)^T}\right] e^{i\beta(j,k)(x-x_C,y-y_C)^T} \\
&\stackrel{def}{=} \sum_{-p\le j,k\le p} L_{j,k}^C e^{i\beta(j,k)(x-x_C,y-y_C)^T} \qquad (4.14)
\end{aligned}$$

In the remaining discussion, we refer to Equation (4.14) as the local expansion about the center of box C and $\{L_{j,k}^C\}$ as the coefficients of the local expansion. The operator mapping $\{M_{j,k}^B\}$ to $\{L_{j,k}^C\}$ is referred to as the $\mathcal{T}_{ML}$ operator. □

Lastly, we describe how to shift the local expansion of box C to its child box D.

Theorem 4.4 (Translation of a Local Expansion ($\mathcal{T}_{LL}$ operator)). *Suppose that box C centered at (x_C, y_C) with side length two has a local expansion in the form of (4.14). Box D centered at (x_D, y_D) is a child box of C. There exists a linear transformation $\mathcal{T}_{LL} : L^C_{j,k} \to L^D_{j,k}$ which converts the local expansion of C into a local expansion about the center of D.*

Proof. *$\mathcal{T}_{LL}$ operator is constructed by processing each term in the local expansion of C. In the proof, we assume that p is even and $p \geq 2H$ where H is defined in Theorem 4.2.*

The local expansion about the center of D is in the form of

$$\sum_{-p \leq j,k \leq p} L^D_{j,k} e^{i\alpha(j,k)(x-x_D, y-y_D)^T}, \quad \alpha = 2\beta.$$

First, we set all $\{L^D_{j,k}\}$ to be zero and then process each term in the local expansion of C, matching it to those of the local expansion of D.

Case I: j and k are even. Specifically, $j = 2j_1$ and $k = 2k_1$.

$$\begin{aligned} & L^C_{j,k} e^{i\beta(j,k)(x-x_C, y-y_C)^T} \\ =& L^C_{j,k} e^{i\alpha(j_1,k_1)(x_D-x_C, y_D-y_C)^T} e^{i\alpha(j_1,k_1)(x-x_D, y-y_D)^T} \end{aligned}$$

Therefore,

$$L^D_{j_1,k_1} \longleftarrow L^D_{j_1,k_1} + L^C_{j,k} e^{i\alpha(j_1,k_1)(x_D-x_C, y_D-y_C)^T} \tag{4.15}$$

Case II: j is odd and k is even. Specifically, $j = 2j_1 + 1$ and $k = 2k_1$.

$$
\begin{aligned}
& L^C_{j,k} e^{i\beta(j,k)(x-x_C,y-y_C)^T} \\
=& L^C_{j,k} e^{i\alpha(j_1,k_1)(x-x_C,y-y_C)^T} e^{i\frac{\alpha}{2}(x-x_C)} \\
=& L^C_{j,k} e^{i\alpha(j_1,k_1)(x-x_C,y-y_C)^T} \sum_{-H\le h\le H} \tau_h e^{i\alpha h(x-x_C)} \\
=& \sum_{-H\le h\le H} \tau_h L^C_{j,k} e^{i\alpha(j_1+h,k_1)(x_D-x_C,y_D-y_C)^T} e^{i\alpha(j_1+h,k_1)(x-x_D,y-y_D)^T}
\end{aligned}
$$

Therefore, for each h, $-H \le h \le H$,

$$
L^D_{j_1+h,k_1} \longleftarrow L^D_{j_1+h,k_1} + \tau_h L^C_{j,k} e^{i\alpha(j_1+h,k_1)(x_D-x_C,y_D-y_C)^T} \tag{4.16}
$$

Analogously, when $j = 2j_1$ *and* $k = 2k_1 + 1$, *for each* h, $-H \le h \le H$,

$$
L^D_{j_1,k_1+h} \longleftarrow L^D_{j_1,k_1+h} + \tau_h L^C_{j,k} e^{i\alpha(j_1,k_1+h)(x_D-x_C,y_D-y_C)^T} \tag{4.17}
$$

Case III: j *and* k *are both odd. Specifically,* $j = 2j_1 + 1$ *and* $k = 2k_1 + 1$.

$$
\begin{aligned}
& L^C_{j,k} e^{i\beta(j,k)(x-x_C,y-y_C)^T} \\
=& L^C_{j,k} e^{i\alpha(j_1,k_1)(x-x_C,y-y_C)^T} e^{i\frac{\alpha}{2}(x-x_C)} e^{i\frac{\alpha}{2}(y-y_C)} \\
=& L^C_{j,k} e^{i\alpha(j_1,k_1)(x-x_C,y-y_C)^T} \sum_{-H\le h_1\le H} \tau_{h_1} e^{i\alpha h_1(x-x_C)} \sum_{-H\le h_2\le H} \tau_{h_2} e^{i\alpha h_2(y-y_C)} \\
=& \sum_{-H\le h_1,h_2\le H} \tau_{h_1}\tau_{h_2} L^C_{j,k} e^{i\alpha(j_1+h_1,k_1+h_2)(x_D-x_C,y_D-y_C)^T} e^{i\alpha(j_1+h_1,k_1+h_2)(x-x_D,y-y_D)^T}
\end{aligned}
$$

Therefore, for each (h_1, h_2), $-H \le h_1, h_2 \le H$,

$$
L^D_{j_1+h_1,k_1+h_2} \longleftarrow L^D_{j_1+h_1,k_1+h_2} + \tau_{h_1}\tau_{h_2} L^C_{j,k} e^{i\alpha(j_1+h_1,k_1+h_2)(x_D-x_C,y_D-y_C)^T} \tag{4.18}
$$

4.4 Algorithm Structure and Further Improvements

In this section, we present a pseudo-code to describe the algorithm structure. We start with the uniform version, and then discuss several strategies to reduce the algorithmic complexity, and conclude the section with the adaptive version of the algorithm.

Uniform FMM Algorithm

Initialization

Choose a level of refinement $l_{max} \approx \log_4 N$, a precision ϵ. Determine the number of terms p used in each direction in the kernel approximation.

Upward Pass

Step 1

for each box b at the finest refinement level l_{max}

Compute the factor $\{M_{j,k}^b\}$ of the multipole expansion using Theorem 4.1.

end

Step 2

for $l = l_{max} - 1, \ldots, 0$

for each box b at level l

Compute $\{M_{j,k}^b\}$ by merging expansions from its children via Theorem 4.2.

end

end

Downward Pass

Step 3

Set $\{L_{j,k}^b\} = (0, 0, \ldots, 0)$ for each box b at level 1.

for $l = 2, \ldots, l_{max} - 1$

for each box b at level l

Shift the local expansion of b's parent about b's center using Theorem 4.4.

end

for each box b at level l

Convert the multipole expansion of each box j in the interaction list of b into a local expansion about the center of b via Theorem 4.3.

end

end

Step 4

for each box b at level l_{max}

Convert the multipole expansion of each box j in the interaction list of b into a local expansion about the center of b via Theorem 4.3.

end

Step 5

for each box b at level l_{max}

Evaluate the local expansion of b for every particle located inside b.

end

Step 6

for each box b at level l_{max}

For every particle located inside b, compute interactions with all other particles within b and its nearest neighbors directly.

end

Step 7

for each box b at level l_{max}

For every particle inside b, add direct and far-field results together.

end

A brief analysis of the algorithmic complexity is given below.

Step	**Operation Count**	**Explanation**
1	order Np^2	Each particle contributes to one expansion at the finest level.
2	order Np^4	At the lth level, 4^l shifts involving order p^4 work per shift must be performed.
3	order $\leq Np^4 + 27Np^2$	The first loop requires order Np^4 work. There are at most 27 entries in the interaction list for each box at each level. Order p^2 work is required to convert one multipole into local expansion.
4	order $\leq 27Np^2$	There are at most 27 entries in the interaction list for each box, and $4^{l_{max}} \approx N$ boxes.
5	order Np^2	p^2 -term expansion is evaluated at each particle location.
6	order $\frac{9}{2}k_nN$	Let k_n be an bound on the number of particles per box at the finest level. Interactions must be computed within the box and its eight nearest neighbors. Newton's third law allows computing only half of the pairwise interactions.
7	order N	Adding two terms for each particle.

In the above algorithm, there have several places for further improvement. We now discuss the strategies to reduce the operation counts.

First, we will construct $\mathcal{T}_{LL}$ operator from a receiver-initiated manner rather than the current sender-driven fashion. From the parallelization perspective, the former choice allows computing all the coefficients of $\{L_{j,k}\}$ simultaneously. In other words, we process each term of the local expansion of a child box D concurrently by collecting information from appropriate terms of its parent C's local expansion.

Without loss of generality, we assume that p is an even number. The case when p is odd can be handled analogously. As shown in the proof of Theorem 4.4, we only need to process terms whose indices (j,k) satisfy the inequality $-\frac{p}{2}-H \le j,k \le \frac{p}{2}+H-1$. Given such a pair (j,k), $L^D_{j,k}$ is constructed as follows.

1. If $-\frac{p}{2} \le j,k \le \frac{p}{2}$, then $L^D_{j,k}$ receives from $L^C_{2j,2k}$

$$L^C_{2j,2k} e^{i\alpha(j,k)(x_D-x_C,y_D-y_C)^T} \tag{4.19}$$

2. If $-\frac{p}{2} \le j \le \frac{p}{2}$ and $-\frac{p}{2}-H \le k \le \frac{p}{2}+H-1$, then $L^D_{j,k}$ receives from $L^C_{2j,2k-2h+1}$

$$\begin{aligned} &\tau_h L^C_{2j,2k-2h+1} e^{i\alpha(j,k)(x_D-x_C,y_D-y_C)^T}, \quad h_{min} \le h \le h_{max} \\ &h_{min} = \max\{-H, k-\frac{p}{2}+1\}, \qquad h_{max} = \min\{H, k+\frac{p}{2}\} \end{aligned} \tag{4.20}$$

3. If $-\frac{p}{2}-H \le j \le \frac{p}{2}+H-1$ and $-\frac{p}{2} \le k \le \frac{p}{2}$, then $L^D_{j,k}$ receives from $L^C_{2j-2h+1,2k}$

$$\begin{aligned} &\tau_h L^C_{2j-2h+1,2k} e^{i\alpha(j,k)(x_D-x_C,y_D-y_C)^T}, \quad h_{min} \le h \le h_{max} \\ &h_{min} = \max\{-H, j-\frac{p}{2}+1\}, \qquad h_{max} = \min\{H, j+\frac{p}{2}\} \end{aligned} \tag{4.21}$$

4. If $-\frac{p}{2}-H \le j,k \le \frac{p}{2}+H-1$, then $L^D_{j,k}$ receives from $L^C_{2j-2h_1+1,2k-2h_2+1}$

$$\begin{aligned} &\tau_{h_1}\tau_{h_2} L^C_{2j-2h_1+1,2k-2h_2+1} e^{i\alpha(j,k)(x_D-x_C,y_D-y_C)^T} \\ &\max\{-H, j-\frac{p}{2}+1\} \le h_1 \le \min\{H, j+\frac{p}{2}\} \\ &\max\{-H, k-\frac{p}{2}+1\} \le h_2 \le \min\{H, k+\frac{p}{2}\} \end{aligned} \tag{4.22}$$

From (4.19) – (4.22) and (4.10) – (4.13), several improvements are available. First, we consider shifting the local expansion from one parent box C to its four child boxes. Notice that only the values of $\{e^{i\alpha(j,k)(x_D-x_C,y_D-y_C)^T}\}$ are changed between different child boxes. Therefore, we first compute the common factors only once and then do a component-wise multiplication

to obtain the result for each child box. Second, we consider two different parent boxes C_1 and C_2. Let D_1 and D_2 be their child boxes, respectively. If the relative position of C_1 and D_1 is the same as that of C_2 and D_2, then

$$e^{i\alpha(j,k)(x_{D_1}-x_{C_1},y_{D_1}-y_{C_1})^T} = e^{i\alpha(j,k)(x_{D_2}-x_{C_2},y_{D_2}-y_{C_2})^T}.$$

This suggests at any level l, we only need to compute four sets of $\{e^{i\alpha(j,k)(x_D-x_C,y_D-y_C)^T}\}$ for D being the lower left, lower right, upper left, and upper right child of C and store them as global variables which are used both in the upward pass and downward pass.

In certain circumstances, we can relax the accuracy requirement for the kernel approximation to reduce the value of p. For example, we can require

$$K(x,y) - \sum_{-p\le j,k\le p} \omega_{j,k} e^{i\alpha(j,k)(x,y)^T} = (1+\theta)K_{max}, \tag{4.23}$$

where $K_{max} = \|K(x,y)\|_\infty$, $(x,y) \in \Omega_{1,4}^2$, and $|\theta| < \epsilon$. Essentially, this leads to the best uniform approximation of the kernel within a scaling factor. Then we have

$$\begin{aligned}
&\left| \sum_{1\le m\le M} q_m K(x-x_m, y-y_m) - \sum_{1\le m\le M} q_m \sum_{-p\le j,k\le p} \omega_{j,k} e^{i\alpha(j,k)(x-x_m,y-y_m)^T} \right| \\
=&\left| \sum_{1\le m\le M} q_m K(x-x_m, y-y_m) - \sum_{1\le m\le M} q_m K(x-x_m, y-y_m) + \sum_{1\le m\le M} q_m \theta_m K_{max} \right| \\
\le&\epsilon \sum_{1\le m\le M} |q_m| K_{max}
\end{aligned} \tag{4.24}$$

The multipole expansion will achieve a relative precision ϵ with respect to the maximum potential induced by the total charge (assuming all particles carry positive charges and reside at the same location).

We now develop an adaptive version of the algorithm which needs some modification to the current data structure.

We first recall the adaptive algorithm described in Section 2.3. For each box b belonging to the list 3 of some box c, we can either directly compute the interactions or can evaluate the multipole expansion of b at every particle location in c. For each box b belonging to the list 4 of some box c, we can either directly compute the interactions or can generate a local expansion about c's center using the particle information contained in b.

Compared with that, we can only process the lists 3 and 4 of certain box in the current data structure by means of direct calculation. For example, we consider the situation depicted in Figure 4.2. Both boxes c_1 and c_2 belong to the list 3 of box b. However, only the multipole expansion of box c_2 is valid in the entire region covered by box b. As a result, we can only collect potential due to charges inside box c_1 by computing the interaction direction. When it comes to box c_2, we can alternatively evaluating its multipole expansion. Since the number of particles contained in boxes c_1 and c_2 are approximately both $4s$, the direct interaction with cost of $2s^2$ operations will become inefficient compared with $p^2 s$ operations of evaluating multipole expansion when $s > \frac{p^2}{2}$. From this example, we observe that for any pair of boxes b and c where c belongs to the list 3 of b, the multipole expansion of c is invalid in the entire region covered by b when b and c are more than one level apart in the tree. Therefore, in the following adaptive algorithm, we impose a constraint requiring that adjacent boxes (either sharing a vertex or an edge) cannot be more than one level apart in the tree. Under this condition, we can always process lists 3 and 4 using the more efficient approach available.

Following is a formal description of the algorithm. Description of the notation can be found in Section 2.3.

Adaptive FMM Algorithm

Initialization

Choose precision ϵ and then determine the number of terms used in each direction of the kernel approximation p. Choose the maximum number s of charges allowed in a childless box.

Build Tree Structure

Step 1

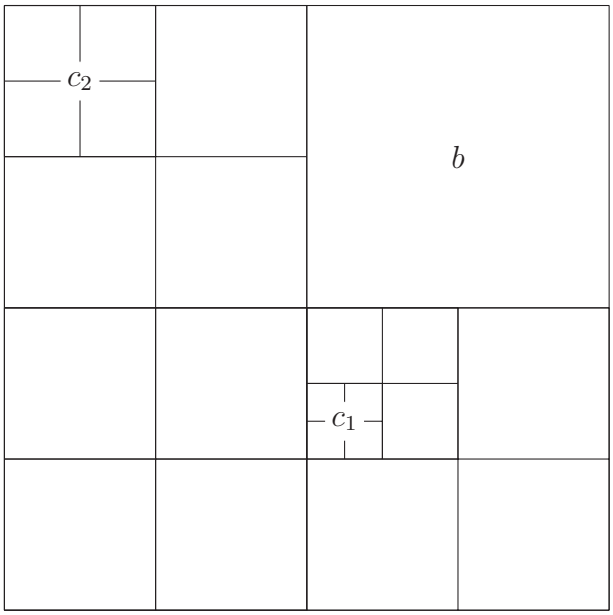


Figure 4.2: (b) and two entries c_1 and c_2 of b's list 3

for $l = 0, 1, 2, \ldots$

for each box $b \in B_l = \{$nonempty boxes at level $l\}$

if b contains more than s charges **then**

Divide b into four child boxes. Ignore empty children and

add the nonempty child boxes to B_{l+1}.

endif

end

end

Comment [Denote the maximum refinement level obtained by l_{max}]

for $l = l_{max}, \ldots, 0$

for each box b at level l

Refine each adjacent box c of b so that c and b are at most one level

apart in the tree. Update the nonempty box sets accordingly.

end

end

Comment [Denote the total number of boxes created by $nbox$.]

for each box b_i, $i = 1, 2, \ldots, nbox$

Create Lists $L_1(b_i)$, $L_2(b_i)$, $L_3(b_i)$, $L_4(b_i)$

end

Upward Pass

Step 2

for $l = l_{max}, \ldots, -1, 0$

for each box b in B_l

if b is childless **then**

Compute the factor $\{M^b_{j,k}\}$ by using Theorem 4.1.

else

Compute $\{M^b_{j,k}\}$ by merging expansions from its children via Theorem 4.2.

endif

end

end

Downward Pass

Step 3

for each box b_i, $i = 1, \ldots, nbox$

for each box $c \in L_4(b_i)$

if the number of charges in $b_i \leq p^2$ **then**

Comment [The number of charges in b_i is small. It is faster to use direct calculation than to generate the contribution to the local expansion $\{L^{b_i}_{j,k}\}$ due to charges in c; act accordingly.]

Calculate potential field at each particle point in b_i directly from charges in c.

else

Comment [The number of charges in b_i is large. It is faster to generate the contribution to the local expansion $\{L_{j,k}^{b_i}\}$ due to charges in c than to use direct calculation; act accordingly.]

Generate a local expansion at b_i's center due to charges in c.

endif

end

end

Step 4

for $l = 2, 3, \ldots, l_{max}$

for each box $b \in B_l$

Convert the multipole expansion of each box j in interaction list of b into a local expansion about the center of b via Theorem 4.3.

end

end

Step 5

for each box b_i, $i = 1, 2, \ldots, nbox$

if b_i is a parent box **then**

Shift the local expansion $\{L_{j,k}^{b_i}\}$ to the centers of its children using Theorem 4.4, and add the translated expansions to children's local expansions.

endif

end

Evaluation of Potentials

Step 6

for each box $b_i = 1, 2, \ldots, nbox$

if b_i is childless **then**

Calculate the potential at each charge in b_i from the local expansion $\{L^{b_i}_{j,k}\}$

endif

end

Step 7

for each box b_i, $i = 1, 2, \ldots, nbox$

if b_i is childless **then**

for each box $c \in L_3(b_i)$

if the number of charges in $c \leq p^2$ **then**

Comment [The number of charges in c is small. It is faster to use direct calculation than to evaluate the multipole expansion $\{\omega^c_{j,k} M^c_{j,k}\}$; act accordingly.]

Calculate the potential at each charge b_i directly from charges in c.

else

Comment [The number of charges in c is large. It is faster to evaluate the expansion $\{\omega^c_{j,k} M^c_{j,k}\}$ than to use direct calculation; act accordingly.]

Calculate the potential at each charge in b_i from multipole expansion $\{\omega^c_{j,k} M^c_{j,k}\}$

endif

end

endif

end

Step 8

for each box b_i, $i = 1, 2, \ldots, nbox$

if b_i is childless **then**

Calculate the potential at each charge in b_i directly due to all charges in $L_1(b_i)$.

endif

end

The analysis of the complexity of the adaptive algorithm uses the following results from [16].

1. There are at most $\log_2 \varepsilon$ levels of refinement where ε is the machine precision.
2. The number of nonempty boxes is at most $\frac{5N}{s} \cdot \log_2 \varepsilon$.
3. The number of boxes in all list 1's is bounded by $\frac{44N}{s} \cdot \log_2 \varepsilon$.
4. For any box, its list 2 has no more than 27 entries.
5. The total number of boxes in list 3 (or list 4) is bounded by $\frac{32N}{s} \cdot \log_2 \varepsilon$.

Step	**Operation Count**	**Explanation**
1	$N \log_2 \varepsilon + N$	Each particle is assigned to box at every level and there are at most $\log_2 \varepsilon$ levels of refinement. The extra N is required to modify the tree if necessary and generate lists.
2	$Np^2 + \frac{5p^4 N}{s} \cdot \log_2 \varepsilon$	Each particle contributes to a p^2-term expansion. Each multipole to multipole shift requires order p^4 operation and there are at most $\frac{5N}{s} \cdot \log_2 \varepsilon$ boxes.
3	$\sum_{\substack{i \le \frac{32N}{s} \cdot \log_2 \varepsilon \\ c_i \in L_4(b_i)}} \min\{p^2 s, s\lvert b_i\rvert\}$	Each box c_i that belongs to the list 4 of some box b_i contains less than s charges. $\lvert b_i\rvert$ is the number of charges contained in box b_i. The interactions between all particles in c_i and b_i require either $p^2 s$ work by generating local expansion about b_i's center or $s\lvert b_i\rvert$ work by using direct interaction. The option yielding the smaller operation count is selected. The total number of boxes in list 4 is bounded by $\frac{32N}{s} \cdot \log_2 \varepsilon$.
4	$\frac{5N}{s} \cdot \log_2 \varepsilon \cdot 27p^2$	For each box, its list 2 has at most 27 entries. Each multipole to local translation costs order p^2. There are at most $\frac{5N}{s} \cdot \log_2 \varepsilon$ boxes.

5	$\frac{5N}{s} \cdot \log_2 \varepsilon \cdot p^4$	Shifting the local expansion of a parent box to its four children can be done in order p^4 operations. There are at most $\frac{5N}{s} \cdot \log_2 \varepsilon$ boxes.
6	Np^2	A p^2-term expansion is evaluated at each particle location.
7	$\sum_{\substack{i \le \frac{32N}{s} \cdot \log_2 \varepsilon \\ c_i \in L_3(b_i)}} \min\{p^2 s, s\|c_i\|\}$	$\|c_i\|$ is the number of charges contained in box c_i. c_i belongs to list 3 of b_i which has less than s charges. The interactions between all particles in b_i and c_i require either $p^2 s$ by evaluating c_i's multipole expansion or $s\|c_i\|$ by using direct interaction. The option yielding the smaller operation count is selected. The total number of boxes in list 3 is bounded by $\frac{32N}{s} \cdot \log_2 \varepsilon$.
8	$\frac{44N}{s} \cdot \log_2 \varepsilon \cdot \frac{s^2}{2}$	The number of boxes in all list 1 is bounded by $\frac{44N}{s} \cdot \log_2 \varepsilon$. The work required to compute all interactions between particles in two boxes is $\frac{s^2}{2}$ when Newton's third law is used.

4.5 Numerical Results

In this section, we present three sets of numerical experiments to examine the correctness of the translation operators developed in Section 4.3 and explore the applicable scope of the algorithm. The experiments are based on an implementation of the uniform version of the algorithm in Matlab. The computing environment is a Dell workstation with two Xeon 5520 processors at 2.26 GHz and 48 GB memory.

The numerical results are summarized in Tables 4.5, 4.6, and 4.7. In each set of experiments, particles are distributed uniformly inside the unit square $[-0.5, 0.5]^2$. In each table, N denotes the number of particles used in the simulation. *nlev* denotes the maximum refinement level. For each N, we executed the program with several choices of *nlev* and the one yielding the smallest running time is reported. We compute the results using direct calculation at the first 200 generated point locations and then calculate the algorithm error according to Equation (2.24). The timing results in the T_{FMM} column are measured in seconds.

In the first set of experiments, the interaction between particles is described by $\frac{1}{r^2}$. The criterion for kernel approximation is to have at least 6 digits accuracy in the sense of relative error. Using the method described in Section 4.2, we obtained a particular set of p, α, and $\{\omega_{jk}\}$ that satisfies the requirement. The corresponding relative error plot in the first quadrant of $\Omega_{1,4}^2$ is depicted in Figure 4.3. In this figure, we have $p = 21$, $\alpha = \frac{2\pi}{9.25}$, and the maximum relative error is $8.4274 \cdot 10^{-7}$. From the plot, we conclude that the current result is not optimal and we should expect to have a smaller p value (fewer terms) in the optimal approximation. The timing results of this set of experiments are summarized in Table 4.5. First, we can observe that the CPU time requirements T_{FMM} grow approximately linearly with respect to the number of particles N. Secondly, we observe that for this particular particle distribution, we achieve 4 more digit accuracy than the expected 6-digit precision.

In the second set of experiments, the interaction between particles is still described by $\frac{1}{r^2}$. However, we impose a different criterion in the kernel approximation, requiring the result have at least 6 digits accuracy with respect to K_{max}, where $K_{max} = \|K(x,y)\|_\infty, (x,y) \in \Omega_{1,4}^2$. Notice that $K_{max} = 1$ and therefore in this case, we essentially control the absolute error in the kernel approximation. Again, a particular set of p, α, and $\{\omega_{j,k}\}$ is found to satisfy the accuracy requirement. The corresponding error plot in the first quadrant of $\Omega_{1,4}^2$ is depicted in Figure 4.4. In this figure, we have $p = 21$, $\alpha = \frac{2\pi}{9.05}$, and the maximum error is $3.6356 \cdot 10^{-7}$. We can conclude from the plot that the result is not optimal and we could use fewer terms to achieve the accuracy requirement. The timing results for this set of experiments are summarized in Table 4.6. We observe that the CPU time requirements T_{FMM} still grow approximately linearly with respect to the number of particles N. The accuracy and timing results are also comparable to those in Table 4.5. It implies that controlling error with respect to K_{max} is an alternative and competitive criterion to use in the kernel approximation.

In the third set of experiments, the interaction between particles is described by $\frac{x^2}{r^4}$. Notice that when $x = 0$, the exact value of the kernel function is zero and it is impossible to control relative error here. Alternatively, the criterion is to have at least 6 digits accuracy with respect to K_{max} defined the same as above. Again, we plot the error plot in the first quadrant of $\Omega_{1,4}^2$ corresponding to the set of p, α, and $\{\omega_{j,k}\}$ used in further computation, see Figure 4.5. In this

figure, we have $p = 24$, $\alpha = \dfrac{2\pi}{8.85}$, and the maximum error is $5.0338 \cdot 10^{-7}$. The timing results for this set of experiments are summarized in Table 4.7. We can observe the linear relationship between the CPU time requirements and the number of particles.

From the above experiments, we observe that once we satisfy the criterion on the kernel approximation, we will achieve the accuracy requirement in the computation of potentials. Comparing Table 4.5 and Table 4.7, we see that the CPU time requirements increase with p. It is crucial to develop schemes to find out optimal kernel approximation to reduce the value of p which further improves the CPU time requirements. This issue is currently under investigation and results will be reported at a further time.

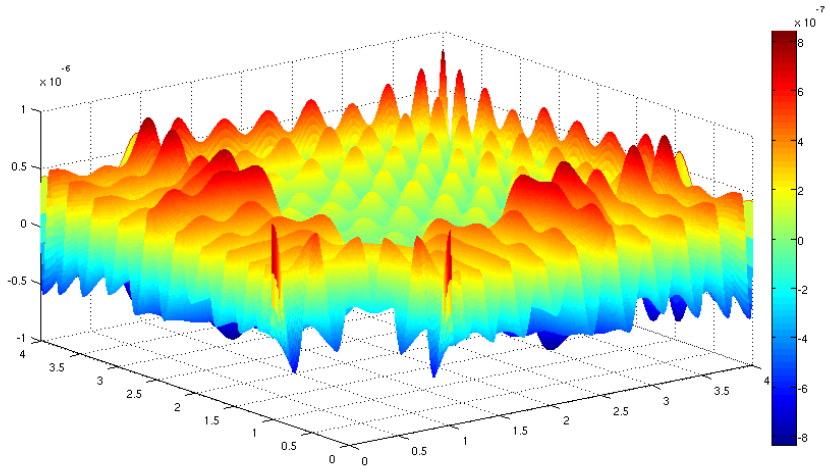


Figure 4.3: Relative approximation error for kernel $\frac{1}{r^2}$ in the first quadrant

N	$nlev$	Error	T_{FMM}
500	2	$1.8488 \cdot 10^{-10}$	0.9770
1000	2	$6.2183 \cdot 10^{-10}$	1.1254
2000	2	$1.6348 \cdot 10^{-10}$	1.4210
4000	2	$3.8448 \cdot 10^{-10}$	2.1359
8000	2	$7.3288 \cdot 10^{-11}$	3.9688
16000	2	$6.3476 \cdot 10^{-11}$	9.6384
32000	3	$6.1992 \cdot 10^{-11}$	18.8773
64000	3	$2.4997 \cdot 10^{-11}$	43.5001
128000	4	$3.5943 \cdot 10^{-11}$	79.9500

Table 4.5: Timing results of uniform FMM for 6-digit accuracy. Charges are distributed in the unit square whose interactions are described by kernel $\frac{1}{r^2}$. Control relative error in kernel approximation.

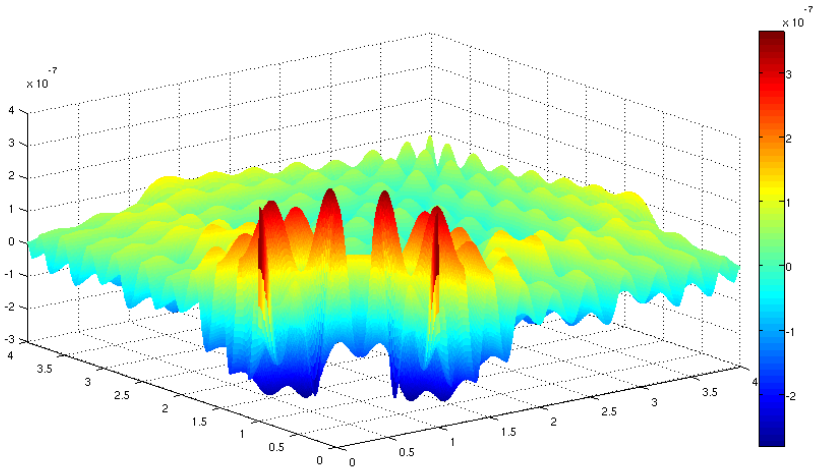


Figure 4.4: Absolute approximation error for kernel $\frac{1}{r^2}$ in the first quadrant

N	$nlev$	Error	T_{FMM}
500	2	$1.0432 \cdot 10^{-9}$	1.0043
1000	2	$2.5532 \cdot 10^{-10}$	1.2059
2000	2	$1.3817 \cdot 10^{-10}$	1.4667
4000	2	$2.3094 \cdot 10^{-10}$	2.1474
8000	2	$4.7091 \cdot 10^{-11}$	3.9991
16000	2	$9.5984 \cdot 10^{-11}$	10.1514
32000	3	$7.5233 \cdot 10^{-11}$	19.1434
64000	3	$1.2618 \cdot 10^{-11}$	46.0349
128000	4	$1.5739 \cdot 10^{-11}$	81.5294

Table 4.6: Timing results of uniform FMM for 6-digit accuracy. Charges are distributed in the unit square whose interactions are described by kernel $\frac{1}{r^2}$. Control absolute error in kernel approximation.

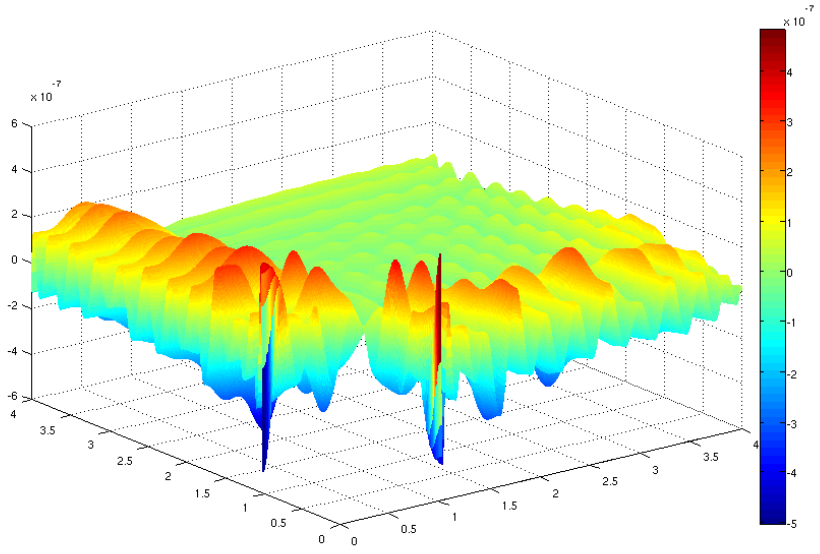


Figure 4.5: Absolute approximation error for kernel $\frac{x^2}{r^4}$ in the first quadrant

N	$nlev$	Error	T_{FMM}
500	2	$1.3523 \cdot 10^{-9}$	1.2287
1000	2	$2.1911 \cdot 10^{-9}$	1.4230
2000	2	$1.7255 \cdot 10^{-9}$	1.9184
4000	2	$8.7495 \cdot 10^{-10}$	3.2643
8000	2	$5.1591 \cdot 10^{-10}$	7.3304
16000	3	$6.2384 \cdot 10^{-10}$	15.8875
32000	3	$1.1186 \cdot 10^{-9}$	33.3443
64000	4	$1.7560 \cdot 10^{-9}$	71.1998
128000	4	$9.1502 \cdot 10^{-11}$	138.6034

Table 4.7: Timing results of uniform FMM for 6-digit accuracy. Charges are distributed in the unit square whose interactions are described by kernel $\frac{x^2}{r^4}$. Control absolute error in kernel approximation.

Chapter 5

Conclusion

The work in this thesis can be summarized from three aspects.

First, efforts have been devoted to software integration and dissemination. The complexities in mathematics, algorithms, architectures, and programming have made it important and necessary to provide FMM software to the community of computational science and engineering. I have contributed to the open-source new-version FMM packages with a host of solvers for the Laplace equation, the Yukawa equation, and the low-frequency Helmholtz equation, which frequently arise in scientific simulations and engineering designs. The packages are released under general public license at the website http://fastmultipole.org, which also hosts educational materials on various topics on integral equation methods.

Secondly, a graph-theoretic approach has been proposed to map FMM onto the parallel computer architectures (PCAs). Particularly, I established the critical path analysis, the exponential node growth condition for concurrency-breadth , a spatio-temporal graph partition scheme. With the theoretical results underlying the practical multi-threading parallel FMM design, simulations on the scale of hundred-millions of particles have been enabled on workstations with multicore processors in minutes.

Thirdly, a new kernel-independent FMM based on Fourier series expansions has been developed and applied to kernels with scaling property. The scheme only relies on the ability of kernel evaluation in a finite region, which extends the ideas of FMM to situations that the analytical kernel expansion is either unavailable or too expensive to compute. The translation operators are constructed in the receiver-initiated manner which shows great promise in parallelization on modern multicore computers.

The work in this dissertation has broad range of applications. Particularly, the FMM-Laplace and FMM-Yukawa packages have been incorporated into the Adaptive-Fast-Multipole-Poisson-Boltzmann (AFMPB) solver in collaboration with J. A. McCammon's group at UCSD in the molecular dynamics simulation. Currently, the corresponding parallel version is under development to achieve further performance improvement. For the kernel-independent FMM, it will be applied in the dislocation dynamics simulation in material science.

The work in this dissertation will be continued in several ways in the future. In the parallelization area, we will use dynamic thread creation and termination [22] to reach maximal parallel processing with minimal synchronization scope and overhead. We also would like to combine the scheme with MPI for inter-processors data exchanges, to accommodate larger data sets and increase parallel processing capacity. In the area of kernel independent FMM, we will study better strategies for kernel approximation to reduce the number of multipole expansion terms used in the algorithm. We also plan to further accelerate the multipole-to-multipole and local-to-local operators with fast convolution algorithms.

BIBLIOGRAPHY

[1] Fast Multipole Methods. http://fastmultipole.org.

[2] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. 1965.

[3] B. Alpert, C. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-Like Bases for the Fast Solution of Second-Kind Integral Equations. *SIAM Journal on Scientific Computing*, 14:159–184, 1993.

[4] C. R. Anderson. An Implementation of the Fast Multipole without Multipoles. *SIAM Journal on Scientific and Statistical Computing*, 13:923–947, 1992.

[5] A. W. Appel. An Efficient Program for Many-Body Simulation. *SIAM Journal on Scientific and Statistical Computing*, 6:85–103, 1985.

[6] J. Barnes and P. Hut. A Hierarchical $O(N \log N)$ Force-Calculation Algorithm. *Nature*, 324:446–449, 1986.

[7] A. J. Bernstein. Analysis of Program for Parallel Processing. *IEEE Transactions on Electronic Computers*, 15:757–762, 1966.

[8] G. Beylkin, V. Cheruvu, and F. Pérez. Fast Adaptive Algorithms in the Non-Standard Form for Multidimensional Problems. *Applied and Computational Harmonic Analysis*, 24:354–377, 2008.

[9] G. Beylkin, R. Coifman, and V. Rokhlin. Fast Wavelet Transforms and Numerical Algorithms I. *Communications on Pure and Applied Mathematics*, 44:141–183, 1991.

[10] G. Beylkin, R. Cramer, G. Fann, and R. Harrison. Multiresolution Separated Representations of Singular and Weakly Singular Operators. *Applied and Computational Harmonic Analysis*, 23:235–253, 2007.

[11] G. Beylkin and M. J. Mohlenkamp. Algorithms for Numerical Analysis in High Dimensions. *SIAM Journal on Scientific Computing*, 26:2133–2159, 2005.

[12] G. Beylkin and L. Monzón. On Approximation of Functions by Exponential Sums. *Applied and Computational Harmonic Analysis*, 19:17–48, 2005.

[13] A. Boschitsch, M. Fenley, and H. Zhou. Fast Boundary Element Method for the linear Poisson-Boltzmann Equations. *The Journal of Physical Chemistry B*, 106:2741–2754, 2002.

[14] A. H. Boschtisch, M. O. Fenley, and W. K. Olson. A Fast Adaptive Multipole Algorithm for Calculating Screened Coulomb (Yukawa) Interactions. *Journal of Computational Physics*, 151:212–241, 1999.

[15] A. Brandt. Multilevel Computations of Integral Transforms and Particle Interactions with Oscillatory Kernels. *Computer Physics Communications*, 65:24–38, 1991.

[16] J. Carrier, L. Greengard, and V. Rokhlin. A Fast Adaptive Multipole Algorithm for Particle Simulations. *SIAM Journal on Scientific and Statistical Computing*, 9:669–686, 1988.

[17] N. Carriero and D. Gelernter. Linda in Context. *Communication of the ACM*, 32:444–458, 1989.

[18] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. Greengard, F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao. A Wideband Fast Multipole Method of the Helmholtz Equation in Three Dimensions. *Journal of Computational Physics*, 216:300–325, 2006.

[19] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin. On the Compression of Low-Rank matrices. *SIAM Journal on Scientific Computing*, 26:1389–1404, 2005.

[20] H. Cheng, L. Greengard, and V. Rokhlin. A Fast Adaptive Multipole Algorithm in Three Dimensions. *Journal of Computational Physics*, 155:468–498, 1999.

[21] T. Darden, D. York, and L. Pedersen. Particle Mesh Ewald: An $N\log(N)$ Method for Ewald Sums in Large Systems. *Journal of Chemical Physics*, 98:10089–10092, 1993.

[22] E. R. Davidson and T. H. Cormen. Asynchronous Buffered Computational Design and Engineering Framework Generator (ABCDEFG).

[23] P. Debye and E. Hückel. Zur Theorie der Elektrolyte. *Physikalische Zeitschrift*, 24(9):185–206, 1923.

[24] F. Ethridge and L. Greengard. A New Fast-Multipole Accerlated Poisson Solver in Two Dimensions. *SIAM Journal on Scientific Computing*, 23:741–760, 2001.

[25] W. Fong and E. Darve. The Black-Box Fast Multipole Method. *Journal of Computational Physics*, 228:8712–8725, 2009.

[26] Y. Fu and G. J. Rodin. Fast Solution Methods for 3D Stokesian Many-Particle Problems. *Communications in Numerical Methods in Engineering*, 16:145–149, 2000.

[27] M. Gilson, A. Rashin, R. Fine, and B. Honig. On the Calculation of Electrostatic Interactions in Proteins. *Journal of Molecular Biology*, 184:503–516, 1985.

[28] Z. Gimbutas and V. Rokhlin. A Generalized Fast Multipole Method for Nonoscillatory Kernels. *SIAM Journal on Scientific Computing*, 24:796–817, 2003.

[29] D. Goeddeke, R. Strzodka, and S. Turek. Accelerating double precision FEM simulations with GPUs. In *Proceedings of ASIM 2005*, 2005.

[30] L. Greengard and W. Gropp. A Parallel Version of the Fast Multipole Method. *Computers & Mathematics with Applications*, 20:63–71, 1990.

[31] L. Greengard and J. Huang. A New Version of the Fast Multipole Method for Screened Coulomb Interactions in Three Dimensions. *Journal of Computational Physics*, 180(2):642–658, 2002.

[32] L. Greengard, J. Huang, and V. Rokhlin. Accelerating Fast Multipole Methods for the Helmholtz Equation at Low Frequencies. *IEEE Computational Science & Enginerring*, 5:32–38, 1998.

[33] L. Greengard, M. Kropinski, and A. Mayo. Integral Equation Methods for Stokes Flow and Isotropic Elasticity in the Plane. *Journal of Computational Physics*, 125:403–414, 1996.

[34] L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[35] L. Greengard and V. Rokhlin. A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions. *Acta Numerica*, 6:229–269, 1997.

[36] N. A. Gumerov and R. Duraiswami. Fast Multipole Methods on Graphics Processors. *Journal of Computational Physics*, 227:8290–8313, 2008.

[37] R. Harrison, G. Fann, T. Yanai, Z. Gan, and G. Beylkin. Multiresolution Quantum Chemistry: Basic Theory and Initial Applications. *Journal of Chemical Physics*, 121:11587–11598, 2004.

[38] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. Taylor & Francis, Inc., 1988.

[39] T. Hrycak and V. Rokhlin. An Improved Fast Multipole Algorithm for Potential Fields. *SIAM Journal on Scientific Computing*, 19:1804–1826, 1998.

[40] J. Huang, J. Jia, and B. Zhang. FMM-Yukawa: An Adaptive Fast Multipole Method for Screened Coulomb Interactions. *Computer Physics Communications*, 180:2331–2338, 2009.

[41] J. A. B. Jr., J. W. Causey, and J. F. L. Jr. Accelerated Molecular Dynamics Simulation with the Parallel Fast Multipole Algorithm. *Chemical Physics Letters*, 198:89–94, 1992.

[42] A. J. Juffer, E. F. F. Botta, B. A. M. van Keulen, A. van der Ploeg, and H. J. C. Berendsen. The Electric Potential of a Macromolecule in a Solvent: A Fundamental Approach. *Journal of Computational Physics*, 97(1):144–171, 1991.

[43] S. S. Kuo, M. D. Altman, J. P. Bardhan, B. Tidor, and J. K. White. Fast Methods for Simulation of Biomolecule Electrostatics. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design*, pages 466–473, 2002.

[44] J. Liang and S. Subramaniam. Computation of Molecular Electrostatics with Boundary Element Methods. *Biophysical Journal*, 73(4):1830–1841, 1997.

[45] B. Lu, X. Cheng, J. Huang, and J. McCammon. Order N Algorithm for Computation of Electrostatic Interactions in Biomolecular Systems. In *Proceedings of the National Academy of Sciences*, volume 103, pages 19314–19319, 2006.

[46] B. Lu, X. Cheng, J. Huang, and J. A. McCammon. AFMPB: An Adaptive Fast Multipole Poisson-Boltzmann Solver for Calculating Electrostatics in Biomolecular Systems. *Computer Physics Communications*, 181:1150–1160, 2010.

[47] B. Lu, X. Cheng, and J. A. McCammon. "New-Version-Fast-Multipole-Method" Accelerated Electrostatic Calculations in Biomolecular Systems. *Journal of Computational Physics*, 226:1348–1366, 2007.

[48] C. C. Lu and W. C. Chew. Fast Algorithm for Solving Hybrid Integral Equations. *IEEE Proceedings H*, 140:455–460, 1993.

[49] C. C. Lu and W. C. Chew. A Multilevel Algorithm for Solving a Boundary Integral Equation of Wave Scattering. *Microwave and Optical Technology Letters*, 7:466–470, 1994.

[50] P. G. Martinsson and V. Rokhlin. A Fast Direct Solver for Boundary Integral Equations in Two Dimensions. *Journal of Computational Physics*, 205:1–23, 2005.

[51] P. G. Martinsson and V. Rokhlin. An Accelerated Kernel-Independent Fast Multipole Method in One Dimensions. *SIAM Journal on Scientific Computing*, 29:1160–1178, 2007.

[52] E. Michielssen, A. Ergin, B. Shanker, and D. Weile. The Multilevel Plane Wave Time Domain Algorithm and Its Applications to the Rapid Solution of Electromagnetic Scattering Problems: A Review. In *Fifth International Conference on Mathematical and Numerical Aspects of Wave Propagation*, pages 24–33, 2000.

[53] K. Nabors, S. Kim, and J. White. Fast Capacitance Extraction of General Three-Dimensional Structures. *IEEE Transactions on Microwave Theory and Technology*, 40:1496–1505, 1992.

[54] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White. Preconditioned, Adaptive, Multipole-Accelerated Iterative Methods for Three-Dimensional First-Kind Integral Equations of Potential Theory. *SIAM Journal on Scientific and Statistical Computing*, 15:713–735, 1994.

[55] N. Nishimura. Fast Multipole Accelerated Boundary Integral Equation Methods. *Applied Mechanics Reviews*, 55:299–324, 2002.

[56] L. Nyland, M. Harris, and J. Prins. Fast N-Body Simulation with CUDA. In H. Nguyen, editor, *GPU Gems3*, pages 677–696. Addison Wesley, 2007.

[57] J. R. Phillips, J. K. White, and A. Member. A Precorrected-FFT Method for Electrostatic Analysis of Complicated 3-D Structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16:1059–1072, 1997.

[58] V. Popov and H. Power. An $O(N)$ Talyor Series Multipole Boundary Element Method for Three-Dimensional Elasticity Problems. *Engineering Analysis with Boundary Elements*, 25:7–18, 2002.

[59] V. Rokhlin. Rapid Solution of Integral Equations of Scattering Theory in Two Dimensions. *Journal of Computational Physics*, 86:414–439, 1990.

[60] V. Rokhlin. Diagonal Forms of Translation Operators for the Helmholtz Equation in Three Dimensions. *Applied and Computational Harmonic Analysis*, 1:82–93, 1993.

[61] J. Singh, C. Holt, J. Hennessy, and A. Gupta. A Parallel Adaptive Fast Multipole Method. In *SC 93': Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, 1993.

[62] J. Tausch. The Fast Multipole Method for Arbitrary Green's Functions. *Contemporary Mathematics*, 329:307–314, 2003.

[63] A. Tornberg and L. Greengard. A Fast Multipole Method for the Three-Dimensional Stokes Equations. *Journal of Computational Physics*, 227:1613–1619, 2008.

[64] D. A. S. W. B. Russell and W. R. Schowalter, editors. *Colloidal Dispersions*. Cambridge University Press, 1991.

[65] H. Wang, T. Lei, J. Li, J. Huang, and Z. Yao. A Parallel Fast Multipole Accelerated Integral Equation Scheme for 3D Stokes Equations. *International Journal for Numerical Methods in Engineering*, 70:812–839, 2007.

[66] M. Warren and J. Salmon. Astrophysical N-Body Simulation Using Hierarchical Tree Data Structures. In *SC 92': Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, 1992.

[67] M. Warren and J. Salmon. A Parallel Hashed Oct-Tree N-Body Algorithm. In *SC 93': Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, 1993.

[68] A. Windemuth and K. Schulten. Molecular Dynamics Simulation on the Connection Machine . *Molecular Simulation*, 5:353–361, 1991.

[69] Z. H. Yao, P. B. Wang, T. Lei, and H. T. Wang. Large-Scale Boundary Element Analysis in Solid Mechanics Using Fast Multipole Method. In *Proceedings of "Enhancement and Promotion of Computational Methods in Engineering and Science X"*, 2006.

[70] N. Yarvin and V. Rokhlin. Generalized Gaussian Quadratures and Singular Value Decomposition of Integral Operators. *SIAM Journal on Scientific Computing*, 20:699–718, 1998.

[71] N. Yarvin and V. Rokhlin. An Improved Fast Multipole Algorithm for Potential Fields on the Line. *SIAM Journal on Numerical Analysis*, 36:629–666, 1999.

[72] W. Ye, J. Kanapa, and J. White. A Fast 3-D Solver for Unsteady Stokes Flow with Application to Micro-Electro-Mechanical Systems. In *International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, 1999.

[73] W. Ye, J. Kanapka, X. Wang, and J. White. Efficiency and Accuracy Improvements for FastStokes: A Precorrected-FFT Accelerated 3-D Stokes Solver. In *International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, 1999.

[74] W. Ye, X. Wang, and J. White. A Fast Stokes Solver for Generalized Flow Problems. In *International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, 2000.

[75] L. Ying, G. Biros, and D. Zorin. A Kernel-Independent Adaptive Fast Multipole Algorithm in Two and Three Dimensions. *Journal of Computational Physics*, 196:591–626, 2004.

[76] K. Yoshida. *Applications of Fast Multipole Method to Boundary Integral Equation Method*. PhD thesis, Department of Global Environment Engineering, Kyoto University, 2001.

[77] K. Yoshida, N. Nishimura, and S. Kobayashi. Application of Fast Multipole Galerkin Boundary Integral Equation Method to Elastostatic Crack Problems in 3D. *International Journal for Numerical Methods in Engineering*, 50:525–547, 2001.

[78] K. Yoshida, N. Nishimura, and S. Kobayashi. Application of New Fast Multipole Boundary Integral Equation Method to Crack Problems in 3D. *Engineering Analysis with Boundary Elements*, 25:239–247, 2001.

[79] D. Zhao, J. Huang, and Y. Xiang. A New Version Fast Multipole Method for Evaluating the Stress Field of Dislocation Ensembles. *Modeling and Simulation in Materials Sciences and Engineering*, 18, 2010.

[80] F. Zhao and S. L. Johnsson. The Parallel Multipole Method on the Connection Machine. *SIAM Journal on Scientific and Statistical Computing*, 12:1420–1437, 1991.

[81] J. S. Zhao and W. C. Chew. MLFMA for Solving Integral Equations of 2-D Electromagnetic Problems form Static to Electrodynamic. *Microwave and Optical Technology Letters*, 20:306–311, 1999.

[82] J. S. Zhao and W. C. Chew. Three-Dimensional Multilevel Fast Multipole Algorithm from Static to Electrodynamic. *Microwave and Optical Technology Letters*, 26:43–48, 2000.