Cartesian Treecode Algorithms for Electrostatic Interactions in Molecular Dynamics Simulations

by

Henry A. Boateng

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Applied and Interdisciplinary Mathematics)
in The University of Michigan
2010

Doctoral Committee:

Professor Robert Krasny, Co-Chair
Associate Professor Eitan Geva, Co-Chair
Professor Smadar Karni
Assistant Professor Barry Dov Dunietz

ACKNOWLEDGEMENTS

Thanks to my advisor Robert Krasny for a very fulfilling education. His patience, guidance and tact were instrumental to the completion of this thesis. I value the training he gave me and his insistence on excellence in every aspect of research.

Thanks also to my co-advisor Eitan Geva who helped demystify theoretical chemistry for me and to Barry Dunietz for all the feedback.

Very special thanks to Smadar Karni for her wonderful mentoring, friendship and support throughout my studies. They mean more than I can say.

Most will confess that a pursuing a Ph.D can be very lonely. During those long lonely days, I was fortunate to have office mates, Tomoki Ohsawa and Catherine Kublik, who were also friends. I will miss all the good-natured teasing and the commiseration.

Thanks to the Graduate Student Services office, especially Tara McQueen. I have lost count of how many times you exceeded my expectations with your prompt response to my queries.

Finally, I want to thank my wife, Francesca L. Kaglar-Boateng, for providing unconditional support and love through all the turbulence and shocks. This is for you.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER I

Introduction

This dissertation presents work at the intersection of scientific computing and computational chemistry. The focus is on the development and application of fast Cartesian multipole treecode algorithms for evaluating coulombic (or electrostatic) interactions in molecular dynamics (MD) simulations.

MD simulations provides a means to study the dynamical and structural properties of a system of particles by numerically following the path of the system in phase space. We can verify the theories and models underlying our simulations by comparing the simulation results to experimental results. Having verified the underlying theories and models, MD simulations then provides us a means to computationally study systems which will otherwise be too difficult or impossible to probe experimentally and to have confidence in the results of our study.

Tracing the path of a system in phase space involves propagating the system in space by a time-stepping method according to Newton's equations of motion. For a system of N particles interacting through a potential V, MD simulations entails solving Newton's equations of motion

$$m_i \ddot{\mathbf{r}}_i = \mathbf{f}_i, \tag{1.0.1}$$

in time for each particle, i, in the system with $1 \leq i \leq N$, m_i the particle mass and

$\mathbf{r}_i$ the position of particle. The force $\mathbf{f}_i$ on particle i is given as

$$\mathbf{f}_i = -\nabla_{\mathbf{r}_i} V. \tag{1.0.2}$$

The interaction potential V usually includes electrostatic interactions because of the high prevalence of charged species in most systems of interest. The computation of the electrostatic interactions are frequently the bottleneck of MD simulations because they require interactions between all particles to achieve an acceptable accuracy. As a result methods that speed up the computation of electrostatic interactions are very important. Fast methods for electrostatic interactions are especially important in the modeling of chemical systems and biological molecules where the number of atoms in a system is routinely greater than 10^4. Simulations using explicit particle-particle interactions will take very long times, hence the need for faster and accurate alternative methods [11, 26, 38].

In addition to MD simulations, treecode algorithms can also be employed in several other fields including Monte Carlo [2, 25], dissipative particle dynamics [23] and fluid dynamics [38] simulations where N-body interactions are prevalent.

1.1 Thesis Contributions

The Particle Mesh Ewald (PME) method is the state of the art for performing fast coulombic interactions in systems with periodic boundary conditions. The overarching goal of this work is to advance a treecode alternative or addition to the Particle Mesh Ewald (PME) method [12, 21]. This thesis presents three related projects in furtherance of this goal.

- The first project validates the Cartesian treecode Ewald (CTE) method [16] by comparing the results of molecular dynamics (MD) simulations using CTE to those using PME. The algorithms are compared based on the computed

structural and dynamical properties of liquid methyl chloride, CH_3Cl, and on cpu time. The thesis also investigates the possibility of a hybrid CTE-PME algorithm.

- The second project develops a cluster-cluster Cartesian treecode algorithm. This is in response to the original particle-cluster CTE being slower than PME for the same sized system. The leaf-cluster Cartesian treecode algorithm, a variant of cluster-cluster, is shown to be about twice as fast as the particle-cluster algorithm for the free space Coulomb potential at the same accuracy level.

- The third project addresses the efficiency of CTE by parallelizing the algorithm. The goal is to incorporate CTE in DL_POLY_2 [52, 53] and DL_POLY_3 [55, 54], which are parallel software packages for performing MD simulations. Two different parallel treecode algorithms are developed since DL_POLY_2 and DL_POLY_3 are based on two different paradigms for parallel algorithms. In DL_POLY_2, the replicated data strategy, where all particles are replicated on all nodes, is used to achieve load balancing. DL_POLY_3 load balances via a decomposition of the simulation space into the nodes. This approach is known as domain decomposition where only subsets of the full system are stored on each node and the union of all these subsets produces the whole system.

1.2 Overview

The thesis is organized as follows. Chapter II presents a derivation of the Ewald sum for handling the Coulomb potential for periodic boundary conditions. We also develop the particle mesh Ewald method and Cartesian treecode Ewald method in this chapter. In Chapter III, we explain the approach for handling periodic boundary conditions for CTE and provide results from MD simulation of liquid CH_3Cl which

validates the CTE algorithm when compared to PME. The chapter also provides time comparison and studies of a hybrid CTE-PME algorithm. The cluster-cluster algorithm is presented in Chapter IV including the development, error analysis, the code and comparisons to particle-cluster. Chapter V presents the parallel CTE algorithms and provides scaling results for the two different algorithms. We conclude in Chapter VI with a summary and provide directions for possible future work.

CHAPTER II

Coulomb Potential in Periodic Boundary Conditions

2.1 Molecular Dynamics (MD)

Molecular dynamics involves simulating the time evolution of a system of chemical species by integrating the classical equations of motion of the microscopic constituents of the species, usually atoms and molecules. Insights into the behavior of the system are gained from computing statistical ensemble averages of relevant structural and dynamical functions [2, 25].

Molecular dynamics (MD) simulations of biomolecular systems and liquids with charged species usually involve computing Coulomb pair potentials due to electrostatic interactions. The total Coulomb pair potential of N charged particles in free space, is given as

$$V = \frac{1}{2}\sum_{i\neq j}^{N} \frac{q_i q_j}{4\pi\epsilon_o \mid \mathbf{x}_i - \mathbf{x}_j \mid}, \tag{2.1.1}$$

where q_i and $\mathbf{x}_i \in \mathbb{R}^3$ are the charge and position of particle i respectively and $\epsilon_o = 1$ is the dielectric constant

In practice, to study bulk liquids, simulations are frequently carried out by considering a system of net zero total charge,

$$\sum_{i=1}^{N} q_i = 0, \tag{2.1.2}$$

in a cubic box of size L. Periodic boundary conditions (PBC) are then employed to model the bulk liquid and to minimize surface effects. Periodic boundary conditions are implemented by replicating the fundamental system periodically in space to form an infinite system as depicted in (Figure 2.1) in two dimensions.

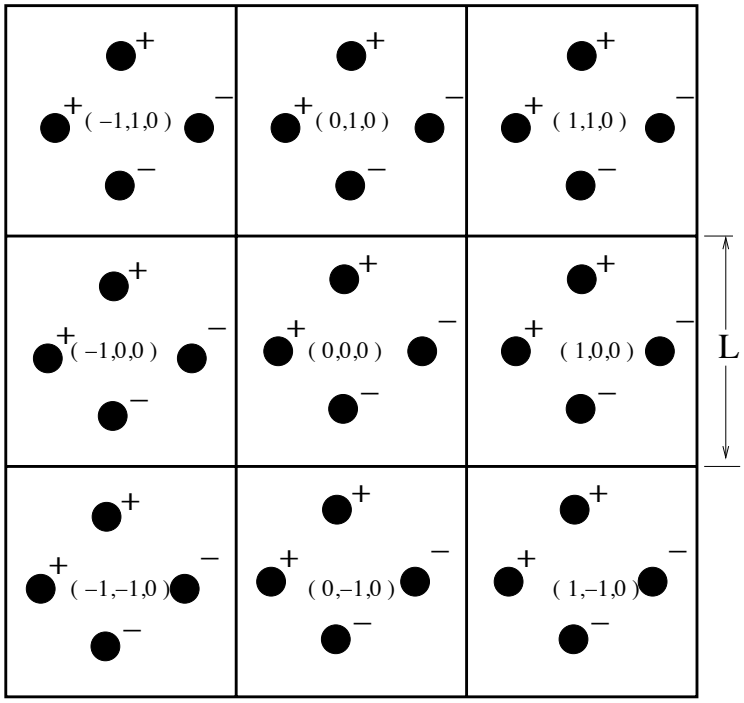


Figure 2.1: The fundamental cell has net zero charge meaning the image cells also have net zero charge. Thus, the net charge of the infinite system is zero.

The total Coulomb potential is then a sum over the fundamental cell and its infinite periodic images,

$$V = \frac{1}{2}\sum_{i,j=1}^{N} q_i q_j \sum_{\mathbf{n}}{}' \frac{1}{4\pi\epsilon_o \mid \mathbf{x}_i - \mathbf{x}_j + \mathbf{n}L \mid}, \tag{2.1.3}$$

where $\mathbf{n} = (n_1, n_2, n_3)$ identifies a unique cell. Here, $n_i \in \mathbb{Z}$ for $i = 1, 2, 3$ and the prime refers to the fact that the $i = j$ term is omitted when $\mathbf{n} = (0, 0, 0)$ which is the fundamental cell. The series in Equation 2.1.3 is a conditionally convergent infinite series and hence it converges slowly.

2.2 The Ewald Sum

Because of the long range effect of the potential in Equation 2.1.3, cutoff methods employed in computing short range potentials are not suitable for computing the Coulomb potential [2, 45]. In addition, the sum in Equation 2.1.3 depends on the order of summation since the series is conditionally convergent [13, 32].

Ewald summation [22] provides an efficient method for evaluating Equation 2.1.3. The summation arises from being able to rewrite the series in Equation 2.1.3 as a sum of an absolutely convergent series in real space, another absolutely convergent series in reciprocal space and a finite sum. Here, we give a self-contained derivation of the Ewald sum following the development in [32]. An alternative derivation is given in [13].

We can rewrite Equation 2.1.3 as the sum of interactions where $i \neq j$, meaning that $\mathbf{x}_i - \mathbf{x}_j \neq \mathbf{0}$, and interactions where $i = j$ which means that $\mathbf{x}_i - \mathbf{x}_j = \mathbf{0}$ yielding

$$V = \frac{1}{2}\sum_{i\neq j}^{N} q_i q_j \sum_{\mathbf{n}} \frac{1}{4\pi\epsilon_o \mid \mathbf{x}_i - \mathbf{x}_j + \mathbf{n}L \mid} + \frac{1}{2}\sum_{i=1}^{N} q_i^2 \sum_{\mathbf{n}\neq 0} \frac{1}{4\pi\epsilon_o \mid \mathbf{n}L \mid}. \tag{2.2.1}$$

The first summation in Equation 2.2.1 runs over all values of $\mathbf{n}$ since $i \neq j$, while the second summation with $i = j$ excludes the $\mathbf{n} = 0$ term. Now note that

$$\sum_{\mathbf{n}} \frac{1}{4\pi\epsilon_o \mid \mathbf{x}_i - \mathbf{x}_j + \mathbf{n}L \mid} \tag{2.2.2}$$

is a divergent lattice sum for all $\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j$, however, it makes sense as a distribution in the form

$$-\Delta \sum_{\mathbf{n}} \frac{1}{4\pi \mid \mathbf{x} + \mathbf{n}L \mid} = \sum_{\mathbf{n}} \delta(\mathbf{x} + \mathbf{n}L), \tag{2.2.3}$$

where we have set $\epsilon_o = 1$. Additionally, when $\mathbf{n} \neq 0$,

$$-\Delta \sum_{\mathbf{n}\neq 0} \frac{1}{4\pi \mid \mathbf{x} + \mathbf{n}L \mid} = \sum_{\mathbf{n}} \{\delta(\mathbf{x} + \mathbf{n}L)\} - \delta(\mathbf{x}). \tag{2.2.4}$$

This means that, we can reformulate the problem of evaluating the series in Equation 2.2.1 as a problem of finding the potential functions $\psi(\mathbf{x})$ and $\tilde{\psi}(\mathbf{x})$ induced by point charge densities $\rho(\mathbf{x})$ and $\tilde{\rho}(\mathbf{x})$ respectively from

$$-\Delta\psi(\mathbf{x}) = \rho(\mathbf{x}) = \sum_{\mathbf{n}} \delta(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3}, \tag{2.2.5}$$

$$-\Delta\tilde{\psi}(\mathbf{x}) = \tilde{\rho}(\mathbf{x}) = \rho(\mathbf{x}) - \delta(\mathbf{x}), \tag{2.2.6}$$

with periodic boundary conditions. We evaluate $\tilde{\psi}(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j = 0$ and replace the divergent lattice sums in Equation 2.2.1 by the respective induced potentials $\psi(\mathbf{x})$ and $\tilde{\psi}(0)$ then Equation 2.2.1 becomes

$$V = \frac{1}{2}\sum_{i\neq j}^{N} q_i q_j \psi(\mathbf{x}_i - \mathbf{x}_j) + \frac{1}{2}\sum_{i=1}^{N} q_i^2 \tilde{\psi}(0). \tag{2.2.7}$$

Poisson's equation with periodic boundary conditions is solvable if the system is charge neutral, which is why the charge density in Equation 2.2.5 is augmented by the constant term $\frac{-1}{L^3}$. Then, integrating the charge density over the fundamental cell or any of the images, with volume L^3, results in

$$\int_{cell} \rho(\mathbf{x})dx = \int_{cell} \sum_{n} \delta(\mathbf{x}+\mathbf{n}L)dx - \frac{1}{L^3}\int_{cell} dx = 1 - 1 = 0, \tag{2.2.8}$$

which shows that each cell and hence the bulk system is charge neutral.

Similarly, the charge density $\tilde{\rho}(\mathbf{x})$ in Equation 2.2.6 also yields a charge neutral system. To show this we note that $\tilde{\rho}(\mathbf{x})$ is only defined for $\mathbf{n} \neq 0$. We expand $\tilde{\rho}(\mathbf{x})$ from Equation 2.2.6 to arrive at

$$\begin{aligned} \tilde{\rho}(\mathbf{x}) &= \sum_{\mathbf{n}} \delta(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3} - \delta(\mathbf{x}) \\ &= \sum_{\mathbf{n}\neq 0} \delta(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3} + \delta(\mathbf{x}) - \delta(\mathbf{x}) \\ &= \sum_{\mathbf{n}\neq 0} \delta(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3}, \end{aligned} \tag{2.2.9}$$

which is charge neutral from the same arguments used in Equation 2.2.8 where the fundamental cell, i.e. $\mathbf{n} = 0$ is not part of the cells we consider.

We will find the potential function $\psi(\mathbf{x})$ from Equation 2.2.5 as a sum of two absolutely convergent series and a finite series. First, we rewrite $\rho(\mathbf{x})$ as

$$\rho(\mathbf{x}) = \sum_{\mathbf{n}} [\delta(\mathbf{x}+\mathbf{n}L) - f(\mathbf{x}+\mathbf{n}L)] + \sum_{\mathbf{n}} f(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3} = \rho_1(\mathbf{x}) + \rho_2(\mathbf{x}), \tag{2.2.10}$$

where $f(\mathbf{x})$ is a Gaussian with width α,

$$f(\mathbf{x}) = \frac{\alpha^3}{\pi^{3/2}} e^{-\alpha^2|\mathbf{x}|^2}, \tag{2.2.11}$$

and

$$\int_{\mathbb{R}^3} f(\mathbf{x}) d\mathbf{x} = 1. \tag{2.2.12}$$

We should note that other forms of $f(\mathbf{x})$ can be used and that the choice made here is only specific to Ewald sums. Figure 2.2 shows the split of the charge density $\rho(\mathbf{x})$ in Equation 2.2.10. The effect of $-f(\mathbf{x})$ is to screen a point charge, $\delta(\mathbf{x})$, in $\rho(\mathbf{x})$ and this effect is nullified by adding $f(\mathbf{x})$. So we now have two charge densities $\rho_1(\mathbf{x})$ which is short range and $\rho_2(\mathbf{x})$ which is smooth.

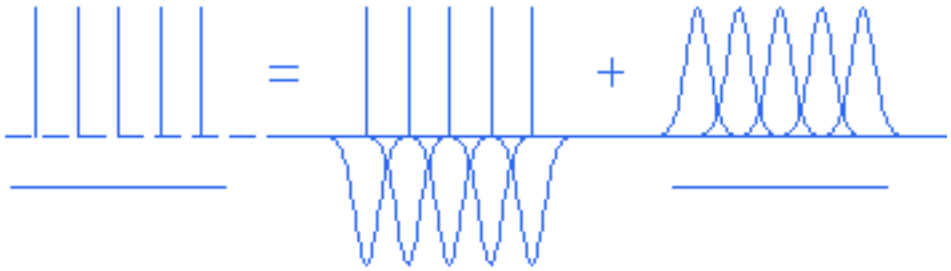


Figure 2.2: The figure on the left hand side of the equation is the original density $\rho(\mathbf{x})$. The first figure on the right hand side of the equation is $\rho_1(\mathbf{x})$ which is the Gaussian screenings the delta functions. The last figure on the right hand side is $\rho_2(\mathbf{x})$ which is a combination of the nullifying Gaussians and the constant term that ensures charge neutrality.

We note that both $\rho_1(\mathbf{x})$ and $\rho_2(\mathbf{x})$ individually satisfy the necessary condition of

charge neutrality with,

$$\int_{cell} \rho_1(\mathbf{x})d\mathbf{x} = \int_{cell} \sum_{\mathbf{n}} \left[\delta(\mathbf{x}+\mathbf{n}L) - f(\mathbf{x}+\mathbf{n}L)\right] d\mathbf{x}, \tag{2.2.13}$$

$$= \int_{\mathbb{R}^3} \{\delta(\mathbf{x}) - f(\mathbf{x})d\mathbf{x}\} = 0, \tag{2.2.14}$$

$$\int_{cell} \rho_2(\mathbf{x})d\mathbf{x} = \int_{cell} \left[\sum_{\mathbf{n}} f(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3}\right] d\mathbf{x}, \tag{2.2.15}$$

$$= \int_{\mathbb{R}^3} f(\mathbf{x})d\mathbf{x} - \frac{1}{L^3}\int_{cell} d\mathbf{x} = 0. \tag{2.2.16}$$

In effect, the problem of finding $\psi(\mathbf{x})$ has been reformulated into finding $\psi_1(\mathbf{x})$ and $\psi_2(\mathbf{x})$ where,

$$\psi(\mathbf{x}) = \psi_1(\mathbf{x}) + \psi_2(\mathbf{x}), \tag{2.2.17}$$

such that

$$-\Delta\psi_1(\mathbf{x}) = \rho_1(\mathbf{x}) = \sum_{\mathbf{n}} \left[\delta(\mathbf{x}+\mathbf{n}L) - f(\mathbf{x}+\mathbf{n}L)\right] \tag{2.2.18}$$

$$-\Delta\psi_2(\mathbf{x}) = \rho_2(\mathbf{x}) = \sum_{\mathbf{n}} f(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3}, \tag{2.2.19}$$

with periodic boundary conditions. We can solve for $\psi_1(\mathbf{x})$ and $\psi_2(\mathbf{x})$ up to an additive constant. We enforce the conditions

$$\int_{cell} \psi_1(\mathbf{x})d\mathbf{x} = = 0, \tag{2.2.20}$$

$$\int_{cell} \psi_2(\mathbf{x})d\mathbf{x} = = 0, \tag{2.2.21}$$

in order to fully determine $\psi_1(\mathbf{x})$ and $\psi_2(\mathbf{x})$.

We now show that

$$\psi_1(\mathbf{x}) = \sum_{\mathbf{n}} \frac{\operatorname{erfc}(\alpha \mid \mathbf{x}+\mathbf{n}L \mid)}{4\pi \mid \mathbf{x}+\mathbf{n}L \mid} - \frac{1}{4L^3\alpha^2}, \tag{2.2.22}$$

$$\psi_2(\mathbf{x}) = \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2} + 2\pi i \mathbf{k}\cdot\mathbf{x}/L}}{4\pi^2 L|\mathbf{k}|^2} \tag{2.2.23}$$

satisfy Equations 2.2.18 and 2.2.19 respectively with $\mathbf{k} = (k_1, k_2, k_3)$ and $k_i \in \mathbb{Z}$ for $i = 1, 2, 3$. $\psi_1(\mathbf{x})$ and $\psi_2(\mathbf{x})$ are both absolutely convergent series with $\psi_1(\mathbf{x})$ defined in real space and $\psi_2(\mathbf{x})$ defined in reciprocal space.

To prove Equation 2.2.18, first note that

$$\operatorname{erfc}(\alpha r) = 1 - \operatorname{erf}(\alpha r) = 1 - \frac{2}{\sqrt{\pi}} \int_0^{\alpha r} e^{-s^2} ds. \tag{2.2.24}$$

Then,

$$\begin{aligned}
-\Delta\psi_1(\mathbf{x}) &= -\Delta \sum_{\mathbf{n}} \frac{1}{4\pi \mid \mathbf{x} + \mathbf{n}L \mid} + \Delta \sum_{n} \frac{\operatorname{erf}(\alpha \mid \mathbf{x} + \mathbf{n}L \mid)}{4\pi \mid \mathbf{x} + \mathbf{n}L \mid}, && (2.2.25)\\
&= \sum_{\mathbf{n}} \left[\delta(\mathbf{x} + \mathbf{n}L) + \frac{1}{r^2} \partial_r (r^2 \partial_r) \frac{\operatorname{erf}(\alpha r)}{4\pi r} \right], && (2.2.26)\\
&= \sum_{\mathbf{n}} \left[\delta(\mathbf{x} + \mathbf{n}L) - \frac{\alpha^3}{\pi^{3/2}} \mathrm{e}^{-\alpha^2 r^2} \right], && (2.2.27)\\
&= \sum_{\mathbf{n}} \left[\delta(\mathbf{x} + \mathbf{n}L) - f(\mathbf{x} + \mathbf{n}L) \right], && (2.2.28)\\
&= \rho_1(\mathbf{x}). && (2.2.29)
\end{aligned}$$

which proves Equation 2.2.18. The constant term, $C = -\frac{1}{4L^3\alpha^2}$, in Equation 2.2.22 is accounted for by the requirement from Equation 2.2.20 that

$$\int_{cell} \psi(\mathbf{x})_1 d\mathbf{x} = 0, \tag{2.2.30}$$

which gives,

$$\begin{aligned}
\int_{cell} \left[\sum_{\mathbf{n}} \frac{\operatorname{erfc}(\alpha \mid \mathbf{x} + \mathbf{n}L \mid)}{4\pi \mid \mathbf{x} + \mathbf{n}L \mid} + C \right] d\mathbf{x} &= 0,\\
\int_{\mathbb{R}^3} \frac{\operatorname{erfc}(\alpha x)}{4\pi |\mathbf{x}|} d\mathbf{x} + C \cdot L^3 &= 0,\\
-\frac{1}{L^3} \cdot \int_S \int_0^{\infty} \frac{\operatorname{erfc}(\alpha r)}{4\pi r} r^2 dr d\Omega &= C,\\
-\frac{1}{L^3\alpha^2} \cdot \int_0^{\infty} \operatorname{erfc}(s) s ds &= C.
\end{aligned}$$

By integration by parts,

$$\int_0^\infty \operatorname{erfc}(s)sds = \int_0^\infty \frac{2}{\sqrt{\pi}} \int_s^\infty \mathrm{e}^{-t^2} dtsds \tag{2.2.31}$$

$$= \frac{2}{\sqrt{\pi}} \left[\frac{s^2}{2} \int_s^\infty \mathrm{e}^{-t^2} dt \right]_0^\infty + \frac{2}{\sqrt{\pi}} \int_0^\infty \frac{s^2}{2} \mathrm{e}^{-s^2} ds, \tag{2.2.32}$$

$$= \frac{1}{\sqrt{\pi}} \int_0^\infty s^2 \mathrm{e}^{-s^2} ds = \frac{1}{4}, \tag{2.2.33}$$

which proves that $C = -\frac{1}{4L^3\alpha^2}$.

To prove Equation 2.2.19 we first expand $\rho_2(\mathbf{x})$ and $\psi_2(\mathbf{x})$ in Fourier basis as

$$\rho_2(\mathbf{x}) = \sum_{\mathbf{n}} f(\mathbf{x} + \mathbf{n}L) - \frac{1}{L^3} = \sum_{\mathbf{k}} \hat{\rho}_2(\mathbf{k}) \mathrm{e}^{2\pi i \mathbf{k}\cdot\mathbf{x}/L}, \tag{2.2.34}$$

$$\psi_2(\mathbf{x}) = \sum_{\mathbf{k}} \hat{\psi}_2(\mathbf{k}) \mathrm{e}^{2\pi i \mathbf{k}\cdot\mathbf{x}/L}, \tag{2.2.35}$$

where $\hat{\rho}_2(\mathbf{k})$ and $\hat{\psi}_2(\mathbf{k})$ are the Fourier coefficients of $\rho_2(\mathbf{x})$ and $\psi_2(\mathbf{x})$ respectively. We will then find $\psi_2(\mathbf{x})$ expressed by the Fourier series in Equation 2.2.35.

We relate the $\hat{\psi}_2(\mathbf{k})$ to $\hat{\rho}_2(\mathbf{k})$, by taking a Fourier transform of Equation 2.2.19 which yields the relationship

$$\frac{4\pi^2|\mathbf{k}|^2}{L^2} \hat{\psi}_2(\mathbf{k}) = \hat{\rho}_2(\mathbf{k}). \tag{2.2.36}$$

From Equation 2.2.34, we observe that for $\mathbf{k} = 0$,

$$\hat{\rho}_2(0) = \frac{1}{L^3} \int_{cell} \rho_2(\mathbf{x}) d\mathbf{x} = 0, \tag{2.2.37}$$

because of the charge neutrality of $\rho_2(\mathbf{x})$ shown in Equation 2.2.16 to Equation 2.2.15.

Setting $\hat{\psi}_2(0) = 0$ enforces $\int_{\mathbb{R}^3} \psi_2(\mathbf{x}) dx = 0$, as demanded by Equation 2.2.21. We will now find $\hat{\rho}_2(\mathbf{k})$ from using (2.2.34) and combine the result with (2.2.36) to find $\hat{\psi}_2(\mathbf{k})$. We recall the definition of $\rho_2(\mathbf{x})$ from Equation 2.2.34 and with $\mathbf{k} \neq 0$, we

obtain

$$\hat{\rho}_2(\mathbf{k}) = \frac{1}{L^3}\int_{cell}\rho_2(\mathbf{x})\mathrm{e}^{-2\pi i\mathbf{k}\cdot\mathbf{x}/L}d\mathbf{x} \tag{2.2.38}$$

$$= \frac{1}{L^3}\int_{cell}\left(\sum_n f(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3}\right)\mathrm{e}^{-2\pi i\mathbf{k}\cdot\mathbf{x}/L}d\mathbf{x} \tag{2.2.39}$$

$$= \frac{1}{L^3}\int_{\mathbb{R}^3}\frac{\alpha^3}{\pi^{3/2}}\mathrm{e}^{-\alpha^2|\mathbf{x}|^2-2\pi i\mathbf{k}\cdot\mathbf{x}/L}d\mathbf{x} \tag{2.2.40}$$

$$= \frac{1}{L^3}\int_{\mathbb{R}^3}\frac{\alpha^3}{\pi^{3/2}}\mathrm{e}^{-\alpha^2\left|\mathbf{x}+\frac{\pi i\mathbf{k}}{L\alpha^2}\right|^2-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}d\mathbf{x} \tag{2.2.41}$$

$$= \frac{1}{L^3}\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}. \tag{2.2.42}$$

Then from Equation 2.2.36, we obtain

$$\hat{\psi}_2(\mathbf{k}) = \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2L|\mathbf{k}|^2}, \tag{2.2.43}$$

which together with Equation 2.2.35 yields Equation 2.2.23 and hence shows that $\psi_2(\mathbf{x})$ satisfies Equation 2.2.19.

Thus,

$$\psi(\mathbf{x}) = \sum_{\mathbf{n}}\frac{\mathrm{erfc}(\alpha\mid\mathbf{x}+\mathbf{n}L\mid)}{4\pi\mid\mathbf{x}+\mathbf{n}L\mid} - \frac{1}{4L^3\alpha^2} + \sum_{\mathbf{k}\neq 0}\frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}+2\pi i\mathbf{k}\cdot\mathbf{x}/L}}{4\pi^2L|\mathbf{k}|^2}, \tag{2.2.44}$$

is the potential function induced by $\rho(\mathbf{x})$ in Equation 2.2.5.

To find the potential $\tilde{\psi}(\mathbf{x})$ induced by the density $\tilde{\rho}(\mathbf{x})$ from Equation 2.2.6, we write

$$\begin{aligned}
\tilde{\rho}(\mathbf{x}) &= \rho(\mathbf{x}) - \delta(\mathbf{x}) = \rho_1(\mathbf{x}) + \rho_2(\mathbf{x}) - \delta(\mathbf{x}),\\
&= \sum_{\mathbf{n}}(\delta(\mathbf{x}+\mathbf{n}L) - f(\mathbf{x}+\mathbf{n}L)) + \sum_{\mathbf{n}} f(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3} - \delta(\mathbf{x})\\
&= \sum_{\mathbf{n}\neq 0}(\delta(\mathbf{x}+\mathbf{n}L) - f(\mathbf{x}+\mathbf{n}L)) + \sum_{\mathbf{n}} f(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3} + \delta(\mathbf{x}) - f(\mathbf{x}) - \delta(\mathbf{x})\\
&= \sum_{\mathbf{n}\neq 0}(\delta(\mathbf{x}+\mathbf{n}L) - f(\mathbf{x}+\mathbf{n}L)) + \sum_{\mathbf{n}} f(\mathbf{x}+\mathbf{n}L) - \frac{1}{L^3} - f(\mathbf{x})\\
&= \tilde{\rho}_1(\mathbf{x}) - f(\mathbf{x}).
\end{aligned} \tag{2.2.45}$$

Then, the potential induced by $\tilde{\rho}_1(\mathbf{x})$ is similar to the potential $\psi(\mathbf{x})$ from Equation 2.2.44 induced by $\rho(\mathbf{x})$ except that in this instance $\mathbf{n} \neq 0$. The potential $g(\mathbf{x})$ induced by the density $f(\mathbf{x})$ was shown in Equations 2.2.25 to 2.2.29 to be

$$g(\mathbf{x}) = \frac{\operatorname{erf}(\alpha|\mathbf{x}|)}{4\pi\mathbf{x}}. \tag{2.2.46}$$

Thus, from

$$\tilde{\psi}(\mathbf{x}) = \psi(\mathbf{x}) - g(\mathbf{x}), \tag{2.2.47}$$

we obtain,

$$\tilde{\psi}(\mathbf{x}) = \sum_{\mathbf{n}\neq 0} \frac{\operatorname{erfc}(\alpha|\mathbf{x}+\mathbf{n}L|)}{4\pi|\mathbf{x}+\mathbf{n}L|} - \frac{1}{4L^3\alpha^2} + \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{\frac{-\pi^2|\mathbf{k}|^2}{L^2\alpha^2}+2\pi i\mathbf{k}\cdot\mathbf{x}/L}}{4\pi^2 L|\mathbf{k}|^2} - \frac{\operatorname{erf}(\alpha|\mathbf{x}|)}{4\pi\mathbf{x}}. \tag{2.2.48}$$

Now since $i = j$, we set $\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j = 0$ and obtain

$$\tilde{\psi}(0) = \sum_{\mathbf{n}\neq 0} \frac{\operatorname{erfc}(\alpha|\mathbf{n}L|)}{4\pi|\mathbf{n}L|} - \frac{1}{4L^3\alpha^2} + \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{\frac{-\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L|\mathbf{k}|^2} - \lim_{|\mathbf{x}|\to 0} \frac{\operatorname{erf}(\alpha|\mathbf{x}|)}{4\pi\mathbf{x}}, \tag{2.2.49}$$

$$= \sum_{\mathbf{n}\neq 0} \frac{\operatorname{erfc}(\alpha|\mathbf{n}L|)}{4\pi|\mathbf{n}L|} - \frac{1}{4L^3\alpha^2} + \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{\frac{-\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L|\mathbf{k}|^2} - \frac{1}{4\pi}\frac{2}{\sqrt{\pi}}\alpha. \tag{2.2.50}$$

With $\psi(\mathbf{x})$ and $\tilde{\psi}(0)$ available from Equation 2.2.44, and Equation 2.2.50 respectively, and using the splitting from Equation 2.2.7, we obtain

$$V = \frac{1}{2}\sum_{i\neq j}^{N} q_i q_j \psi(x_i - x_j) + \frac{1}{2}\sum_{i=1}^{N} q_i^2 \tilde{\psi}(0) \tag{2.2.51}$$

$$= \frac{1}{2}\sum_{i\neq j}^{N} q_i q_j \left(\sum_{\mathbf{n}} \frac{\operatorname{erfc}(\alpha \mid \mathbf{x}+\mathbf{n}L \mid)}{4\pi \mid \mathbf{x}+\mathbf{n}L \mid} - \frac{1}{4L^3\alpha^2} + \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}+2\pi i\mathbf{k}\cdot\mathbf{x}/L}}{4\pi^2 L \mid \mathbf{k} \mid^2} \right) \tag{2.2.52}$$

$$+ \frac{1}{2}\sum_{i=1}^{N} q_i^2 \left(\sum_{\mathbf{n}\neq 0} \frac{\operatorname{erfc}(\alpha \mid \mathbf{n}L \mid)}{4\pi \mid \mathbf{n}L \mid} - \frac{1}{4L^3\alpha^2} + \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L \mid \mathbf{k} \mid^2} - \frac{2\alpha}{4\pi^{3/2}} \right).$$

This result can be simplified by noting that

$$\sum_{i\neq j}^{N} q_i q_j \frac{1}{4L^3\alpha^2} + \sum_{i=1}^{N} q_i^2 \frac{1}{4L^3\alpha^2} = \frac{1}{4L^3\alpha^2}\sum_{i=1}^{N} q_i \sum_{j=1}^{N} q_j = 0, \tag{2.2.53}$$

so we arrive at

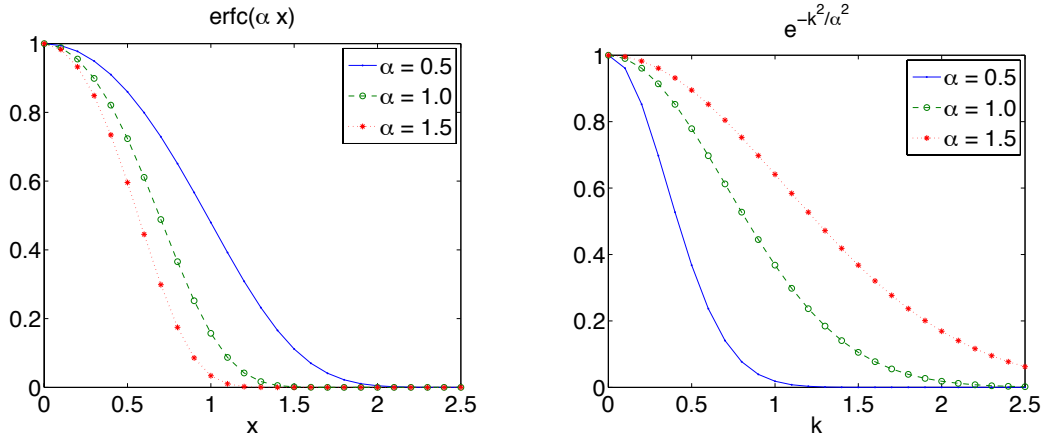
$$V = V^r + V^k + V^s, \tag{2.2.54}$$

with

$$V^r = \frac{1}{2}\sum_{\mathbf{n}}\sum_{i,j=1}^{N}{}' q_i q_j \frac{\operatorname{erfc}(\alpha \mid \mathbf{x}_i - \mathbf{x}_j + \mathbf{n}L \mid)}{4\pi \mid \mathbf{x}_i - \mathbf{x}_j + \mathbf{n}L \mid}, \tag{2.2.55}$$

$$V^k = \frac{1}{2}\sum_{i,j=1}^{N} q_i q_j \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2} + 2\pi i \mathbf{k}\cdot(\mathbf{x}_i - \mathbf{x}_j)/L}}{4\pi^2 L \mid \mathbf{k} \mid^2}, \tag{2.2.56}$$

$$V^s = \frac{-\alpha}{4\pi^{3/2}}\sum_{i=1}^{N} q_i^2. \tag{2.2.57}$$



(a) $\operatorname{erfc}(\alpha x)$ (b) $\mathrm{e}^{-\frac{k^2}{\alpha^2}}$

Figure 2.3: Figures (a), and (b) show $\operatorname{erfc}(\alpha x)$ and $\mathrm{e}^{-\frac{k^2}{\alpha^2}}$ respectively for different values of α. When the value of α increases, erfc decays faster whiles the decay rate of $\mathrm{e}^{-\frac{k^2}{\alpha^2}}$ slows down.

2.3(a) shows how $\operatorname{erfc}(\alpha x)$ varies with α. The larger the α value, the faster the decay rate of $\operatorname{erfc}(\alpha x)$. The decay rate of $\operatorname{erfc}(\alpha x)$ dominates and thus determines the rate of convergence of the real space sum V^r in Equation 2.2.55. As a result, we can adjust the rate of convergence of V^r by varying the value of α.

On the other hand, the effect of α on $\mathrm{e}^{-\frac{k^2}{\alpha^2}}$ is opposite of the effect on $\operatorname{erfc}(\alpha x)$. The larger the value of α, the slower the decay rate of $\mathrm{e}^{-\frac{k^2}{\alpha^2}}$. The decay rate of $\mathrm{e}^{-\frac{k^2}{\alpha^2}}$

also dominates and and hence determines the rate of convergence of the reciprocal space sum V^k in Equation 2.2.56. As such, the rate of convergence of V^k can be adjusted by varying the value of α.

Though V^r and V^k are both absolutely convergent, V^r decays faster with larger α while the rate of decay of V^k decreases with larger α. As such, for any choice of α value, we can determine the length scales for which contributions to V^r and V^k are significant. Then in computing the total potential, V, in Equation 2.2.54, the contributions to V^r or V^k of interactions between particles which are separated by a distance greater than the length scales for significant contributions can be ignored. Even though we can vary V^r, V^k and clearly V^s by varying the value of α, the total potential energy, V, is independent of the choice of $\alpha > 0$. This opposite behavior of the convergence rates of V^r and V^k as a function of α is at the crux of approaches for computing the Ewald Sum efficiently.

V^r and V^k are computed to a desired accuracy by specifying values for α, the real space cutoff, r_c, where $|\mathbf{x}_i - \mathbf{x}_j + \mathbf{n}L| < r_c$ and a reciprocal space cutoff k_c with $|\mathbf{k}| < k_c$. For N-body interactions, the cost of computing V in Equation 2.2.54 with cutoffs is $O(N^2)$. Perram et al. [41] have shown that this cost can be reduced to $O(N^{3/2})$ by a certain choice of α, r_c and k_c as functions of N. Kolafa and Perram [33] have derived and verified analytical error estimates in potential and forces for both V^r and V^k as a function of α, r_c and k_c.

Toukmaji and Board [57] have provided an overview of several approaches to computing the Ewald. One approach is to pick a large α value that makes it possible to compute V^r to a desired accuracy in $O(N)$ time using direct summation. This is because V^r decays quickly for large α values. However V^k decays slowly in this regime and the cost of computation is $O(N^2)$. Darden and co-workers [12, 21, 42]

developed the Particle Mesh Ewald (PME) method which employs interpolation and fast Fourier transforms to reduce the cost of computing V^k to $O(N\log N)$. Thus the overall cost of computing V is $O(N\log N)$.

Yet another way to compute V is to choose a small value of α. In this regime V^k can be evaluated at a cost of $O(N)$ to the desired accuracy, but the cost of evaluating V^r is $O(N^2)$. Duan and Krasny [16] developed the particle-cluster Cartesian treecode Ewald (CTE) algorithm which employs a three dimensional Taylor expansion and the hierarchical tree structure of Barnes and Hut [4] to reduce the cost of computing V^r to $O(N\log N)$. Hence, the overall cost is also $O(N\log N)$. The CTE method is a major focus of this thesis.

The next two sections will explain the PME method and CTE method, respectively.

2.3 Particle Mesh Ewald (PME) Method

The first version of PME with Lagrange interpolation is detailed in [12], while the present and most used version, also called smooth Particle Mesh Ewald (sPME) is described in [21]. The thesis' focus is on comparison with smooth PME which we simply refer to as PME. The exposition here is taken from several authors [12, 21, 57, 30, 53].

The Particle Mesh Ewald method is most efficient in the regime where most of the computational work is shifted onto the reciprocal space sum (large α). The work for the real space sum becomes $O(N)$ by specifying a value of r_c which together with the specified desired error, ϵ, in the real space sum, determines the value of α from the relation

$$\epsilon \approx \frac{\operatorname{erfc}(\alpha r_c)}{4\pi r_c}. \tag{2.3.1}$$

With α computed, then the relation

$$\epsilon \approx \frac{\mathrm{e}^{-\frac{\pi^2 k_c^2}{L^2\alpha^2}}}{4\pi^2 L k_c^2} \tag{2.3.2}$$

determines the reciprocal space cutoff k_c [53]. Petersen [42] derived analytical error estimates for the energy and forces when the reciprocal space sum is approximated by PME.

We now express V^k from Equation 2.2.56 in an equivalent form which is more convenient for computation,

$$V^k = \frac{1}{2}\sum_{i,j=1}^{N} q_i q_j \sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}+2\pi i\mathbf{k}\cdot(\mathbf{x}_i-\mathbf{x}_j)/L}}{4\pi^2 L|\ \mathbf{k}\ |^2}, \tag{2.3.3}$$

$$= \frac{1}{2}\sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L|\ \mathbf{k}\ |^2} \sum_{i,j=1}^{N} q_i q_j \mathrm{e}^{2\pi i\mathbf{k}\cdot(\mathbf{x}_i-\mathbf{x}_j)/L}, \tag{2.3.4}$$

$$= \frac{1}{2}\sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L|\ \mathbf{k}\ |^2} \sum_{i=1}^{N} q_i \mathrm{e}^{2\pi i\mathbf{k}\cdot\mathbf{x}_i/L} \sum_{j=1}^{N} q_j \mathrm{e}^{-2\pi i\mathbf{k}\cdot\mathbf{x}_j/L}, \tag{2.3.5}$$

$$= \frac{1}{2}\sum_{\mathbf{k}\neq 0} \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L|\ \mathbf{k}\ |^2} |S(\mathbf{k})|^2, \tag{2.3.6}$$

where,

$$S(\mathbf{k}) = \sum_{j=1}^{N} q_j \mathrm{e}^{2\pi i\mathbf{k}\cdot\mathbf{x}_j/L}, \tag{2.3.7}$$

is called the structure factor [57]. The main idea in PME is in approximating the structure factor, $S(\mathbf{k})$, on a rectangular grid that fills the simulation cell. To perform this approximation,

- The complex exponentials in Equation 2.3.7 are interpolated on to the grid using Euler exponential splines which involve Cardinal B-splines.
- Then the charge of each particle is also interpolated onto a grid, also with Cardinal B-splines to form a charge array.

- Finally $S(\mathbf{k})$ is approximated on the rectangular grid using the interpolation of the complex exponentials and the charges.

Before we present a detailed explanation of the three steps outlined above, we introduce the Cardinal B-spline.

The Cardinal B-splines form a basis for a vector space consisting of polynomial splines of a specified order with equally spaced knots [21]. The nth order Cardinal B-spline, $M_n(u)$, is defined by

$$M_n(u) = \frac{u}{n-1} M_{n-1}(u) + \frac{n-u}{n-1} M_{n-1}(u-1) \qquad \text{for } n > 2, \tag{2.3.8}$$

which is $n-2$ times differentiable and the derivative is given by

$$\frac{d}{du} M_n(u) = M_{n-1}(u) - M_{n-1}(u-1). \tag{2.3.9}$$

The starting function, $M_2(u)$ is defined as

(2.3.10)

$$M_2(u) = \begin{cases} 1 - |u-1| & \text{if } 0 \le u \le 2 \\ 0 & \text{if } u < 0 \quad \text{or} \quad u > 2 \end{cases}$$

which is shown in Figure 2.4 together with Cardinal B-splines up to $n = 7$. $M_n(u)$ has compact support with non-zero values in the interval $0 \le u \le n$.

In order to interpolate the complex exponentials on to a grid, the simulation cell is filled with a uniform grid with $K_1 \times K_2 \times K_3$ dimensions where $max\{K_1, K_2, K_3\} = k_c$. With the fundamental cell of length L centered at $(0,0,0)$, the fractional coordinates $\mathbf{x}_j/L = (x_{j_1}/L, x_{j_2}/L, x_{j_3}/L) = (f_{j_1}, f_{j_2}, f_{j_3})$ of each particle, j, in the definition of $S(\mathbf{k})$ in Equation 2.3.7 are scaled to give u_{ji} such that

$$u_{jm} = K_m \frac{x_{j_k}}{L}, \qquad \text{for } m = 1, 2, 3. \tag{2.3.11}$$

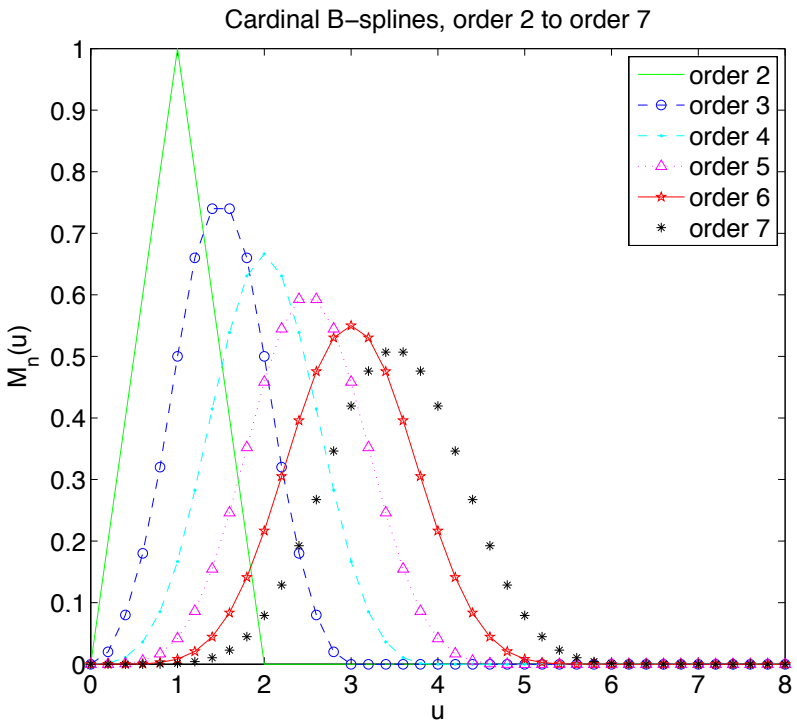


Figure 2.4: The B-splines $M_n(u)$ with non-zero values for $0 \leq u \leq n$.

Then the structure factor can be rewritten as

$$S(\mathbf{k}) = \sum_{j=1}^{N} q_j \mathrm{e}^{2\pi i\left(\frac{k_1}{K_1}u_{j1}+\frac{k_2}{K_2}u_{j2}+\frac{k_3}{K_3}u_{j3}\right)}. \tag{2.3.12}$$

Each complex exponential in the structure factor, $S(\mathbf{k})$, is approximated on the grid using an Euler exponential spline

$$e^{2\pi i\frac{k_j}{K_j}u_{jm}} \approx b_j(k_j) \sum_{l=-\infty}^{\infty} M_n(u_{jm}-l)e^{2\pi i\frac{k_j}{K_j}l}, \tag{2.3.13}$$

where $M_n(u)$ are the previously introduced Cardinal B-splines and the factor

$$b_j(k_j) = e^{2\pi i(n-1)k_j/K_j}\left[\sum_{l=0}^{n-2} M_n(l+1)e^{2\pi i k_j l/K_j}\right]^{-1}. \tag{2.3.14}$$

The Cardinal B-splines are evaluated at $u_{jm}-l$ and are thus dependent on K_m where $m \in \{1,2,3\}$. However, since $M_n(u)$ has compact support, the infinite sum in Equation 2.3.13 becomes a finite sum over the finite number of points, $u_{jm}-l$, for which $M_n(u)$ is non-zero.

The charge of each particle, j, with scaled fractional coordinate (u_{j1}, u_{j2}, u_{j3}) is then interpolated on to a grid which spans the simulation cell and its periodic images specified by (n_1, n_2, n_3), which usually are just the nearest images. The charge array, Q, from this interpolation is given by

(2.3.15)
$$Q(l_1,l_2,l_3) = \sum_{i=1}^{N} q_j \sum_{n_1,n_2,n_3} M_n(u_{j_1} - l_1 - n_1K_1) M_n(u_{j_2} - l_2 - n_2K_2) M_n(u_{j_3} - l_3 - n_3K_3),$$

where $u_{ji} - l_i - n_iK_i$ are evaluation points of the Cardinal B-spline on the grid that spans the fundamental cell and the periodic images.

With the complex exponentials and charges interpolated on their respective grids, the structure factor can then be approximated on the grid. Recall from Equation 2.3.12 that

$$S(\mathbf{k}) = \sum_{j=1}^{N} q_j \mathrm{e}^{2\pi i \frac{k_1}{K_1} u_{j_1}} \mathrm{e}^{2\pi i \frac{k_2}{K_2} u_{j2}} \mathrm{e}^{2\pi i \frac{k_1}{K_1} u_{j3}} \tag{2.3.16}$$

Then, using the approximation of the exponential splines in Equation 2.3.13, $S(\mathbf{k})$ is approximated as

$$\begin{aligned} S(\mathbf{k}) \approx \hat{S}(\mathbf{k}) &= b_1(k_1) b_2(k_2) b_3(k_3) \\ &\times \sum_{l_1,l_2,l_3} \sum_{j=1}^{N} q_j \sum_{n_1,n_2,n_3} M_n(u_{j_1} - l_1 - n_1K_1) M_n(u_{j_2} - l_2 - n_2K_2) \\ &\times M_n(u_{j_3} - l_3 - n_3K_3) e^{2\pi i \left(\frac{k_1}{K_1} l_1 + \frac{k_2}{K_2} l_2 + \frac{k_3}{K_3} l_3 \right)}, \end{aligned} \tag{2.3.17}$$

Thus, from Equation 2.3.15,

$$S(\mathbf{k}) = b_1(k_1) b_2(k_2) b_3(k_3) \sum_{l_1,l_2,l_3} Q(l_1,l_2,l_3) e^{2\pi i \left(\frac{k_1}{K_1} l_1 + \frac{k_2}{K_2} l_2 + \frac{k_3}{K_3} l_3 \right)}, \tag{2.3.18}$$

$$= b_1(k_1) b_2(k_2) b_3(k_3) \mathcal{F}(Q)(k_1,k_2,k_3). \tag{2.3.19}$$

where $\mathcal{F}(Q)(k_1,k_2,k_3)$ is the discrete Fourier transform of $Q(l_1,l_2,l_3)$. Then, from Equation 2.3.6, setting

$$B(k_1)B(k_2)B(k_3) = |b_1(k_1)|^2 |b_2(k_2)|^2 |b_3(k_3)|^2, \tag{2.3.20}$$

$$V^k \approx \frac{1}{2}\sum_{\mathbf{k}\neq 0}\frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L|\ \mathbf{k}\ |^2}B(k_1,k_2,k_3)\mathcal{F}(Q)(k_1,k_2,k_3)\mathcal{F}(Q)(-k_1,-k_2,-k_3), \tag{2.3.21}$$

$$= \frac{1}{2}\sum_{k_1=0}^{K_1-1}\sum_{k_2=0}^{K_2-1}\sum_{k_3=0}^{K_3-1}H(k_1,k_2,k_3)\mathcal{F}(Q)(k_1,k_2,k_3), \tag{2.3.22}$$

where

$$H(k_1,k_2,k_3) = \frac{\mathrm{e}^{-\frac{\pi^2|\mathbf{k}|^2}{L^2\alpha^2}}}{4\pi^2 L|\ \mathbf{k}\ |^2}B(k_1,k_2,k_3)\mathcal{F}^*(Q)(k_1,k_2,k_3), \tag{2.3.23}$$

and $\mathcal{F}^*(Q)(k_1,k_2,k_3) = \mathcal{F}(Q)(-k_1,-k_2,-k_3)$ is the complex conjugate of $\mathcal{F}(Q)(k_1,k_2,k_3)$. The gradient of V^k from Equation 2.3.22 will require three Fourier transforms, one for each coordinate, so we will rewrite V^k in a form which requires no Fourier transform upon taking the gradient.

We note that the discrete Fourier transform, $\mathcal{F}(Q)(k_1,k_2,k_3)$ is defined as

$$\mathcal{F}(Q)(k_1,k_2,k_3) = \sum_{l_1=0}^{K_1-1}\sum_{l_2=0}^{K_2-1}\sum_{l_3=0}^{K_3-1}Q(l_1,l_2,l_3)e^{2\pi i\left(\frac{k_1}{K_1}l_1+\frac{k_2}{K_2}l_2+\frac{k_3}{K_3}l_3\right)}. \tag{2.3.24}$$

Then, from Equation 2.3.22 and using the definition in Equation 2.3.24, V^k can be rewritten as

$$\begin{aligned} V^k &\approx \frac{1}{2}\sum_{k_1=0}^{K_1-1}\sum_{k_2=0}^{K_2-1}\sum_{k_3=0}^{K_3-1}H(k_1,k_2,k_3)\sum_{l_1=0}^{K_1-1}\sum_{l_2=0}^{K_2-1}\sum_{l_3=0}^{K_3-1}Q(l_1,l_2,l_3)e^{2\pi i\left(\frac{k_1}{K_1}l_1+\frac{k_2}{K_2}l_2+\frac{k_3}{K_3}l_3\right)} \\ &= \frac{1}{2}\sum_{l_1=0}^{K_1-1}\sum_{l_2=0}^{K_2-1}\sum_{l_3=0}^{K_3-1}\sum_{k_1=0}^{K_1-1}\sum_{k_2=0}^{K_2-1}\sum_{k_3=0}^{K_3-1}H(k_1,k_2,k_3)e^{2\pi i\left(\frac{l_1}{K_1}k_1+\frac{l_2}{K_2}k_2+\frac{l_3}{K_3}k_3\right)}Q(l_1,l_2,l_3) \end{aligned}$$

where in the last equation the inner sum over (k_1,k_2,k_3) defines $\mathcal{F}(H)$, the discrete Fourier transform of H. V^k has thus been rewritten as

$$V^k \approx \frac{1}{2}\sum_{k_1=0}^{K_1-1}\sum_{k_2=0}^{K_2-1}\sum_{k_3=0}^{K_3-1}\mathcal{F}(H)(k_1,k_2,k_3)Q(k_1,k_2,k_3), \tag{2.3.25}$$

and the force from the reciprocal sum on the *jth* particle is approximated by

$$\mathbf{f}_j = -\nabla_{\mathbf{x}_j}V^k \approx -\frac{1}{2}\sum_{k_1=0}^{K_1-1}\sum_{k_2=0}^{K_2-1}\sum_{k_3=0}^{K_3-1}\mathcal{F}(H)(k_1,k_2,k_3)\nabla_{\mathbf{x}_j}Q(k_1,k_2,k_3). \tag{2.3.26}$$

Hence, approximating the reciprocal space interactions with PME consists of constructing the charge array $Q(k_1, k_2, k_3)$, finding $\mathcal{F}(H)(k_1, k_2, k_3)$ and applying equations Equation 2.3.25 and Equation 2.3.26.

The accuracy of PME improves with increasing number of grid points and increasing order of the interpolation. Since PME employs fast Fourier transforms, the algorithm works best for uniform particle distributions with a decrease in efficiency for non-uniform distributions [27]. Also, the fact that $M_n(u)$ is $n-2$ times continuously differentiable means that to be able to compute the forces using Equation 2.3.26, the interpolation must be at least order 3.

2.4 Cartesian Treecode Ewald (CTE) Method

Treecode algorithms have been applied to several three dimensional N-body problems in astrophysics [48, 20], fluid-dynamics [37, 38] and molecular dynamics [6, 7] to cut the $O(N^2)$ cost to $O(N\log N)$. The basic idea, first introduced by Pincus and Scheraga [43], is the recognition that particle interactions can be split into near and far-field interactions. In a system of interacting particles, a majority of a particle's interaction can be designated as far-field. A speed-up in computation time is achieved by evaluating these far-field interactions approximately and computing the near-field interactions exactly. For distance-dependent potentials, like the Coulomb potential, for which the effect of other particles on a target cluster falls off with increasing distance, the approximation of the designated far-field interactions can be made to achieve any required level of accuracy. The designated near-field interactions are computed exactly.

To determine a criterion for deciding whether a particular interaction is near or far-field, methods based on a hierarchical clustering of the system of particles

were developed by Barnes and Hut [4, 5], Appel [3], and Greengard and Rokhlin [26, 11]. The algorithms of Barnes and Hut [4, 5] and Appel [3] have been shown to have complexity $O(N\log N)$ and are much simpler to code than the Fast Multipole method (FMM)[26, 11] of Greengard and Rokhlin which is theoretically $O(N)$. Ding et al. [14, 15] have also presented the Cell Multipole Method (CMM) which is also claimed to be $O(N)$. The Parallel Multipole Treecode Algorithm (PMTA), a hybrid of Barnes-Hut and FMM was developed by Board et al. [6, 7]. This thesis will present another hybrid of Barnes-Hut, Appel and FMM in Chapter IV.

The particle-cluster Cartesian Treecode Ewald method first introduced in [16] is a Barnes-Hut type algorithm. Similar algorithms for the Coulomb [17, 18] and the screened Coulomb [36] potential have also been developed. All of these were preceded and inspired by a particle-cluster treecode developed for vortex sheet motion [37, 38]. Error analysis of several variants of the Cartesian treecode algorithm are given in [37, 38, 47].

As pointed out in the introduction, the CTE method is designed for a variant of the Ewald sum where the bulk of the computational work has been shifted on to V^r, the real space. CTE reduces the time to compute V^r from $O(N^2)$ to $O(N\log N)$. The reduction in computing time is achieved by

- Restructuring the system into a hierarchical tree of clusters of particles to perform particle-cluster interactions,
- Approximating the interactions between a target particle and a distant cluster of particles by a three dimensional Taylor expansion,
- Fast recurrence relations to quickly obtain the Taylor coefficients needed for the expansion.

We will now give a discussion of the three processes outlined above.

2.4.1 The Hierarchical Tree: Construction and Characteristics

The process of constructing the tree is independent of the particle distribution. However, in this thesis, we apply the Cartesian treecode algorithm for uniformly distributed liquid in a cubic box, and all descriptions in this thesis refer to a cubic box.

The construction of a two-dimensional hierarchical tree is depicted in Figure 2.5. In three dimensions the cubic box of particles is recursively divided into a hierarchical

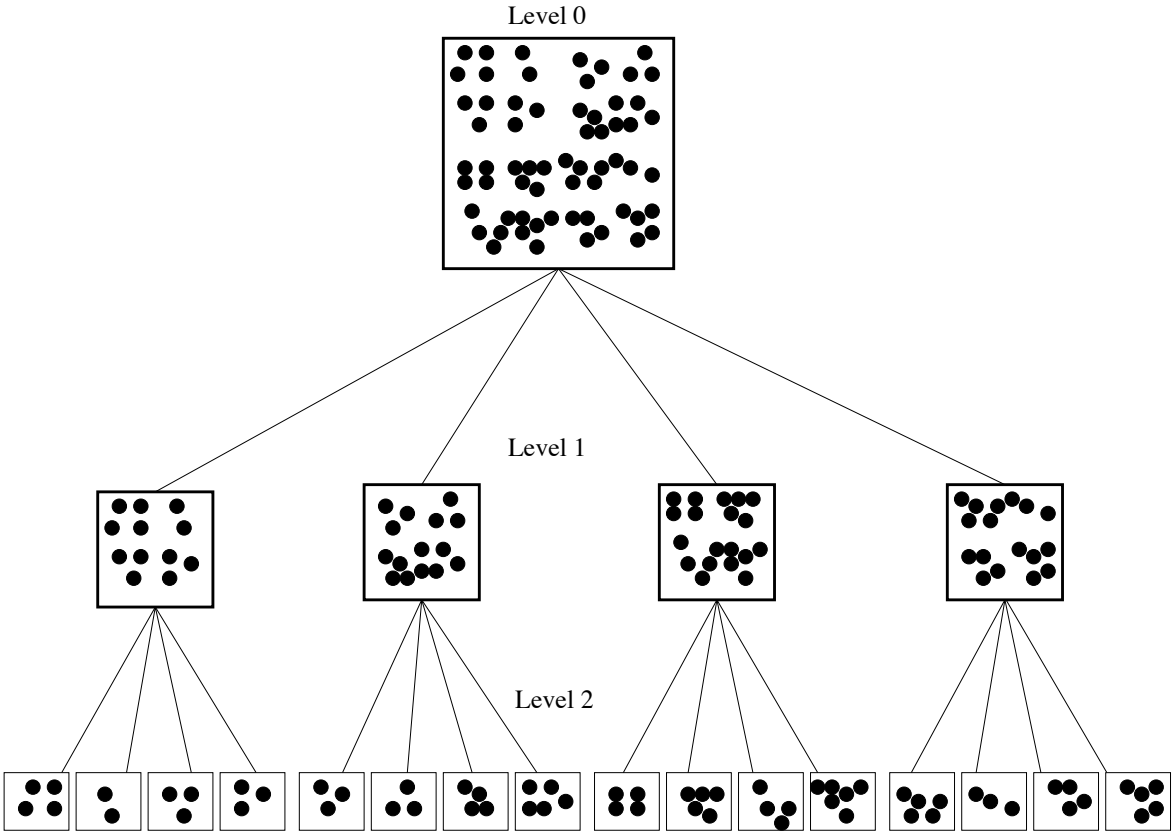


Figure 2.5: Hierarchical clustering to create a tree. The division starts at level zero and continues for each branch until the number of particles in a leaf is less or equal to a preassigned number, N_o. Alternatively we can halt the division of a branch when a preassigned level is attained. In this figure, the division stops at level 2.

oct-tree of clusters of particles. The original cubic box is the root or level 0 of the

tree. In three-dimensions the root is first divided into eight children which form level 1. Each of these eight children are then further divided into eight smaller children to form level 2. The recursive division continues until a preassigned number of levels is attained. Alternatively, the division can proceed until the number of particles in a child is less or or equal to a preassigned number, N_o. A cluster at the deepest or last level of each branch of a tree is called a leaf. Each cluster or node stores its geometric center, $\mathbf{x}_c$, radius, r_c and moments, $M_C^{\mathbf{k}}$, for order $\mathbf{k} = 0$ up to order $\mathbf{k} = p$, where p is the order of the multipole approximation. We will define the moments in the next section.

The next paragraph gives an overview of the procedure to compute the force on a particle. An exhaustive description of this procedure with pseudocode is given in [16].

To compute the force on the particles, the tree is traversed downward for each target particle, i, by a call to a recursive subroutine. At each level of the tree, the algorithm checks whether a cluster on that level is well separated from the target particle. A cluster is well separated from a particle if $\frac{r}{R} \leq \theta$, where r is the radius of the cluster, R is the distance from the geometric center of the cluster to the particle, and $\theta < 1$ is a preassigned parameter. We refer to the condition $\frac{r}{R} \leq \theta$ as the multipole acceptance criterion. If a cluster is well separated from the particle, the force on the particle due to the cluster is approximated using the particle-cluster Taylor approximation, else the algorithm descends to the children of the cluster and applies the same criterion. If a cluster turns out to be a leaf, i.e. on the last level, then the force on the target particle, i, due to the cluster, C, is computed by direct summation.

Figure 2.6 shows an interaction between a particle and a cluster.

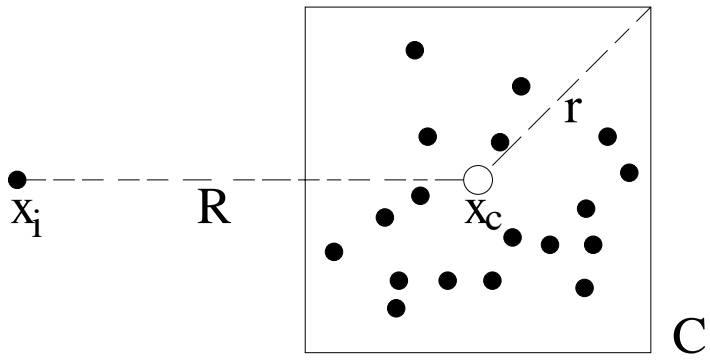


Figure 2.6: The particle at $\mathbf{x}_i$ interacts with cluster C if the multipole acceptability criterion, i.e. $\frac{r}{R} \leq \theta$ is satisfied.

A small θ value forces the algorithm to descend deeper into the tree which results in higher accuracy but longer computation times. As with all numerical methods, a balance between accuracy and speed is essential.

2.4.2 Multipole Approximation

The three dimensional pth order Taylor approximation for the real space potential energy, $V^r_{i,C}$, and the force, $\mathbf{F}^r_{i,C}$, on particle i, due to cluster C as shown in Figure 2.6 are generated below. The particle positions, r_i, r_j are scaled by α such that $\mathbf{x}_i = \alpha\mathbf{r_i}$ and $\mathbf{x}_j = \alpha\mathbf{r_j}$. The real space potential

$$\phi(\mathbf{x}) = \frac{\sqrt{\pi}}{2}\frac{\operatorname{erfc}(|\mathbf{x}|)}{|\mathbf{x}|} \tag{2.4.1}$$

from Equation 2.2.55 is expanded about the center $\mathbf{x}_c = (x_{c_1}, x_{c_2}, x_{c_3})$ of cluster C. The potential energy $V^r_{i,C}$ at position $\mathbf{x}_i$ due to cluster C, scaled by $4\pi\epsilon_o$ then follows

as,

$$
\begin{aligned}
V^r_{i,C} &= \sum_{j\in C} q_i q_j \frac{\operatorname{erfc}(\alpha|\mathbf{r_i}-\mathbf{r_j}|)}{|\mathbf{r_i}-\mathbf{r_j}|} && (2.4.2)\\
&= \frac{2\alpha}{\sqrt{\pi}} \sum_{j\in C} q_i q_j \phi(\mathbf{x}_i-\mathbf{x}_j)\\
&\approx \frac{2\alpha}{\sqrt{\pi}} \sum_{j\in C} q_i q_j \sum_{||\mathbf{k}||\neq 0}^{p} \frac{1}{\mathbf{k}!}\mathbf{D}^{\mathbf{k}}_{\mathbf{x}}\phi(\mathbf{x}_i-\mathbf{x}_c)(\mathbf{x}_j-\mathbf{x}_c)^{\mathbf{k}}\\
&= \frac{2\alpha}{\sqrt{\pi}} q_i \sum_{||\mathbf{k}||\neq 0}^{p} \frac{1}{\mathbf{k}!}\mathbf{D}^{\mathbf{k}}_{\mathbf{x}}\phi(\mathbf{x}_i-\mathbf{x}_c) \sum_{j\in C} q_j(\mathbf{x}_j-\mathbf{x}_c)^{\mathbf{k}}\\
&= \frac{2\alpha}{\sqrt{\pi}} q_i \sum_{||\mathbf{k}||\neq 0}^{p} a_{\mathbf{k}} M^{\mathbf{k}}_C, && (2.4.3)
\end{aligned}
$$

where the vector $\mathbf{k}$ is given as

$$\mathbf{k} = (k_1,k_2,k_3), \qquad \text{and} \tag{2.4.4}$$

$$\mathbf{k}! = k_1!k_2!k_3!, \tag{2.4.5}$$

$$||\mathbf{k}|| = k_1+k_2+k_3. \tag{2.4.6}$$

The operator $\mathbf{D}^{\mathbf{k}}_{\mathbf{x}}$ is a multi-dimensional derivative given as

$$\mathbf{D}^{\mathbf{k}}_{\mathbf{x}} = \frac{\partial^{||\mathbf{k}||}}{\partial^{k_1}_{x_1}\partial^{k_2}_{x_2}\partial^{k_3}_{x_3}} = \mathbf{D}^{k_1}_{x_1}\mathbf{D}^{k_2}_{x_2}\mathbf{D}^{k_3}_{x_3} \tag{2.4.7}$$

while

$$(\mathbf{x}_j-\mathbf{x}_c)^{\mathbf{k}} = (x_{j_1}-x_{c_1})^{k_1}(x_{j_2}-x_{c_2})^{k_2}(x_{j_3}-x_{c_3})^{k_3}. \tag{2.4.8}$$

The coefficients $a_{\mathbf{k}}$ are the $\mathbf{k}$th order Taylor coefficients of the potential given by

$$a_{\mathbf{k}} = \frac{1}{\mathbf{k}!}\mathbf{D}^{\mathbf{k}}_{\mathbf{x}}\phi(\mathbf{x}_i-\mathbf{x}_c), \tag{2.4.9}$$

and $M^{\mathbf{k}}_C$ is the $\mathbf{k}$th multipole moment of cluster C defined as

$$M^{\mathbf{k}}_C = \sum_{j\in C} q_j(\mathbf{x}_j-\mathbf{x}_c)^{\mathbf{k}}. \tag{2.4.10}$$

The cluster moments depend only on the particles in the cluster, thus $M_C^{\mathbf{k}}$ can be computed and stored when the tree is built. The force $\mathbf{F}_{i,C}^r$ on particle i at position $\mathbf{x}_i$ due to cluster C is obtained as the gradient of the potential energy $V_{i,C}^r$ from Equation 2.4.3 and is given by,

$$\mathbf{F}_{i,C}^r = -\nabla_{x_i} V_{i,C}^r(\mathbf{x}_i - \mathbf{x}_j) \tag{2.4.11}$$

$$\approx -\frac{2\alpha^2}{\sqrt{\pi}} q_i \sum_{||\mathbf{k}||=0}^{p} (\nabla_{x_i} a_{\mathbf{k}}) M_C^{\mathbf{k}} \tag{2.4.12}$$

$$= -\frac{2\alpha^2}{\sqrt{\pi}} q_i \sum_{||\mathbf{k}||=0}^{p} \begin{pmatrix} (k_1+1) a_{\mathbf{k}+\mathbf{e_1}} \\ (k_2+1) a_{\mathbf{k}+\mathbf{e_2}} \\ (k_3+1) a_{\mathbf{k}+\mathbf{e_3}} \end{pmatrix} M_C^{\mathbf{k}}, \tag{2.4.13}$$

with $\mathbf{e_1} =< 1,0,0 >$, $\mathbf{e_2} =< 0,1,0 >$ and $\mathbf{e_3} =< 0,0,1 >$.

A check on the convergence of the Ewald sum is to compute the electrostatic contribution to the virial, V_s, of the system defined as

$$V_s = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \mathbf{F}_{ij} \cdot \mathbf{r}_{ij}, \tag{2.4.14}$$

where $\mathbf{F}_{ij}$ is the force on particle i due to particle j and $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$.When the Ewald sum has converged, the relationship

$$V_s = -V, \tag{2.4.15}$$

holds [30]. That is, the total Coulomb potential energy, V, is the negative of the system virial, V_s. The total system virial , V_s is as expected made up of contributions from a real space virial, V_s^r and reciprocal space virial V_s^k such that

$$V_s = V_s^r + V_s^k. \tag{2.4.16}$$

In the treecode algorithm, when the multipole acceptance criterion is satisfied for a particle-cluster pair, the virial contribution, V_s^r of the real space potential energy

V^r is computed with the approximation $\mathbf{F}^r_{i,C}$ derived in Equation 2.4.13 otherwise $\mathbf{F}^r_i$ is computed directly. Then from Equation 2.4.14, V^r_s, due to the real space potential energy is computed as

(2.4.17)
$$V^r_s = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in C}\mathbf{F}^r_{i,C}\cdot\mathbf{r}_{ij} \approx -\frac{\alpha^2}{\sqrt{\pi}}\sum_{C}\sum_{i=1}^{N} q_i\mathbf{r}_{i,C}\cdot\sum_{||\mathbf{k}||=0}^{p}\begin{pmatrix}(k_1+1)a_{\mathbf{k}+\mathbf{e_1}}\\(k_2+1)a_{\mathbf{k}+\mathbf{e_2}}\\(k_3+1)a_{\mathbf{k}+\mathbf{e_3}}\end{pmatrix} M^{\mathbf{k}}_C,$$

where $\mathbf{r}_{i,C}$ is the vector from the target particle to the center of cluster C.

In the reciprocal space, V^k_s is computed as

(2.4.18)
$$V^k_s = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in C}\mathbf{F}^k_i\cdot\mathbf{r}_{ij},$$

where

(2.4.19)
$$\mathbf{F}^k_i = -\nabla_{\mathbf{x}_i}V^k$$

and V^k is defined in Equation 2.2.56.

2.4.3 Recurrence Relation

The recurrence relation

(2.4.20)
$$a_{\mathbf{k}} = \frac{1}{|x|^2}\left\{\left(\frac{1}{||\mathbf{k}||}-2\right)\sum_{i=1}^{3}x_i a_{\mathbf{k}-\mathbf{e_i}} - \left(1-\frac{1}{||\mathbf{k}||}\right)\sum_{i=1}^{3}a_{\mathbf{k}-\mathbf{2e_i}} + b_{\mathbf{k}}\right\}$$

is employed in computing the Taylor coefficients $a_{\mathbf{k}}$ defined in Equation 2.4.9 where the $b_{\mathbf{k}}$ terms are the Taylor coefficients of the function

(2.4.21)
$$\psi(\mathbf{x}) = \frac{1}{2}\mathrm{e}^{-|\mathbf{x}|^2}.$$

The derivation of the above reccurrence is outlined in [16] but we will give a full derivation here. We note that the Taylor coefficients $a_{\mathbf{k}}$ are those of the solution to

the Poisson equation with a source term composed of a point charge screened by a Gaussian. The recurrence relation for $a_{\mathbf{k}}$ provides a convenient way to compute the Taylor coefficients and the derivatives of the solution to the Poisson equation

$$-\Delta\phi(\mathbf{x}) = \rho(\mathbf{x}), \tag{2.4.22}$$

where

$$\rho(\mathbf{x}) = \delta(\mathbf{x}) - \frac{\alpha^3}{\pi^{3/2}} e^{-\alpha^2|\mathbf{x}|^2} \tag{2.4.23}$$

and

$$\phi(\mathbf{x}) = \frac{\sqrt{\pi}}{2} \frac{\operatorname{erfc}(|\mathbf{x}|)}{|\mathbf{x}|}. \tag{2.4.24}$$

To find the recurrence relation for $a_{\mathbf{k}}$ we will first find a recurrence relation for the Taylor coefficients $b_{\mathbf{k}}$ of $\psi(\mathbf{x})$. Let $\mathbf{x} = \mathbf{x}_i - \mathbf{x}_c$ and

$$\psi(\mathbf{x}) = \frac{1}{2}\mathrm{e}^{-|\mathbf{x}|^2} = \mathrm{e}^{-(x_1^2+x_2^2+x_3^2)}. \tag{2.4.25}$$

With the definition of $\psi(\mathbf{x})$ in Equation 2.4.25, the identity

$$\frac{\partial\psi(\mathbf{x})}{\partial x_i} + 2x_i\psi(\mathbf{x}) = 0, \qquad \text{for } i = 1, 2, 3 \tag{2.4.26}$$

holds. We apply $\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}$ to Equation 2.4.26, and using Leibniz's rule for differentiating a product we get

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\left\{\frac{\partial\psi(\mathbf{x})}{\partial x_i}\right\} = \mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}) \tag{2.4.27}$$

for the first term and,

$$\begin{aligned}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\left[2x_i\psi(\mathbf{x})\right] &= 2x_i\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\psi(\mathbf{x}) + \binom{k_i-1}{1}\frac{\partial(2x_i)}{\partial x_i}\cdot\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\psi(\mathbf{x}) \\ &= 2x_i\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\psi(\mathbf{x}) + 2(k_i-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\psi(\mathbf{x}),\end{aligned} \tag{2.4.28}$$

for the second term. When we put Equation 2.4.27 and Equation 2.4.28 together, we get

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}) + 2x_i\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\psi(\mathbf{x}) + 2(k_i-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\psi(\mathbf{x}) = 0, \qquad \text{for } i = 1,2,3. \tag{2.4.29}$$

Multiplying Equation 2.4.29 by $\frac{1}{\mathbf{k}!}$ yields,

$$\frac{1}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}) + \frac{2}{\mathbf{k}!}x_i\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\psi(\mathbf{x}) + \frac{2(k_i-1)}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\psi(\mathbf{x}) = 0, \tag{2.4.30}$$

The coefficients of the kth order term in the Taylor expansion of $\psi(\mathbf{x})$ are defined as

$$b_{\mathbf{k}} = \frac{1}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}), \tag{2.4.31}$$

and as a result, the identities

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\psi(\mathbf{x}) = b_{\mathbf{k}-\mathbf{e}_i}\cdot(\mathbf{k}-\mathbf{e}_i)! \qquad \text{and} \tag{2.4.32}$$

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\psi(\mathbf{x}) = b_{\mathbf{k}-2\mathbf{e}_i}\cdot(\mathbf{k}-2\mathbf{e}_i)! \tag{2.4.33}$$

hold. Hence, Equation 2.4.30 becomes

$$b_{\mathbf{k}} + \frac{2x_i}{k_i}b_{\mathbf{k}-\mathbf{e}_i} + \frac{2}{k_i}b_{\mathbf{k}-2\mathbf{e}_i} = 0, \qquad \text{for } i = 1,2,3, \tag{2.4.34}$$

where $b_{\mathbf{k}} = 0$ when any one of k_1, k_2 or k_3 is negative. If we write Equation 2.4.34 as

$$k_i b_{\mathbf{k}} + 2x_i b_{\mathbf{k}-\mathbf{e}_i} + 2b_{\mathbf{k}-2\mathbf{e}_i} = 0, \tag{2.4.35}$$

and enumerate for $i = 1,2,3$, then

$$k_1 b_{\mathbf{k}} + 2x_1 b_{\mathbf{k}-\mathbf{e}_1} + 2b_{\mathbf{k}-2\mathbf{e}_1} = 0, \tag{2.4.36}$$

$$k_2 b_{\mathbf{k}} + 2x_2 b_{\mathbf{k}-\mathbf{e}_2} + 2b_{\mathbf{k}-2\mathbf{e}_2} = 0, \tag{2.4.37}$$

$$k_3 b_{\mathbf{k}} + 2x_3 b_{\mathbf{k}-\mathbf{e}_3} + 2b_{\mathbf{k}-2\mathbf{e}_3} = 0. \tag{2.4.38}$$

The sum of Equations 2.4.36 - 2.4.38 results in

$$(k_1+k_2+k_3)b_{\mathbf{k}} = -2\sum_{i=1}^{3} x_i b_{\mathbf{k}-\mathbf{e}_i} - \sum_{i=1}^{3} b_{\mathbf{k}-2\mathbf{e}_i}, \tag{2.4.39}$$

which can be written as

$$b_{\mathbf{k}} = -\frac{2}{||\mathbf{k}||}\sum_{i=1}^{3}\left\{x_i b_{\mathbf{k}-\mathbf{e}_i} + b_{\mathbf{k}-2\mathbf{e}_i}\right\}. \tag{2.4.40}$$

Next we will find the recurrence relation for the Taylor coefficients in the three-dimensional Taylor expansion of $\phi(\mathbf{x})$, where

$$\phi(\mathbf{x}) = \frac{\sqrt{\pi}}{2}\frac{\operatorname{erfc}(|\mathbf{x}|)}{|\mathbf{x}|} = \frac{\sqrt{\pi}}{2|\mathbf{x}|}\left\{\frac{2}{\sqrt{\pi}}\int_{|\mathbf{x}|}^{\infty} \mathrm{e}^{-s^2} ds\right\} = \frac{1}{|\mathbf{x}|}\int_{|\mathbf{x}|}^{\infty} \mathrm{e}^{-s^2} ds. \tag{2.4.41}$$

The partial derivative of $\phi(\mathbf{x})$ with respect to x_i is

$$\frac{\partial\phi(\mathbf{x})}{\partial x_i} = \frac{1}{|\mathbf{x}|^2}\left\{-|\mathbf{x}|\cdot\left(\mathrm{e}^{-|\mathbf{x}|^2}\right)\cdot\left(\frac{x_i}{|\mathbf{x}|}\right) - \frac{x_i}{|\mathbf{x}|}\int_{|\mathbf{x}|}^{\infty}\mathrm{e}^{-s^2}ds\right\} \tag{2.4.42}$$

$$= \frac{x_i}{|\mathbf{x}|^2}\left\{-\mathrm{e}^{-|\mathbf{x}|^2} - \frac{1}{|\mathbf{x}|}\int_{|\mathbf{x}|}^{\infty}\mathrm{e}^{-s^2}ds\right\} \tag{2.4.43}$$

$$= \frac{x_i}{|\mathbf{x}|^2}\left\{-2\psi(\mathbf{x}) - \phi(\mathbf{x})\right\}. \tag{2.4.44}$$

By using Equation 2.4.26, Equation 2.4.44 becomes

$$|\mathbf{x}|^2\frac{\partial\phi(\mathbf{x})}{\partial x_i} + x_i\phi(\mathbf{x}) = \frac{\partial\psi(\mathbf{x})}{\partial x_i}. \tag{2.4.45}$$

We note that

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1} = \mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2}\mathbf{D}_{x_1}^{k_1-1}, \tag{2.4.46}$$

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_2} = \mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2-1}\mathbf{D}_{x_1}^{k_1}, \tag{2.4.47}$$

and

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_3} = \mathbf{D}_{x_3}^{k_3-1}\mathbf{D}_{x_2}^{k_2}\mathbf{D}_{x_1}^{k_1}. \tag{2.4.48}$$

Then $\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}$ is applied to (2.4.45) for each $i = 1, 2, 3$ to give

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\left\{|\mathbf{x}|^2\frac{\partial\phi(\mathbf{x})}{\partial x_i}\right\} + \mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\left\{x_i\phi(\mathbf{x})\right\} = \mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\left\{\frac{\partial\psi(\mathbf{x})}{\partial x_i}\right\} = \mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}). \tag{2.4.49}$$

We will now compute Equation 2.4.49 for $i = 1$. The following procedure can be done for $i = 2$ and $i = 3$. We will appeal to this argument later on to complete the derivation. First, we compute the first term of Equation 2.4.49 for $i = 1$.

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\left\{|\mathbf{x}|^2\frac{\partial\phi(\mathbf{x})}{\partial x_1}\right\} = \mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2}\mathbf{D}_{x_1}^{k_1-1}\left\{|\mathbf{x}|^2\frac{\partial\phi(\mathbf{x})}{\partial x_1}\right\}. \tag{2.4.50}$$

The action of the first operator $\mathbf{D}_{x_1}^{k_1-1}$ is given by Leibniz's rule for differentiation of products as

$$\mathbf{D}_{x_1}^{k_1-1}\left\{|\mathbf{x}|^2\frac{\partial\phi(\mathbf{x})}{\partial x_1}\right\} = \tag{2.4.51}$$

$$\begin{aligned} &= |\mathbf{x}|^2\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x}) + \binom{k_1-1}{1}\frac{\partial|\mathbf{x}|^2}{\partial x_1}\mathbf{D}_{x_1}^{k_1-2}\left(\frac{\partial\phi(\mathbf{x})}{\partial x_1}\right) + \binom{k_1-1}{2}\frac{\partial^2|\mathbf{x}|^2}{\partial^2 x_1}\mathbf{D}_{x_1}^{k_1-3}\left(\frac{\partial\phi(\mathbf{x})}{\partial x_1}\right) \\ &= |\mathbf{x}|^2\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x}) + 2(k_1-1)x_1\mathbf{D}_{x_1}^{k_1-1}\phi(\mathbf{x}) + 2\left\{\frac{(k_1-1)(k_1-2)}{2}\right\}\mathbf{D}_{x_1}^{k_1-2}\phi(\mathbf{x}). \end{aligned}$$

The action of the operator $\mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2}$ on the resultant in Equation 2.4.51 is then given as

$$\begin{aligned} \mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2}\mathbf{D}_{x_1}^{k_1-1}\left\{|\mathbf{x}|^2\frac{\partial\phi(\mathbf{x})}{\partial x_1}\right\} &= \mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2}\left\{|\mathbf{x}|^2\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x})\right\} \\ &+ 2(k_1-1)x_1\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\phi(\mathbf{x}) + (k_1-1)(k_1-2)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}\phi(\mathbf{x}). \end{aligned} \tag{2.4.52}$$

The first term of Equation 2.4.52 when expanded yields

$$\begin{aligned} &\mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2}\left\{|\mathbf{x}|^2\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x})\right\} = \mathbf{D}_{x_3}^{k_3}|\mathbf{x}|^2\mathbf{D}_{x_2}^{k_2}\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x}) \\ &+ \binom{k_2}{1}\frac{\partial|\mathbf{x}|^2}{\partial x_2}\mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2-1}\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x}) + \binom{k_2}{2}\frac{\partial^2|\mathbf{x}|^2}{\partial^2 x_2}\mathbf{D}_{x_3}^{k_3}\mathbf{D}_{x_2}^{k_2-2}\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x}) \\ &= |\mathbf{x}|^2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\phi(\mathbf{x}) + \binom{k_3}{1}\frac{\partial|\mathbf{x}|^2}{\partial x_3}\mathbf{D}_{x_3}^{k_3-1}\mathbf{D}_{x_2}^{k_2}\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x}) \\ &+ \binom{k_3}{2}\frac{\partial^2|\mathbf{x}|^2}{\partial^2 x_3}\mathbf{D}_{x_3}^{k_3-2}\mathbf{D}_{x_2}^{k_2}\mathbf{D}_{x_1}^{k_1}\phi(\mathbf{x}) + 2k_2x_2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_2}\phi(\mathbf{x}) + k_2(k_2-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_2}\phi(\mathbf{x}) \\ &= |\mathbf{x}|^2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\phi(\mathbf{x}) + 2k_3x_3\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_3}\phi(\mathbf{x}) + k_3(k_3-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_3}\phi(\mathbf{x}) \\ &+ 2k_2x_2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_2}\phi(\mathbf{x}) + k_2(k_2-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_2}\phi(\mathbf{x}). \end{aligned} \tag{2.4.53}$$

Now, going back to compute the second term of Equation 2.4.49 yields

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1} x_1\phi(\mathbf{x}) = x_1\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\phi(\mathbf{x}) + \binom{k_1-1}{1}\frac{\partial x_1}{\partial x_1}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}\phi(\mathbf{x}), \tag{2.4.54}$$

$$= x_1\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\phi(\mathbf{x}) + (k_1-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}\phi(\mathbf{x}). \tag{2.4.55}$$

Hence, when $\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}$ is applied to Equation 2.4.45 for $i = 1$, the result, Equation 2.4.49, from using Equations 2.4.51 - 2.4.55 becomes

$$\begin{aligned} |\mathbf{x}|^2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\phi(\mathbf{x}) &+ 2k_1x_1\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\phi(\mathbf{x}) + 2k_2x_2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_2}\phi(\mathbf{x}) + 2k_3x_3\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_3}\phi(\mathbf{x}) \\ &+ k_1(k_1-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}\phi(\mathbf{x}) + k_2(k_2-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_2}\phi(\mathbf{x}) + k_3(k_3-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_3} \\ &- x_1\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\phi(\mathbf{x}) - (k_1-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}\phi(\mathbf{x}) \qquad (2.4.56) \\ &= \mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}). \end{aligned}$$

This can be written in a compact form as,

$$\begin{aligned} &|\mathbf{x}|^2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\phi(\mathbf{x}) + 2\sum_{i=1}^{3} k_ix_i\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\phi(\mathbf{x}) + \sum_{i=1}^{3} k_i(k_i-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\phi(\mathbf{x}) \qquad (2.4.57) \\ - \;& x_1\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\phi(\mathbf{x}) - (k_1-1)\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}\phi(\mathbf{x}) = \mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}). \end{aligned}$$

By multiplying through Equation 2.4.57 by $\frac{1}{\mathbf{k}!}$, we get

$$\begin{aligned} &\frac{1}{\mathbf{k}!}|\mathbf{x}|^2\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\phi(\mathbf{x}) + 2\sum_{i=1}^{3}\frac{k_i}{\mathbf{k}!}x_i\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\phi(\mathbf{x}) + \sum_{i=1}^{3}\frac{k_i(k_i-1)}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\phi(\mathbf{x}) - x_1\frac{1}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_1}\phi(\mathbf{x}) \\ &\qquad -\frac{(k_1-1)}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_1}\phi(\mathbf{x}) = \frac{1}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\psi(\mathbf{x}). \end{aligned} \tag{2.4.58}$$

Recall that the coefficients of the kth order term in the Taylor expansion of $\phi(\mathbf{x})$ were defined in Equation 2.4.9 as

$$a_{\mathbf{k}} = \frac{1}{\mathbf{k}!}\mathbf{D}_{\mathbf{x}}^{\mathbf{k}}\phi(\mathbf{x}), \tag{2.4.59}$$

and as a result, the identities

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-\mathbf{e}_i}\phi(\mathbf{x}) = a_{\mathbf{k}-\mathbf{e}_i}\cdot(\mathbf{k}-\mathbf{e}_i)! \quad \text{and} \tag{2.4.60}$$

$$\mathbf{D}_{\mathbf{x}}^{\mathbf{k}-2\mathbf{e}_i}\phi(\mathbf{x}) = a_{\mathbf{k}-2\mathbf{e}_i}\cdot(\mathbf{k}-2\mathbf{e}_i)!, \tag{2.4.61}$$

hold as well.Thus using Equation 2.4.31, Equation 2.4.58 becomes

$$|\mathbf{x}|^2 a_{\mathbf{k}} + 2\sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e}_i} + \sum_{i=1}^{3} a_{\mathbf{k}-2\mathbf{e}_i} - \frac{x_1}{k_1} a_{\mathbf{k}-2\mathbf{e}_1} - \frac{1}{k_1} a_{\mathbf{k}-2\mathbf{e}_1} = b_{\mathbf{k}}. \tag{2.4.62}$$

We then multiply through Equation 2.4.62 with $k_1$which yields

$$k_1|\mathbf{x}|^2 a_{\mathbf{k}} + 2k_1\sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e}_i} + k_1\sum_{i=1}^{3} a_{\mathbf{k}-2\mathbf{e}_i} - x_1 a_{\mathbf{k}-2\mathbf{e}_1} - a_{\mathbf{k}-2\mathbf{e}_1} = k_1 b_{\mathbf{k}}. \tag{2.4.63}$$

The analogous development for $i = 2$ and $i = 3$ results in

$$k_2|\mathbf{x}|^2 a_{\mathbf{k}} + 2k_2\sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e}_i} + k_2\sum_{i=1}^{3} a_{\mathbf{k}-2\mathbf{e}_i} - x_2 a_{\mathbf{k}-2\mathbf{e}_2} - a_{\mathbf{k}-2\mathbf{e}_2} = k_2 b_{\mathbf{k}} \tag{2.4.64}$$

and

$$k_3|\mathbf{x}|^2 a_{\mathbf{k}} + 2k_3\sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e}_i} + k_3\sum_{i=1}^{3} a_{\mathbf{k}-2\mathbf{e}_i} - x_3 a_{\mathbf{k}-2\mathbf{e}_3} - a_{\mathbf{k}-2\mathbf{e}_3} = k_3 b_{\mathbf{k}}. \tag{2.4.65}$$

respectively.

By summing up Equation 2.4.63, Equation 2.4.63 and Equation 2.4.63 we arrive at

$$||\mathbf{k}|||x|^2 a_{\mathbf{k}} + (2||\mathbf{k}|| - 1)\sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e_i}} + (||\mathbf{k}|| - 1)\sum_{i=1}^{3} a_{\mathbf{k}-\mathbf{2e_i}} = ||\mathbf{k}||b_{\mathbf{k}}. \tag{2.4.66}$$

Then the recurrence relation for computing the Taylor coefficients $a_{\mathbf{k}}$ for $V_{i,C}^r$ is

$$a_{\mathbf{k}} = \frac{1}{|x|^2}\left\{\left(\frac{1}{||\mathbf{k}||} - 2\right)\sum_{i=1}^{3} x_i a_{\mathbf{k}-\mathbf{e_i}} - \left(1 - \frac{1}{||\mathbf{k}||}\right)\sum_{i=1}^{3} a_{\mathbf{k}-\mathbf{2e_i}} + b_{\mathbf{k}}\right\}, \tag{2.4.67}$$

where $b_{\mathbf{k}}$ are the Taylor coefficients of $\frac{1}{2}e^{-|\mathbf{x}|^2}$ with recurrence relation

$$b_{\mathbf{k}} = -\frac{2}{||\mathbf{k}||}\sum_{i=1}^{3}\left\{x_i b_{\mathbf{k}-\mathbf{e}_i} + b_{\mathbf{k}-2\mathbf{e}_i}\right\}. \tag{2.4.68}$$

In these expressions, $a_{\mathbf{k}} = 0$ and $b_{\mathbf{k}} = 0$ when any one of k_1, k_2 or k_3 is negative.

2.5 Conclusion

In this chapter we have derived the Ewald Sum which arises from rewriting the Coulomb potential in periodic boundary conditions as a sum of two absolutely convergent series and a finite sum. The two absolutely convergent series, V^r in real space and V^k in reciprocal space decay at different rates for a specified value of α. We derived the particle-mesh Ewald (PME) method which speeds up the computation of the reciprocal space sum, V^k from $O(N^2)$ to $O(N\log N)$. We also provided a full derivation of the Cartesian Treecode Ewald (CTE) method which speeds up the computation of the real space sum, V^k from $O(N^2)$ to $O(N\log N)$. Table 2.1 summarizes the cost of computing the Ewald sum via direct summation, Particle-Mesh Ewald and the Cartesian Treecode Ewald algorithm.

	Direct Sum	Optimized Direct Sum	PME	CTE
Real Space Sum - V^r	$O(N^2)$	$O(N^{3/2})$	$O(N)$	$O(N\log N)$
Reciprocal Space Sum - V^k	$O(N^2)$	$O(N^{3/2})$	$O(N\log N)$	$O(N)$
Overall cost	$O(N^2)$	$O(N^{3/2})$	$O(N\log N)$	$O(N\log N)$

Table 2.1
The cost of the different algorithms for computing the Ewald sum.

In the next chapter we will provide simulation results which compare the CTE method to the PME method.

CHAPTER III

Validation of Cartesian Treecode Ewald Method

This section presents results that validate the Cartesian treecode Ewald (CTE) method as a method for performing molecular dynamics simulations. We apply CTE to compute the electrostatic interactions in a molecular dynamics (MD) simulation of liquid methyl chloride (CH_3Cl). We compute the radial distribution functions, $g(r)$, force-force, $c_{FF}(t)$, and velocity-velocity, $c_{VV}(t)$, auto-correlation functions and compare these to those obtained by computing electrostatic interactions with the Particle Mesh Ewald (PME) method.

We compare the CTE method to the PME method for computation time, relative error in energy and root mean square error in force, for different system sizes with the error in approximation of the Ewald sum set to 1×10^{-5}. We also compare a hybrid CTE-PME algorithm to the PME method to investigate whether the hybrid will be faster than both the original CTE algorithm and the PME algorithm.

The results in this chapter are generated with a particle-cluster CTE algorithm. In chapter IV, we present results generated with a *leaf-cluster* CTE algorithm.

3.1 Implementation of Periodic Boundary Conditions

We performed an MD simulation of 256 molecules of CH_3Cl with periodic boundary conditions to model a bulk system. The problem of implementing periodic bound-

ary conditions for hierarchical tree based multipole algorithms is a challenging one. Schmidt and Lee [49, 50] have developed a method for handling periodic boundary conditions with the Fast Multipole Method of Greengard and Rohklin [26, 11] based on the Ewald sum. Our implementation of periodic boundary conditions in the CTE algorithm is based on a procedure presented by Bouchet and Hernquist [8] for cosmological simulations using the Barnes-Hut tree algorithm. This procedure implements the minimum image convention [2, 25, 45] in addition to periodic boundary conditions. We will now describe how periodic boundary conditions are implemented in the CTE algorithm.

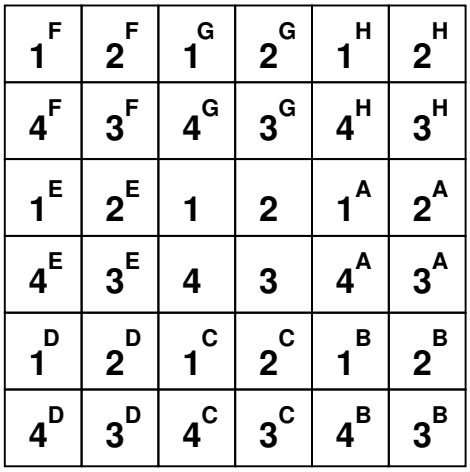


Figure 3.1: The figure shows the fundamental cell divided into four sub-clusters $\{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$ and replicated in space. Each subcell has a tree structure.

The periodic system is a replication of the fundamental cell in space [2]. Figure 2.1 in chapter II showed a two-dimensional representation of a fundamental cell replicated in space. Figure 3.1 also shows a two-dimensional representation of a fundamental cell replicated in space. However in Figure 3.1, the fundamental cell is divided into four sub-clusters $\{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$ as a result of the hierarchical clustering in the CTE algorithm. The exact interactions in a periodic system requires the target particle to interact with every other particle in the system as well as all of its images

and the images of the other particles. However, if the potential energy function is short range then only a finite number of particles close to a target particle have significant interactions with the target particle. All other contributions are too small and can be ignored. If the interactions between particles are insignificant beyond half the length of the fundamental cell, then as Figure 3.2 [2] shows, the relevant interactions of a target particle can be restricted to the particles in a simulation volume with the same dimensions as the fundamental cell centered at the target particle. In the figure, the fundamental cell is the box that is not labeled with a letter. The target particle is the particle labeled **1** with the simulation box centered at **1** shown with broken lines. The target particle interacts with the particles or the images of the particles which are in the box centered on **1**. In this case, **1** interacts with **2** and the images $\mathbf{3}^H$ and $\mathbf{4}^B$. This algorithm is called the minimum image. Thus a

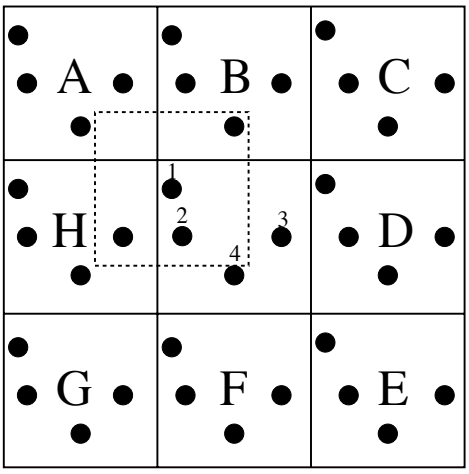


Figure 3.2: The figure shows the minimum image convention applied to the interactions of particle **1**. The image cells of the central fundamental cell are labeled with letters from **A** to **H**. Particle **1** interacts with particles **2**, $\mathbf{3}^H$ and $\mathbf{4}^B$.

target particle interacts with either a particle in the fundamental cell or the closest (minimum) image of that particle.

Because the real and reciprocal space parts of the Ewald sum decay quickly, we

are able to implement the minimum image convention in the CTE algorithm. For a simulation cell of length L, we pick an α value such that Equation 2.3.1 gives an error less or equal to the specified error at a cutoff radius of $r_c = \frac{L}{2}$. We then restrict the particles that interact with a target particle to those particles that are within the simulation volume centered at the target particle. We will use the two-dimensional representation shown in Figure 3.1 to futher explain the procedure.

In three-dimensions the fundamental cell will be divided into eight sub-clusters. However, in Figure 3.1, the fundamental cell is divided into four sub-clusters $\{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$, after the tree building process, and each each sub-cluster in turn has tree structure. We will focus on the process for computing the interactions for the particles in $\mathbf{1}$.

Clearly, to determine the interactions of the particles in $\mathbf{1}$ according to the minimum image convention we will have to pinpoint each target particle and determine the simulation volume centered at the particle. This simulation volume will intersect with the clusters $\mathbf{1}$, $\mathbf{3}^F$, $\mathbf{4}^G$, $\mathbf{3}^G$, $\mathbf{2}$, $\mathbf{3}$, $\mathbf{4}$, $\mathbf{3}^E$ and $\mathbf{2}^E$ which are in direct contact with $\mathbf{1}$. However, for ease of coding we increase the simulation volume to encompass all the sub-clusters that touch $\mathbf{1}$ in order to be able to use the tree built from the fundamental cell.

Then a target particle, i, in $\mathbf{1}$ interacts with the other particles in $\mathbf{1}$ and all of the particles in $\mathbf{3}^F$, $\mathbf{4}^G$, $\mathbf{3}^G$, $\mathbf{2}$, $\mathbf{3}$, $\mathbf{4}$, $\mathbf{3}^E$ and $\mathbf{2}^E$. This interaction proceeds by a call to a subroutine

(3.1.1) `Compute_Interaction(target particle,cluster)` .

The call `Compute_Interaction(i,C)` is straightforward for $C \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$ since these clusters already have a tree structure. The interactions between the target particle and particles in the image cells $\mathbf{3}^F$, $\mathbf{4}^G$, $\mathbf{3}^G$, $\mathbf{3}^E$ and $\mathbf{2}^E$ require a little bit more work. For example, the interaction of the target particle, i, with the particles

in the image cell $\mathbf{3}^E$ by a call to `Compute_Interaction(i,`$\mathbf{3}^E$`)` is replaced by the call `Compute_Interaction(`i^A`,3)` where i^A is the position of the periodic image of particle i in the image cell $\mathbf{1}^A$. This is because the effect of $\mathbf{3}^E$ on i, is the same as the effect of $\mathbf{3}$ on 1^A. Hence, instead of creating $\mathbf{3}^E$ which will involve additional work and storage, the vector between i and the center of cluster $\mathbf{3}^E$ is computed as the vector between i^A and the center of cluster $\mathbf{3}$, which requires no extra storage or work. The moments of cluster $\mathbf{3}$ are the same as the moments of cluster $\mathbf{3}^E$, since the moments are a local property and depend only on the particles in the cluster and their separation from the center of the cluster. Similarly, `Compute_Interaction(i,`$\mathbf{4}^G$`)` is replaced by `Compute_Interaction(`i^C`,4)`. All the interactions with images are handled in a similar fashion.

3.2 MD Simulation of Liquid CH_3Cl

3.2.1 Simulation Details

The simulations were carried out on 256 molecules of CH_3Cl in a 28.20Å cubic box at $220K$ and 0.0365 bars. Each CH_3Cl molecule is modeled as a diatomic with charge +0.25 au at the CH_3 site and −0.25 au at the Cl site. The atomic mass of CH_3 and Cl were given as 15.0345 amu and 35.4530 amu respectively. The two sites are connected by a rigid bond of length 1.781 Å. We used an effective intermolecular potential based on the site-site model presented by Cabral et al. [10, 40] in their simulation of CH_3Cl. This effective potential, V_{total}, is the sum of a short-ranged (van der Waals) potential modeled with the $12-6$ or Lennard Jones potential and the Coulomb potential. V_{total} is given as

$$V_{\text{total}} = \sum_{i<j}^{N} \frac{A}{r_{ij}^{12}} - \frac{B}{r_{ij}^{6}} + \frac{q_i q_j}{4\pi\epsilon_o r_{ij}}. \tag{3.2.1}$$

Here, $r_{ij} = |\mathbf{r_i} - \mathbf{r_j}|$ and the values of A and B given in Table 3.1 depend on the two particles interacting. The Coulomb potential part of this effective potential is what is transformed into the Ewald sum. The Lennard Jones terms were computed directly. Lennard-Jones interactions beyond a cutoff of 14.104Å were discarded and taken to be zero.

Atom pair	A (10J mol^{-1} Å^{12})	B (10J mol^{-1} Å^{6})
CH_3-CH_3	33346480.00	11506.00
CH_3-Cl	27064529.92	11917.06
Cl-Cl	21966000.00	12342.80

Table 3.1: Parameters for the $12-6$ potential of CH_3Cl [10]

The classical Ewald method and the PME method both directly exclude intramolecular interactions when computing the real space sum, V^r. Intramolecular interactions in our simulation refers to an interaction between the CH_3 and Cl species making up the diatomic molecule. However, when computing the reciprocal sum, V^k, each chemical species is initially taken to be a target particle and as such the computation includes the intramolecular interactions. The intramolecular interactions are then subtracted out by a separate routine. The intramolecular part of the reciprocal space potential energy, V^k_{ex}, subtracted from the potential energy is given by the formula [53]

$$V^k_{ex} = \frac{1}{4\pi\epsilon_o} \sum_{\text{molecules}} \sum_{i\leq j}^{M^*} q_i q_j \left\{ \delta_{ij} \frac{\alpha}{\sqrt{\pi}} + \frac{\text{erf}(\alpha r_{ij})}{r_{ij}^{1-\delta_{ij}}} \right\}, \tag{3.2.2}$$

where M^* is the number of excluded atoms in a given molecule. We note that the first term in the bracket is the atomic self correction [53] which is just the limit given by

$$\lim_{r_{ij}\longrightarrow 0} \frac{\text{erf}(\alpha r_{ij})}{r_{ij}} = 2\frac{\alpha}{\sqrt{\pi}}, \tag{3.2.3}$$

where the factor of 2 comes from considering a pair of excluded atoms in an intramolecular bond. When $i = j$, the second term in Equation 3.2.2 evaluates to 0.

The CTE method on the other hand includes intramolecular interactions in the real space sum in addition to the reciprocal space sum in the initial computation of V^r. Hence, in addition to V_{ex}^k from Equation 3.2.2, the intramolecular part of the real space potential energy, V_{ex}^r, has to be subtracted from the potential energy as well. This potential energy is given by

(3.2.4)
$$V_{ex}^r = \frac{1}{4\pi\epsilon_o} \sum_{\text{molecules}} \sum_{i<j}^{M*} q_i q_j \frac{\text{erfc}(\alpha r_{ij})}{r_{ij}} = \frac{1}{4\pi\epsilon_o} \sum_{\text{molecules}} \sum_{i<j}^{M*} q_i q_j \frac{1 - \text{erf}(\alpha r_{ij})}{r_{ij}}.$$

By adding equations Equation 3.2.2 and Equation 3.2.4, we find that the total excluded potential energy, $V_{ex} = V_{ex}^k + V_{ex}^r$ is given by

(3.2.5)
$$V_{ex} = \frac{1}{4\pi\epsilon_o} \sum_{\text{molecules}} \left\{ \sum_{i=1}^{M*} q_i^2 \frac{2\alpha}{\sqrt{\pi}} + \sum_{i<j}^{M*} \frac{q_i q_j}{r_{ij}} \right\}.$$

We performed MD simulations in a constant energy (NVE) ensemble [2, 46] of the CH_3Cl system for two cases, one in which the Ewald sum was handled by PME and the other by CTE. For the simulations employing CTE, we performed runs for different sets of θ and multipole order, p with the maximum number of particles in a leaf, $N_0 = 6$. The (θ, p) pair is for all combinations of $\theta \in \{0.5, 0.6, 0.7\}$ and $p \in \{0, 2, 4, 6, 8\}$. For the simulations using PME, we used an 8th order interpolation.

For both methods, a real space cutoff of one-half the box length, i.e., 14.10Å was used. We generated an equilibrium configuration of the system at $220K$ from an initial lattice arrangement using velocity scaling. This equilibrium configuration was the starting configuration for both the PME and CTE simulations. The Verlet leapfrog algorithm [2] was used for numerical integration with a time step of 4 fs.

The Shake algorithm [24] was used to maintain the rigid bonds specification.

The simulations were performed by incorporating the CTE algorithm into DL_POLY [52] version 2.17. DL_POLY is a suite of Fortran subroutines for performing MD simulations developed by the CCLRC Daresbury Laboratory in the United Kingdom and distributed to academic researchers without charge. The lab which is part of the U.K.'s Science and Technology Facilities Council maintains a website (Figure 3.3) for the DL_POLY package.

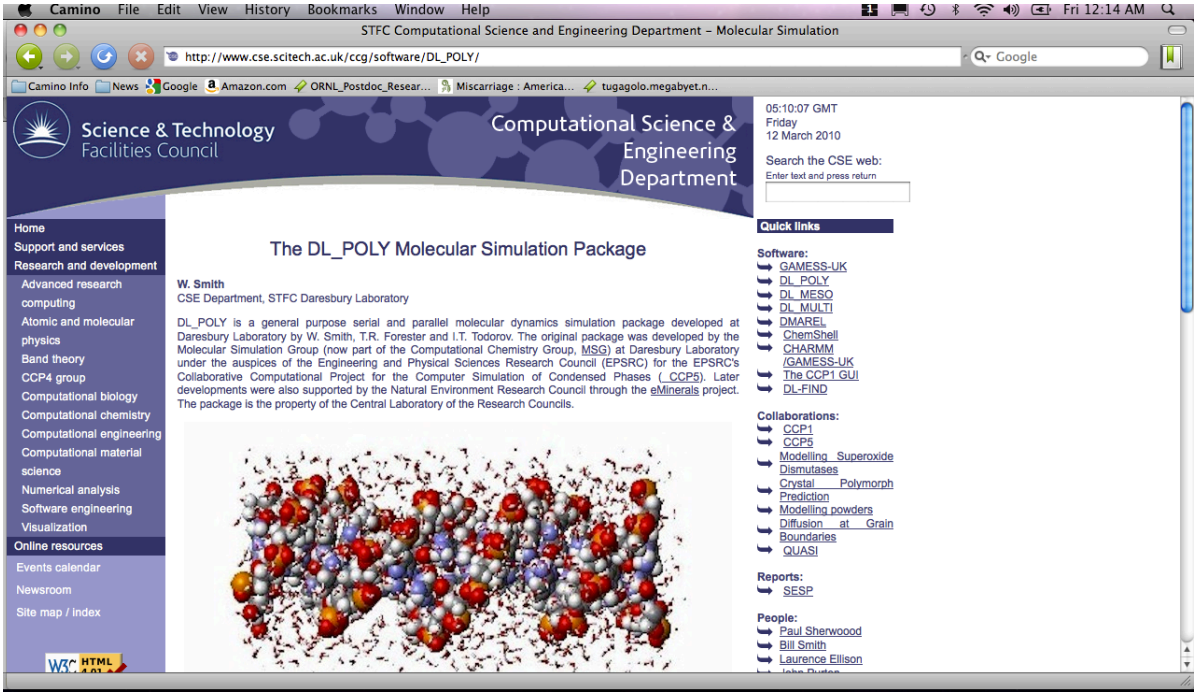


Figure 3.3: The webpage for DL_POLY
http:www.cse.scitech.ac.uk/ccg/software/DL_POLY

A component of this thesis was to incorporate the CTE algorithm into DL_POLY which has been successfully accomplished for both version 2.17 and the most recent version, 2.20. Version 2.20 provides additional options including energy minimization and hyperdynamics which are not present in version 2.17. They both, however, come with an optimized PME method which serves as our basis of comparison.

We computed the velocity autocorrelation function, $c_{VV}(t)$, and the force auto-

correlation function, $c_{FF}(t)$, by averaging over 24 consecutive 40 ps long trajectories. With the time step of 4 fs, each trajectory consists of 10000 different configurations of the system. We also generated the site-site radial distribution functions, $g(r)_{Me-Me}$, $g(r)_{Me-Cl}$ and $g(r)_{Cl-Cl}$ by averaging over 24000 different ensembles of the system. Here, Me refers to CH_3.

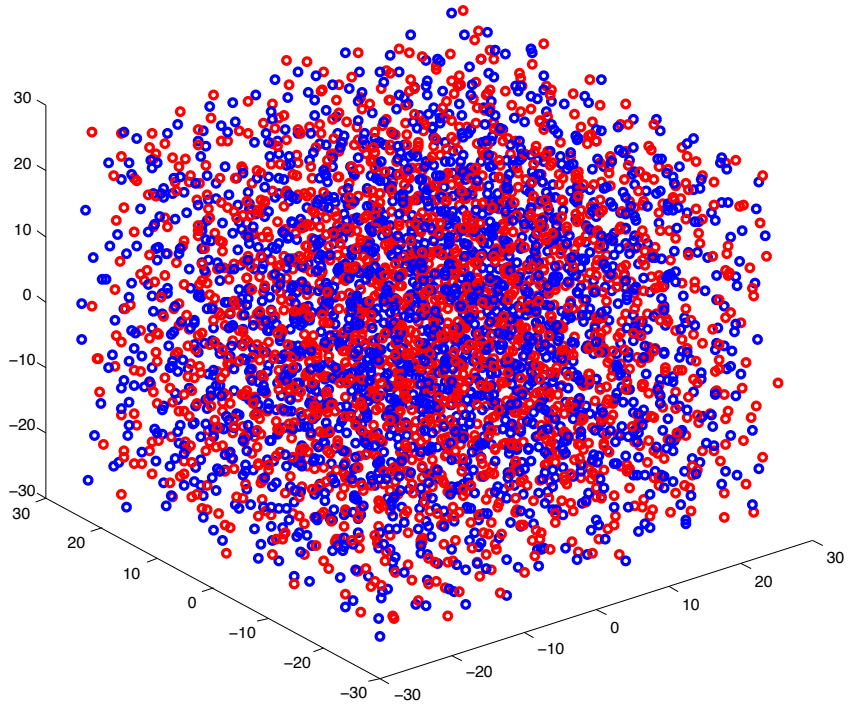


Figure 3.4: 2048 molecules of nonequilibrated CH_3Cl. The blue particles are CH_3 groups and the red Cl. The CH_3-Cl bond is omitted.

3.3 Simulation Results - Comparison with PME

Figure 3.4 is a uniform distribution of 2048 molecules of CH_3Cl.

3.3.1 Radial Distribution Function

The radial distribution function, $g(r)$, is a pair distribution function which gives insight into the structure of the liquid. The radial distribution function of any atom pair is the ratio of the probability of finding the two atoms a certain distance, r, apart

to the probability of finding two atoms of an ideal gas that same distance apart. The radial distribution function for a pair of atoms at a distance r in a system of size L is computed as an ensemble average given by [2]

$$g(r) = \frac{L^3}{N^2} \left\langle \sum_i \sum_{j \neq i} \delta(\mathbf{r} - \mathbf{r}_{ij}) \right\rangle_{ens} . \tag{3.3.1}$$

The radial distribution function of a liquid can be generated experimentally by x-ray or neutron diffraction. A procedure for computing $g(r)$ is outlined in [2, 25] with sample codes. We give a brief explanation here.

The probability of finding an atom pair a distance r apart is directly proportional to the number of one of the atoms of the pair separated by a distance r from the other atom. An atom pair could theoretically be as little as zero distance apart in the absence of a repulsive potential to as much as a distance $\frac{L}{2}$ apart which is the largest possible separation for significant interaction. The goal is to find for a given atom the number of the other atom of its pair which are r part for $0 \leq r \leq \frac{L}{2}$. This is approximated by finding for the given atom, i, the number of the other atom of its pair $n_{i_p}(r)$ separated by a discrete number, n_d, of separation distances r in the given range each with width

$$\delta r = \frac{L}{2 \cdot n_d}. \tag{3.3.2}$$

For a given atom i, $n_{i_p}(r)$ is found by checking all the atoms in its pair and sorting the separation into a histograms with a bin at each separation distance r which extends from r to $r + \delta r$. For example, for the Cl-Cl pair, the algorithm focuses on one Cl atom and checks the separation of this Cl atom with all other Cl atoms. This procedure is repeated for all the other Cl atoms as Equation 3.3.1 shows with the summation over i and the results are averaged over all N Cl atoms. Thus, for a

given atom pair, the total number of pairs, $n_p^T(r)$, for a given separation r is

$$n_p^T(r) = \frac{1}{N}\sum_{i=1}^{N} n_{i_p}(r). \tag{3.3.3}$$

The number of ideal gas pairs $n_{id}(r)$ at a separation distance r is computed as

$$n_{id}(r) = \frac{4\pi\rho}{3}\{(r+\delta r)^3 - r^3\} \tag{3.3.4}$$

which is the the number of ideal gas atoms with the same density $\rho = \frac{N}{L^3}$ as the liquid, in this case CH_3Cl, in the shell region $[r, r+\delta r]$. Then, the radial distribution function at $r + \frac{1}{2}\delta r$ is

$$\begin{aligned} g\left(r+\frac{1}{2}\delta r\right) &= \frac{n_p^T(r)}{n_{id)}(r)} = \frac{3}{4\pi\rho\{(r+\delta r)^3 - r^3\}}\frac{1}{N}\sum_{i=1}^{N} n_{i_p}(r), \\ &= \frac{1}{N\rho}\sum_{i=1}^{N}\frac{3n_{i_p}(r)}{4\pi\{(r+\delta r)^3 - r^3\}}, \\ &= \frac{L^3}{N^2}\sum_{i=1}^{N}\frac{3n_{i_p}(r)}{4\pi\{(r+\delta r)^3 - r^3\}}. \end{aligned} \tag{3.3.5}$$

We compute $g\left(r+\frac{1}{2}\delta r\right)$ for all atom pairs over 24000 different ensembles and average over the ensemble to yield

$$g\left(r+\frac{1}{2}\delta r\right) = \frac{L^3}{N^2}\left\langle\sum_{i=1}^{N}\frac{3n_{i_p}(r)}{4\pi\{(r+\delta r)^3 - r^3\}}\right\rangle_{ens}. \tag{3.3.6}$$

The radial distribution functions for PME and the different (θ, p) pairs in CTE are given in Figure 3.5 to Figure 3.13. These results correctly reproduce the work of Cabral et al. [40]. The PME results are plotted with error bars given by the standard deviation of the PME data. All three radial distribution functions $g(r)_{Me-Me}$, $g(r)_{Me-Cl}$ and $g(r)_{Cl-Cl}$ are zero for separations up to approximately 3Å. This shows the effective width of the atoms as a result of strong repulsive forces. The peaks show that the liquid has high local densities at the distances where the peaks occur. These are distances of high correlations between atom pairs. High local density areas are

followed by low local density areas and vice versa, resulting in the oscillatory behavior the functions. At long distances, the correlation between atom pairs decreases because their effect on each other decreases with distance and screening from intermediate molecules. As a result, the interaction between the atom pairs as the separation increases approaches that of an ideal gas where there is no interaction. Thus, the $g(r)$ plateaus to one with increasing separation.

The highest peak for $g(r)_{Me-Cl}$ occurs at a smaller distance and is higher than either $g(r)_{Me-Me}$ or $g(r)_{Cl-Cl}$ because the Me and Cl groups have additional attraction from electrostatic interactions because of their opposite charges in addition to the attractive part of the Lennard-Jones potential . As such a lot more of the Me-Cl pair are able to overcome the repulsive forces and get closer to each other than the Me-Me or Cl-Cl pairs. The Me-Me and Cl-Cl pairs have repulsive electrostatic interactions because of the identical charges of the atoms in each pair. Although the Me-Me and Cl-Cl pairs have similar repulsive electrostatic interactions, the Cl-Cl pair has a lower first peak because the Cl group with mass 35.4530 amu is bigger than the Me group with mass 15.0345 amu. As a result, clusters of the Cl group are looser than clusters of the Me group which results in a lower but more spread out peak for $g(r)_{Cl-Cl}$ as shown in Figure 3.7 than for $g(r)_{Me-Me}$ shown in Figure 3.5.

The radial distribution functions show that the CTE method is capable of reproducing the structural properties of the liquid within error bars for specific (θ, p) pairs. For Figures (3.5) to (3.7) with $\theta = 0.5$, CTE is well within error bars of the PME results. An increase in θ allows for more multipole approximations and thus requires higher order multipoles in order to achieve the required accuracy. In Figures (3.8) to (3.10) where $\theta = 0.6$, the results for $p = 6$ and $p = 8$ are still completely within one standard deviation of the PME results. The data points for $p = 4$ are

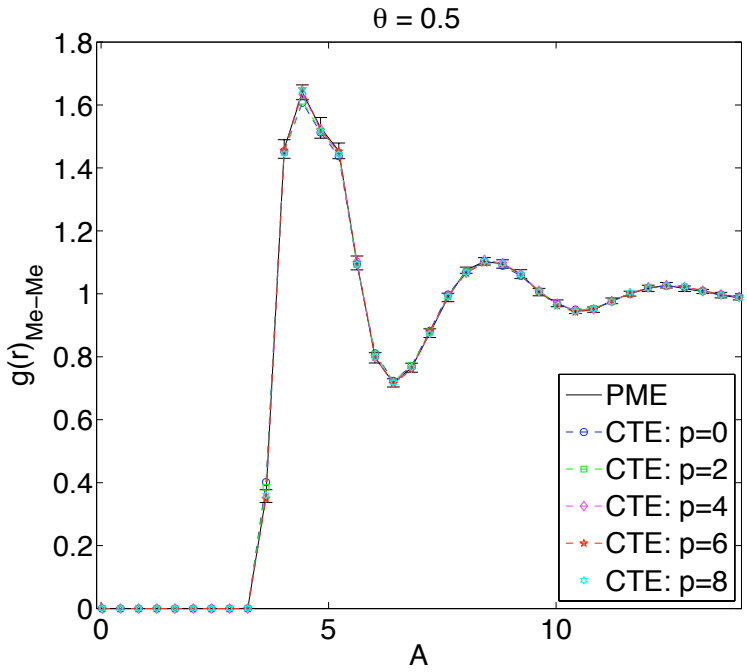


Figure 3.5: CH_3-CH_3 radial distribution function for PME and CTE with ($\theta = 0.5$). The CTE result is within error bounds of the PME result for all orders. In this regime, with the multipole acceptability criterion, $\theta = 0.5$ most of the computation is by direct summation. As a result, the order of the treecode does not have a significant impact on the results.

with a few exceptions at the peaks at troughs of the graph also within one standard deviation of the PME results. For $\theta = 0.7$, Figures (3.11) to (3.13) show that CTE with $p = 6$ and $p = 8$ produce results within the error bars of the PME results.

Setting $\theta = 0.5$ forces the CTE algorithm to descend deeper into the tree which results in a significant portion of the interactions being performed by direct summation, hence explaining the accuracy of the lower order simulations. This however results in an increase in computation time for larger systems. Larger θ values, on the other hand, skew the interactions towards more multipole approximations at higher levels of the tree. This in turn will require higher order multipoles in order to achieve the desired accuracy.

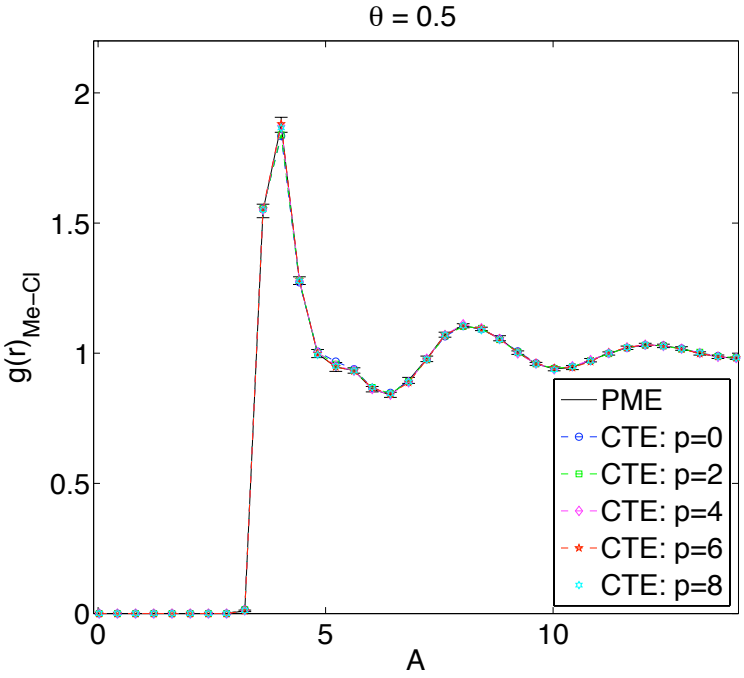


Figure 3.6: CH_3-Cl radial distribution function for PME and CTE with ($\theta = 0.5$). The CTE result is within error bounds of the PME result for all orders.

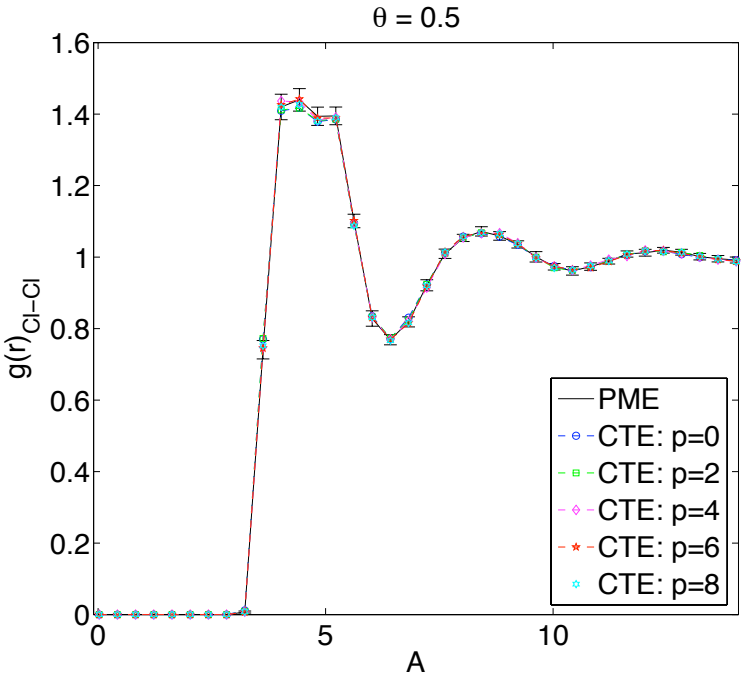


Figure 3.7: Cl-Cl radial distribution function for PME and CTE with ($\theta = 0.5$). The CTE result is within error bounds of the PME result for all orders.

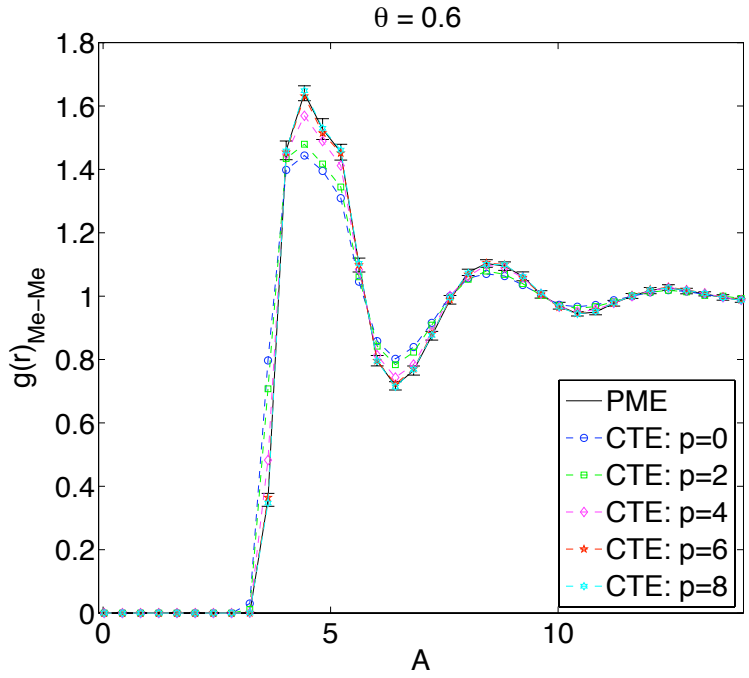


Figure 3.8: CH_3-CH_3 radial distribution function for PME and CTE with ($\theta = 0.6$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$. As expected a higher order multipole yields better accuracy.

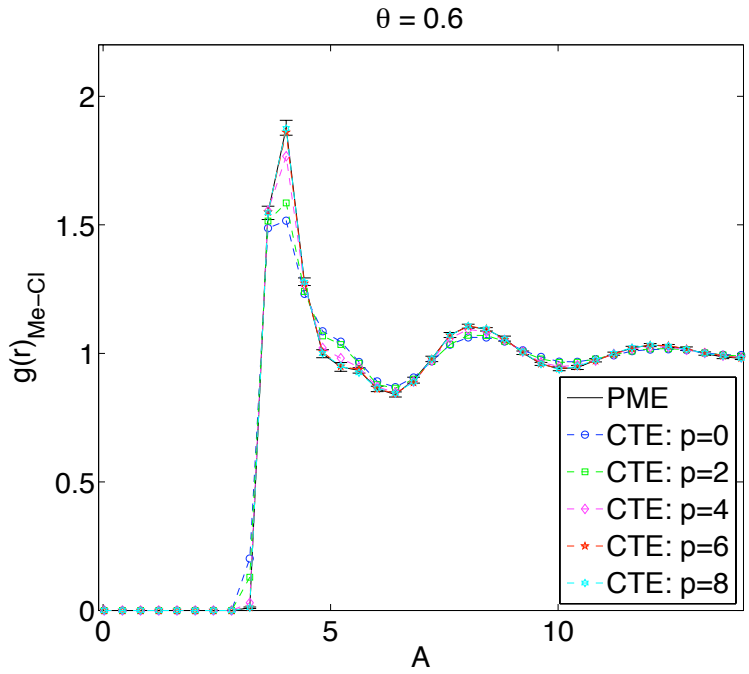


Figure 3.9: CH_3-Cl radial distribution function for PME and CTE with ($\theta = 0.6$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

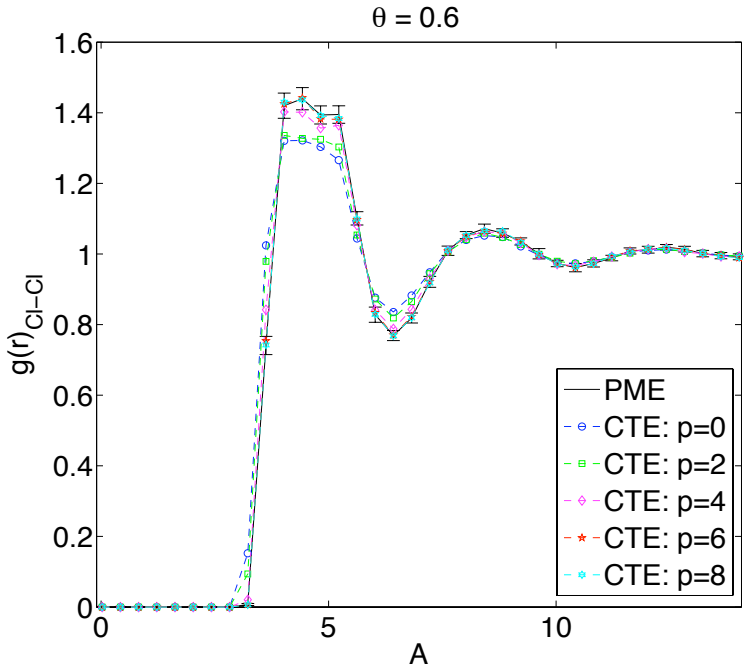


Figure 3.10: Cl-Cl radial distribution function for PME and CTE with ($\theta = 0.6$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

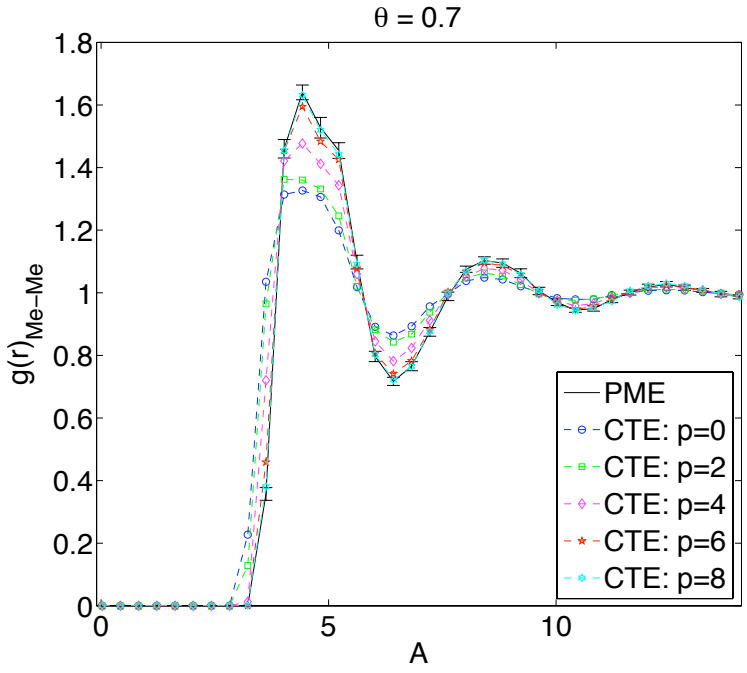


Figure 3.11: CH_3-CH_3 radial distribution function for PME and CTE with ($\theta = 0.7$). Again, the CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

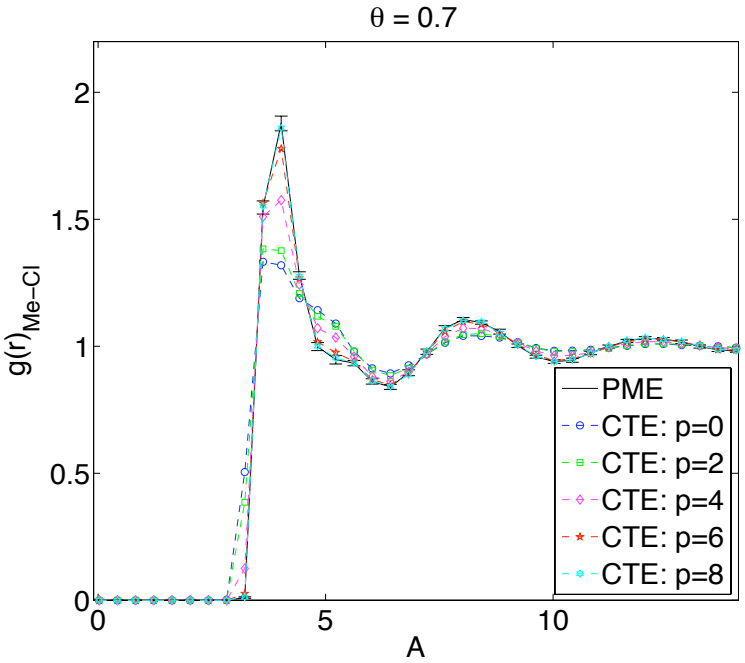


Figure 3.12: CH_3-Cl radial distribution function for PME and CTE with ($\theta = 0.7$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

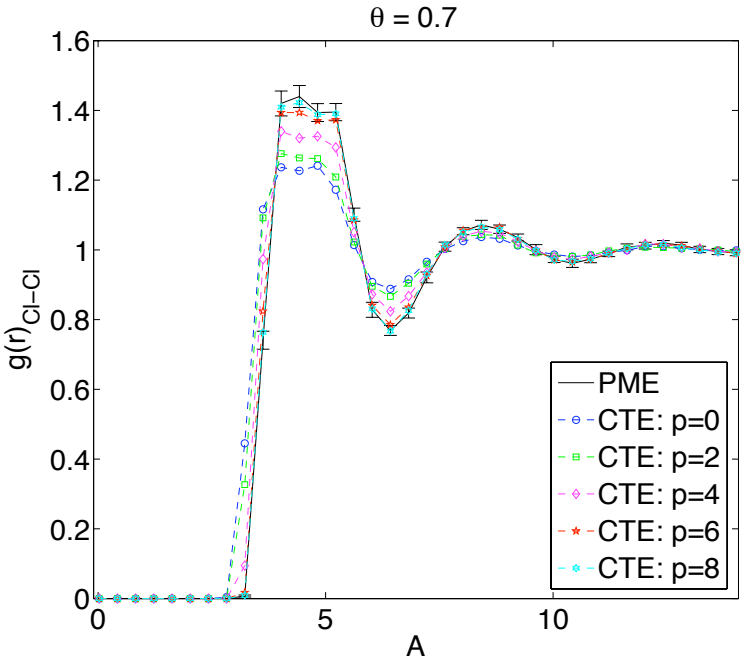


Figure 3.13: Cl-Cl radial distribution function for PME and CTE with ($\theta = 0.7$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

3.3.2 Velocity-Velocity and Force-Force Auto-Correlation Functions

The time correlation [2], $c_{\mathcal{AB}}(t)$ between two quantities, $\mathcal{A}(\tau)$ and $\mathcal{B}(\tau+t)$ for the time difference t is

$$c_{\mathcal{AB}}(t) = \frac{C_{\mathcal{AB}}(t)}{C_{\mathcal{AB}}(0)} = \frac{\langle \delta\mathcal{A}(\tau)\delta\mathcal{B}(\tau+t)\rangle_{ens}}{\sigma(\mathcal{A})\sigma(\mathcal{B})}, \tag{3.3.7}$$

where the numerator of the last equality in Equation 3.3.7 is an ensemble average and $\sigma(\mathcal{A})$ refers to the standard deviation in the quantity $\mathcal{A}$ at time $t = 0$ from averaging over a sufficient number of $t = 0$ realizations. The fluctuation $\delta\mathcal{A}$ is defined as

$$\delta\mathcal{A} = \mathcal{A} - < \mathcal{A} >_{ens} \tag{3.3.8}$$

and

$$\sigma^2(\mathcal{A}) =< \delta\mathcal{A}^2 >_{ens} =< \mathcal{A}^2 >_{ens} - < \mathcal{A} >^2_{ens} . \tag{3.3.9}$$

The ensemble average is employed in place of a time average because of the ergodicity of the system. In a system with N identical molecules, the ergodic theory allows for the complete trajectory of a single molecule to be replaced by a snapshot in time of an ensemble of all the N molecules. The time average is then equivalent to an average over a sufficient number of ensembles. A procedure for computing time correlation functions is provided in [2]. We outline the procedure here for the velocity-velocity $c_{VV}(t)$ and force-force auto-correlation $c_{FF}(t)$ functions.

The velocity vector, $\mathbf{V}(t)$ is the velocity of the center of mass of the CH_3Cl molecule while the force vector, $\mathbf{F}(t)$ is the force along the bond connecting the two atoms in the molecule. To compute $c_{VV}(t)$ and $c_{FF}(t)$ we follow Equation 3.3.7 and first compute $C_{VV}(t)$ and $C_{FF}(t)$ where

$$C_{VV}(t) = \langle \delta\mathbf{V}(\tau) \cdot \delta\mathbf{V}(\tau+t)\rangle_{ens} , \tag{3.3.10}$$

and

$$C_{FF}(t) = \langle \delta\mathbf{F}(\tau) \cdot \delta\mathbf{F}(\tau + t) \rangle_{ens} \,. \tag{3.3.11}$$

The basic operation in computing $C_{VV}(t)$ and $C_{FF}(t)$ is a dot product of the respective vectors for the time difference t. We take 10000 different snapshots of the ensemble with a time gap of $4fs$ between each snapshot. We compute the mean of the coordinates of the velocity and forces for each molecule over all 10000 snapshots and subtract this mean from the velocities and forces of each molecule respectively as given by Equation 3.3.8. We also compute the mean of the square of the velocity and force vectors and then the variance over all 10000 snapshots for each molecule via Equation 3.3.9.

Then for all possible time differences (gap) t within the 10000 snapshots, we compute the dot products of the vectors for each molecule. Note that the first snapshot can be correlated to the other 9999 snapshots with time difference t ranging from $t = 4fs$, which is the time gap between the first and second snapshots, to $t = (10000-1)*4fs$ which is the time gap between the first snapshot and the snapshot number 10000. The second snapshot can also provide correlations for time gaps $t = 4fs$ with snapshot number 3 up to a correlation for time gap $t = (10000-2)*4fs$ with snapshot number 10000. Thus, later snapshots can be correlated for fewer time gaps. Hence, snapshot number 9999 can only be correlated for time gap $t = 4fs$ with number 10000 and snapshot number 10000 can only be correlated to itself for a time gap $t = 0$. Every snapshot can be correlated to itself for a time gap of $t = 0$. This means that the time difference with the most data is $t = 0$ and the amount of data for each t decreases with increasing t. The dot product of the velocity and force vectors for each time difference are then averaged over the number of correlations possible for each time difference. Then we compute $c_{VV}(t)$ and $c_{FF}(t)$ by averaging

by the respective variances.

We perform the above procedure for 23 other different 10000 snapshots and average over all 24. Finally, we average over all the molecules since they are identical to arrive at Figures (3.14) to (3.19).

Further evidence of the effect of θ and p in the CTE simulations are given in the plots of the velocity-velocity autocorrelation function, $c_{VV}(t)$, in Figure 3.14 to Figure 3.16 and the force-force autocorrelation function, $c_{FF}(t)$, in Figure 3.17 to Figure 3.19. The correlation functions as expected show perfect correlation at $t = 0$ where snapshot is correlated to itself. As the time difference increases the two systems at the different times becomes less and less correlated and thus the correlation decays to zero. We say that the system loses memory of its initial time.

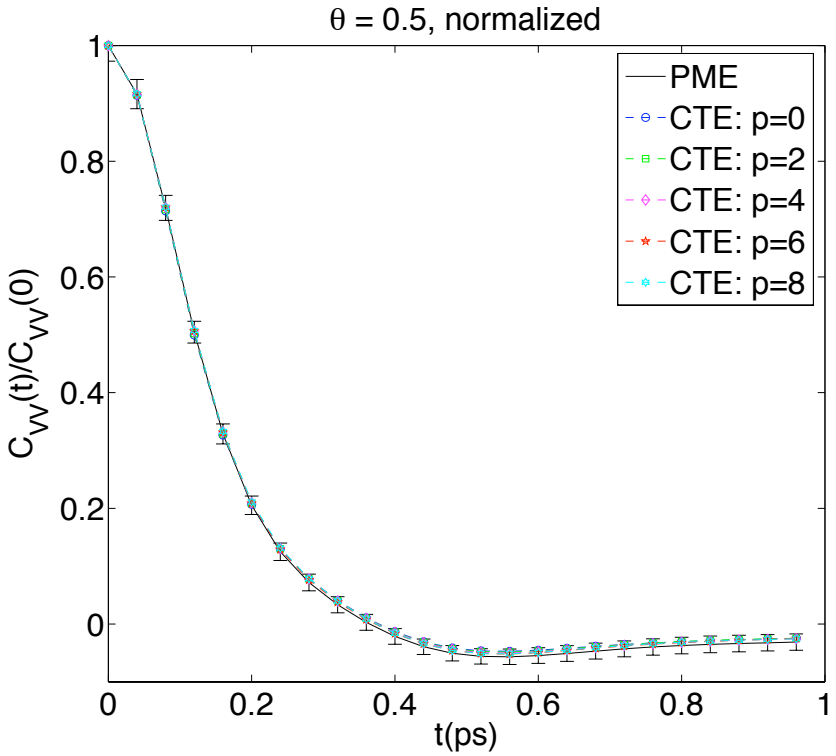


Figure 3.14: Center of mass velocity autocorrelation function for ($\theta = 0.5$). The CTE result is within error bounds of the PME result for all orders.

In addition to the dynamical insights provided by the time correlation functions,

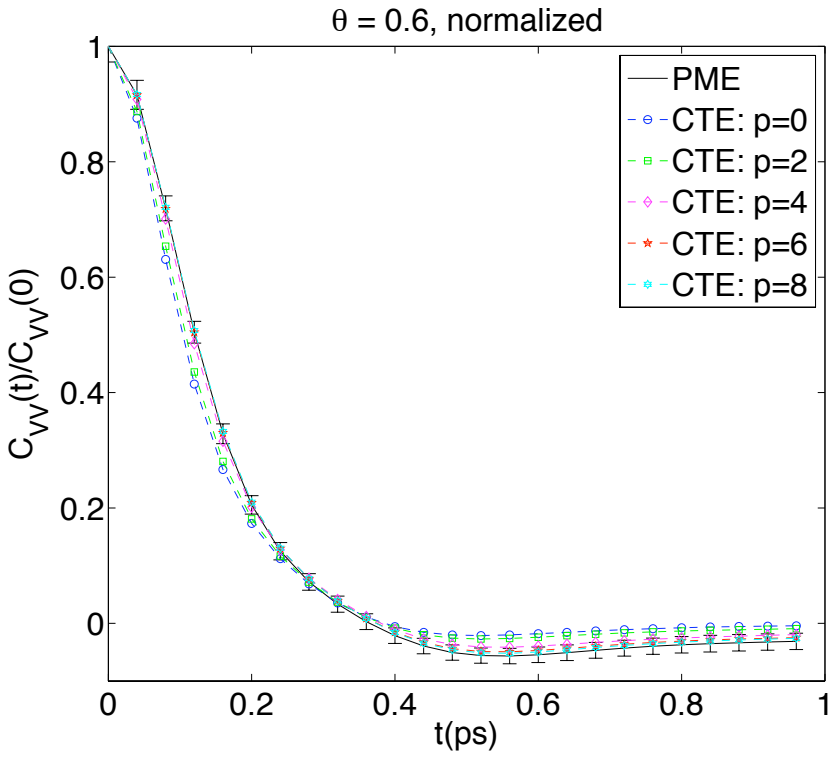


Figure 3.15: Center of mass velocity autocorrelation function for ($\theta = 0.6$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

the diffusion coefficient, D, of the liquid can be computed from the velocity-velocity autocorrelation function by the relation [2]

$$D = \frac{1}{3}\int_0^\infty \langle \mathbf{V_i}(t) \cdot \mathbf{V_i}(0) \rangle \, dt, \tag{3.3.12}$$

where $\mathbf{V_i}$ is the center-of-mass velocity of the ith molecule. Also, by linear response theory, the vibrational energy relaxation rate, $\frac{1}{T_1}$, of a solute in a solvent can be computed from the force-force autocorrelation function by the relation [35, 51]

$$\frac{1}{T_1} = \frac{1 - e^{-\beta\hbar\omega}}{\beta\hbar\omega}\frac{\beta}{2\mu}\int_{-\infty}^{\infty} e^{i\omega_0 t} \langle F(t)F(0) \rangle \, dt, \tag{3.3.13}$$

where, $\beta = 1/(\kappa_B T)$, μ is the reduced mass of the diatomic, and ω_0 is the vibrational frequency of the solute bond. The vibrational energy relaxation rate can be measured experimentally by IR-spectroscopy. Computing D and $\frac{1}{T_1}$ was not in the scope of

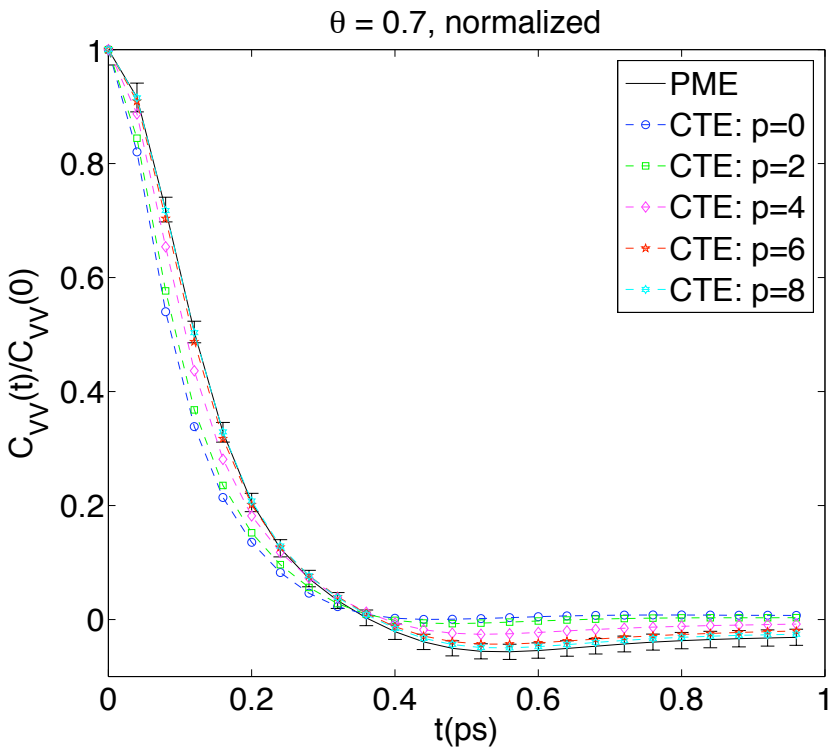


Figure 3.16: Center of mass velocity autocorrelation function for ($\theta = 0.7$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

the thesis. Future work will include computing D and $\frac{1}{T_1}$.

Figure 3.14 to Figure 3.19 show that the CTE algorithm is capable of generating the dynamical properties of systems of interest with the same behavior in relation to θ and p as seen for the radial distribution functions.

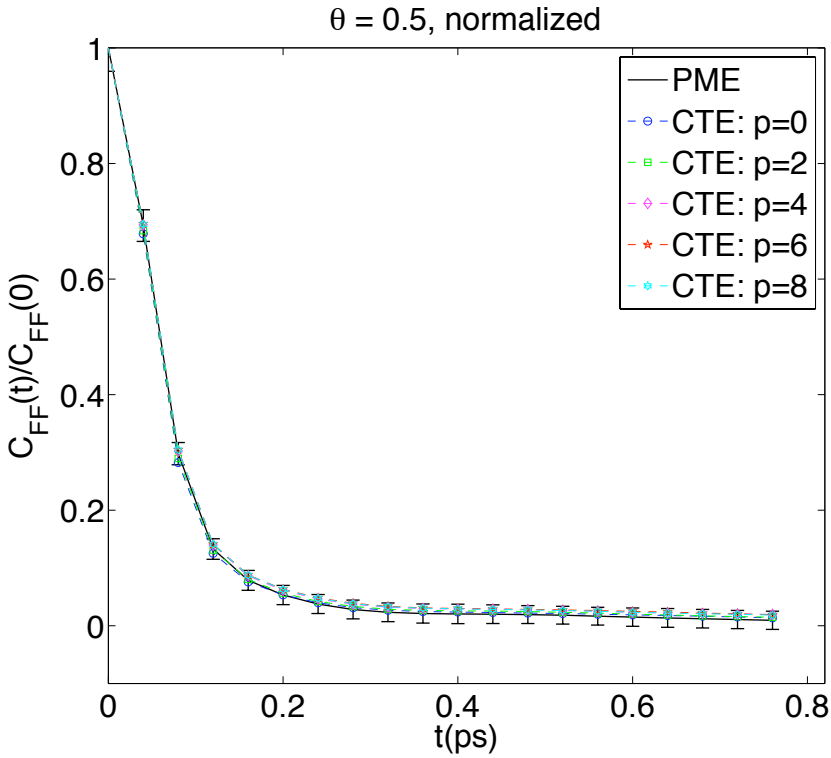


Figure 3.17: Autocorrelation function of force along molecular bond for ($\theta = 0.5$). Again, the CTE result is within error bounds of the PME result for all orders.

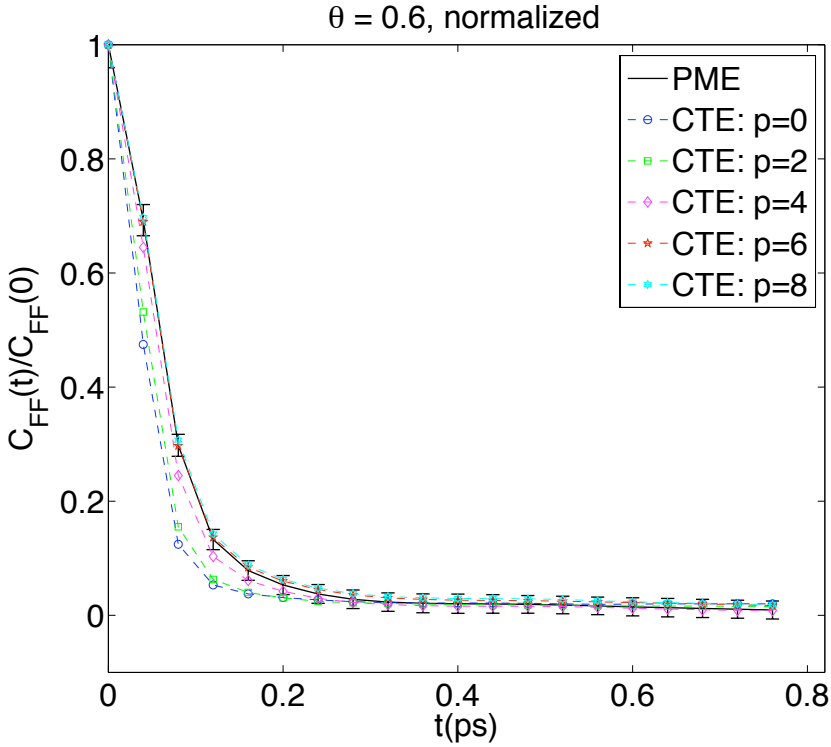


Figure 3.18: Autocorrelation function of force along molecular bond for ($\theta = 0.6$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

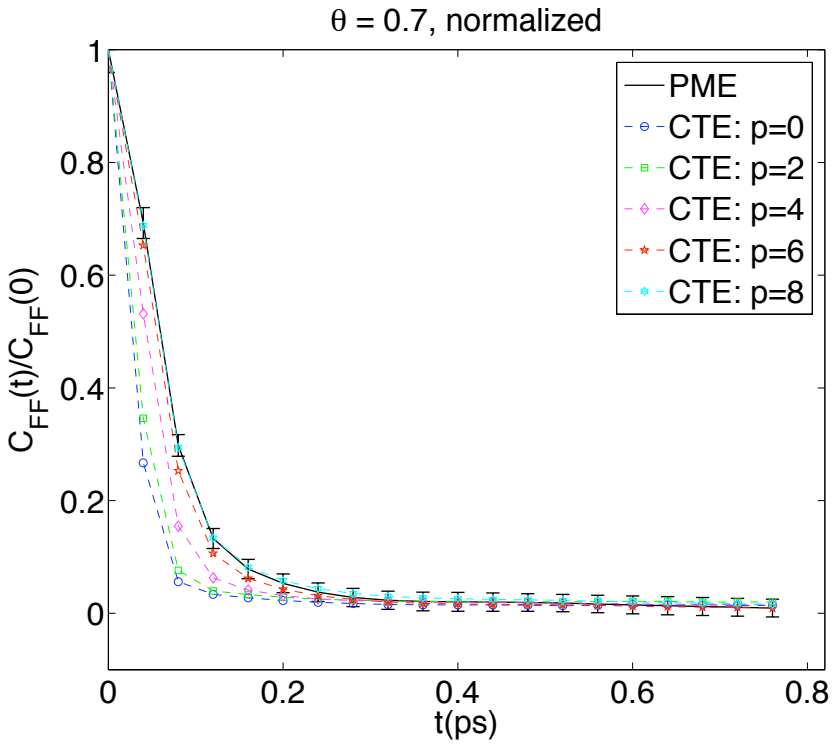


Figure 3.19: Autocorrelation function of force along molecular bond for ($\theta = 0.7$). The CTE result is within error bounds of the PME result for orders $p = 6$ and $p = 8$.

3.3.3 Comparisons for different system sizes

In addition to the structural and dynamical properties, we also compared the CTE method to the PME method for accuracy in energy, forces, series convergence and cpu time. We picked a regime of medium accuracy with the error at the cutoff specified to be 10^{-5}. These tests were run on an Intel Xeon cluster with the Intel Fortran compiler. We used the compiler optimization flag '-fast 'which performs an overall speedup of the code.

We computed the relative error in energy, relative error in the virial and the root means square error in the force. We label these quantities in the table as E_{rel}, V_{rel} and F_{rms} respectively.

The relative error in energy is given as

$$E_{rel} = \frac{\left|\text{Energy}_{\text{method}} - \text{Energy}_{\text{exact}}\right|}{\left|\text{Energy}_{\text{exact}}\right|} \tag{3.3.14}$$

where the energy refers to the sum of the real space potential energy V^r and the reciprocal space potential energy V^k.

The relative virial error gives a measure of the degree of convergence of the Ewald sum. V_{rel} for a method is defined as

$$V_{rel} = \frac{\left|\text{Energy}_{\text{method}} + \text{Virial}_{\text{method}}\right|}{\left|\text{Energy}_{\text{method}}\right|}. \tag{3.3.15}$$

Since the total potential energy is theoretically equal to the negative of the total potential energy virial, $V_{rel} = 0$ for an exact computation of the Ewald sum. Approximations of the Ewald sum are expected to have V_{rel} on the order of the error specified for the approximation.

The root mean square error in force for a system of N particles is defined as

$$F_{rms} = \left\{ \frac{\sum_{i=1}^{N} \left|\mathbf{f}_i^{\text{exact}} - \mathbf{f}_i^{\text{method}}\right|^2}{\sum_{i=1}^{N} \left|\mathbf{f}_i^{\text{exact}}\right|^2} \right\}^{1/2}. \tag{3.3.16}$$

For system sizes $N = 21952$, $N = 39304$ and $N = 74088$, the exact energy and forces for comparisons were computed using direct sum via the classical Ewald method with a cutoff in real space $r_c = \frac{L}{2}$ and a reciprocal space cutoff of $k_c = 5$. We were not able to compute exact energy and forces using the classical Ewald method for system sizes greater than 74088 because of memory limitations. The reciprocal space of the classical Ewald sum requires large memory allocation.

One goal of our project was to combine the CTE method with the PME method to form a hybrid CTE-PME method that would be faster than either CTE or PME. The CTE method employs the direct classical Ewald reciprocal sum while the CTE-PME hybrid employs PME for the reciprocal space sum.

We compared a CTE-PME hybrid to the original CTE method and the PME method in Figures 3.20,3.21 and 3.22 and found that the hybrid is faster than the original CTE method for small systems but slower for large systems for the regimes that were tested. In addition both the CTE method and the CTE-PME hybrid method in their current implementation are slower than the PME method.

The CTE method implemented in this chapter is a particle-cluster variant. In Chapter IV, we will present results generated with the cluster-cluster variant we have developed.

Table 3.2 and Table 3.3 present results generated for $N = 21952$ using the original CTE method and the CTE-PME hybrid method respectively. Table 3.4 and Table 3.5 present results generated for $N = 39304$ using the original CTE method and the CTE-PME hybrid method respectively. The results are presented for different cutoffs in real space, $r_c \in \{L/2, L/4, L/8\}$ which result in different cutoffs in reciprocal space k_c.

Table 3.2 presents comparison of CTE to PME. The CTE method is fastest for

real space cutoff, $r_c = \frac{L}{2} = 48.313$. Decreasing r_c and increasing the reciprocal space cutoff, k_c, to balance the work results in an increase in the overall cpu time as shown for $p = 0$ and $p = 8$. This is because although the cpu time for the real space decreases for smaller cutoffs r_c, the cpu time for the reciprocal space increases significantly with the associated increase in k_c. This is as expected since the standard reciprocal space employed for CTE is at best $O(N^{3/2})$. This situation is reversed for the CTE-PME hybrid as seen in Table 3.3 for $p = 8$. The PME method computes the reciprocal space sum very quickly and as such the overall cpu time is reduced.

Method	p	θ	r_c	k_c	E_{rel}	V_{rel}	F_{rms}	Time(s)
Classical Ewald			48.313	5		3.0E-6		10.788
PME	8		11.0	64	1.2E-6	1.5E-5	1.6E-3	1.703
CTE	0	0.7	48.313	5	9.3E-5	1.3E-4	2.0E-2	3.692
			24.156	11	9.4E-6	5.2E-5	3.0E-3	7.814
			12.078	21	4.6E-6	1.1E-4	1.6E-3	34.661
	2		48.313	5	9.7E-6	2.8E-4	1.2E-2	3.854
	4		48.313	5	2.4E-6	1.4E-4	4.6E-3	4.311
	6		48.313	5	4.3E-6	1.1E-4	2.4E-3	5.105
	8		48.313	5	5.3E-6	2.2E-4	1.8E-3	6.431
			24.156	11	4.3E-6	3.2 E-5	1.8E-3	8.900
			12.078	21	4.6E-6	1.1E-4	1.6E-3	34.784

Table 3.2: Comparison of Classical Ewald, PME and CTE for $N = 21952$ atoms

Method	p	θ	r_c	k_c	E_{rel}	V_{rel}	F_{rms}	Time(s)
Classical Ewald			48.313	5		3.0E-6		10.788
PME	8		11.0	64	1.2E-6	1.5E-5	1.6E-3	1.703
CTE-PME Hybrid	0	0.7	12.078	64	4.7E-7	3.0E-6	1.6E-3	2.706
	2		12.078	64	4.7E-7	3.0E-6	1.6E-3	2.655
	4		12.078	64	4.7E-7	3.0E-6	1.6E-3	2.73
	6		48.313	16	2.8E-4	8.4E-4	9.6E-3	5.416
			12.078	64	4.7E-7	3.0E-6	1.6E-3	2.784
	8		48.313	16	2.9E-4	7.3E-4	9.5E-3	6.649
			24.156	32	1.7E-3	9.5E-3	3.5E-2	4.891
			12.078	64	4.7E-7	3.0E-6	1.6E-3	2.863

Table 3.3: Comparison of Classical Ewald, PME and the CTE-PME hybrid for $N = 21952$ atoms

For the system with 21952 atoms, Table 3.3 shows that the errors for the $r_c =$

$\frac{L}{8} = 12.078$ does not with increasing order, p of the multipole approximation. This is because the small cutoff limits the number of multipole approximations and most of the clusters that are within the cutoff fail the multipole acceptability criterion test.

For the bigger system with 39304 atoms, Table 3.4 show in general a decrease in the relative error in energy and the root mean square force for increasing order p. Table 3.5 shows an increase in accuracy for $r_c = \frac{L}{8} = 14.666$ with increasing order, p. The larger cutoff in comparison to Table 3.3 for $N = 21952$ means that a significant number of clusters within the cutoff satisfy the MAC test and as a result increasing the order of the approximation improves the accuracy albeit only slightly.

Method	p	θ	r_c	k_c	E_{rel}	V_{rel}	F_{rms}	Time(s)
Classical Ewald			58.666	5		8.3E-5		34.055
PME	8		11.0	64	8.5E-6	3.8E-5	1.4E-3	3.099
CTE	0	0.7	58.666	5	3.7E-5	1.8E-4	6.5E-2	2.94
	2		58.666	5	4.4E-5	3.0E-4	2.5E-2	3.427
	4		58.666	5	3.4E-5	9.9E-5	8.7E-2	4.938
	6		58.666	5	1.3E-5	3.8E-5	3.4E-3	7.688
	8		58.666	5	1.7E-5	2.9E-5	1.8E-3	12.100

Table 3.4: Comparison of Classical Ewald, PME and CTE for $N = 39304$ atoms

Method	p	θ	r_c	k_c	E_{rel}	V_{rel}	F_{rms}	Time(s)
Classical Ewald			58.666	5		8.3E-5		34.055
PME	8		11.0	64	8.5E-6	3.8E-5	1.4E-3	3.099
CTE-PME Hybrid	0	0.7	58.666	16	1.2E-4	3.29E-4	6.5E-2	4.703
			29.333	32	1.0E-3	5.3E-3	4.7E-2	4.526
			14.666	64	2.2E-5	6.0E-5	5.1E-3	4.242
	6		58.666	16	1.4E-4	4.7E-4	9.0E-3	9.986
			29.333	32	1.1E-3	4.9E-3	3.0E-2	8.11
			14.666	64	1.8E-5	2.7E-6	3.6E-3	5.329
	8		58.666	16	1.4E-4	5.4E-3	8.6E-3	14.795
			29.333	32	1.1E-3	5.0E-3	3.0E-2	11.51
			14.666	64	1.4E-5	5.4E-5	2.2E-3	6.43

Table 3.5: Comparison of PME and CTE-PME Hybrid for $N = 39304$ atoms

Figures 3.20 and 3.21 show the timing comparisons, in computing total energy

and forces, for different orders of the CTE and CTE-PME methods respectively, to the PME method for a multipole acceptability criterion of $\theta = 0.7$. Clearly, PME is as fast or faster than CTE and the hybrid CTE-PME method for all orders.

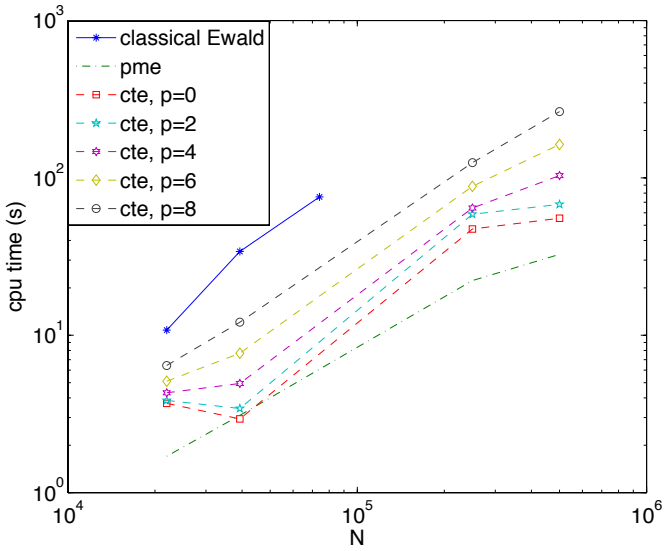


Figure 3.20: This figure compares the cpu time of CTE to PME and classical Ewald. Both PME and CTE are faster than the Classical Ewald with cutoff of $\frac{L}{2}$. The CTE cpu times are within an order of magnitude of the PME cpu time for all orders. Because of the large memory requirements of Classical Ewald, we were only able to obtain results for system sizes $N \leq 74088$.

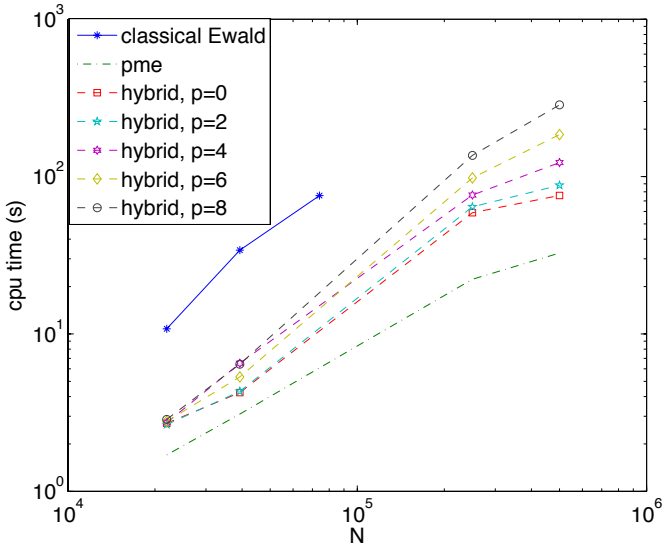
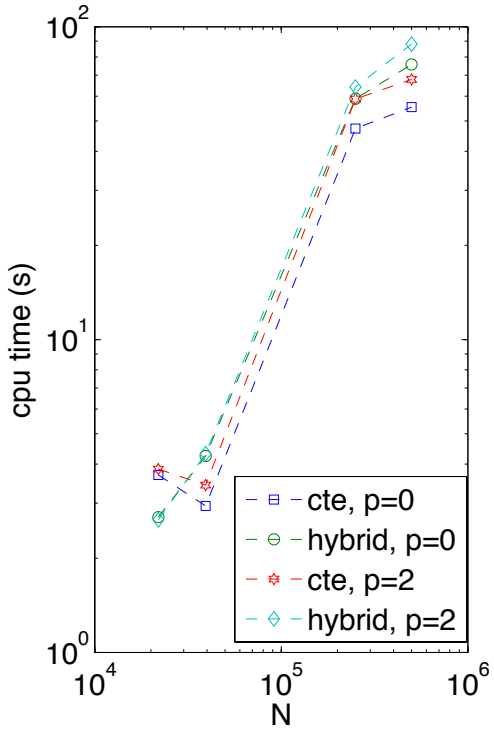
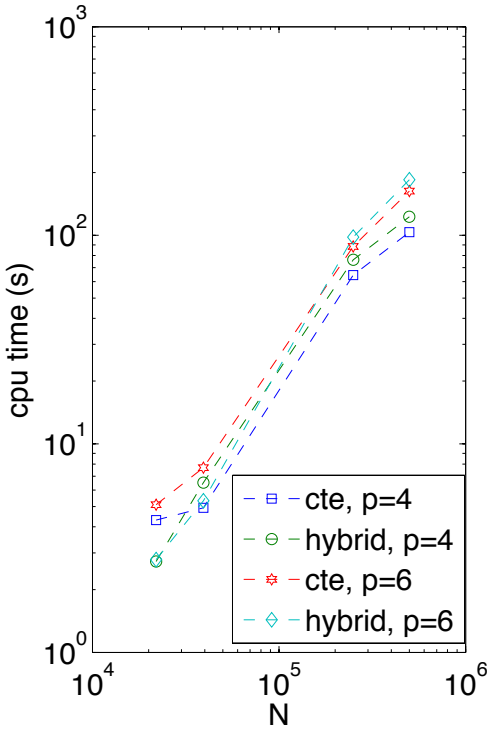


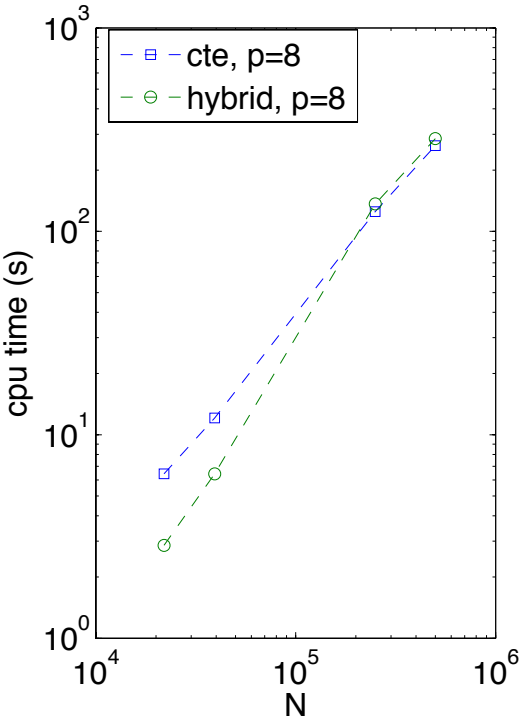
Figure 3.21: In this figure we compare the cpu time of classical Ewald, PME and CTE-PME hybrid. Again, all the algorithms are faster than Classical Ewald.



(a) $p = 0$ and $p = 2$



(b) $p = 4$ and $p = 6$



(c) $p = 8$

Figure 3.22: Figures (a), (b) and (c) compare CTE to CTE-PME hybrid for multipole orders $\{0, 2\}$, $\{4, 6\}$ and $\{8\}$ respectively. They all show that CTE-PME hybrid is faster than CTE for small systems while CTE is faster for larger systems.

Figure 3.22 shows that balancing the work between the real space and reciprocal space parts and employing the CTE-PME hybrid results in a faster algorithm than the original CTE algorithm for comparable accuracy for small systems. However for large systems the CTE-PME hybrid is not faster than the original CTE for comparable accuracy. In fact it is slower.

3.4 Conclusion

This chapter has presented comparisons of results from MD simulations of the Cartesian Treecode Ewald method to the particle-mesh Ewald method. The results show that the CTE method produces the same structural and dynamical properties as the PME method for different sets of parameters (θ, p). We also presented comparisons of a CTE-PME hybrid to PME in cpu time for an accuracy of 10^{-5} in the Ewald sum computation. We see that for the system sizes considered, PME is faster than the current implementation of the CTE and CTE-PME hybrid methods.

The treecode algorithm used here was a particle-cluster formulation. The next chapter will develop a new cluster-cluster treecode algorithm. It will show that for the same medium accuracy level, the cluster-cluster treecode formulation is faster than the particle-cluster formulation. This moves us closer to competing in cpu time with PME.

CHAPTER IV

Cluster-Cluster Cartesian Multipole Treecode Algorithm

We develop here a cluster-cluster Cartesian treecode algorithm as an alternative to the particle-cluster Cartesian treecode algorithms presented in [19, 16, 37, 38]. The goal is for the cluster-cluster algorithm to achieve the same accuracy as the particle-cluster algorithm while being faster.

The tree building and computation of moments are the same for both the cluster-cluster and particle-cluster algorithms. For both algorithms, the near-field interactions are computed directly. The far-field interactions are between a particle and a cluster for the particle-cluster algorithm, Figure 2.6. However, for the cluster-cluster algorithm, the far-field interaction is between two well separated clusters, Figure 4.1. Two clusters **A** and **B** with centers R apart and radii r_A and r_B respectively, are well separated if the multipole acceptance criterion (MAC)

$$\frac{r_A + r_B}{R} \leq \theta \tag{4.0.1}$$

is satisfied for some $\theta < 1$.

Cluster-cluster interactions form the basis of the Fast Multipole Method [26, 11] and the Cell Multipole Method [14, 15]. Duan and Krasny [17] have also used a cluster-cluster approach, based on Appel's [3] formulation, to compute the total

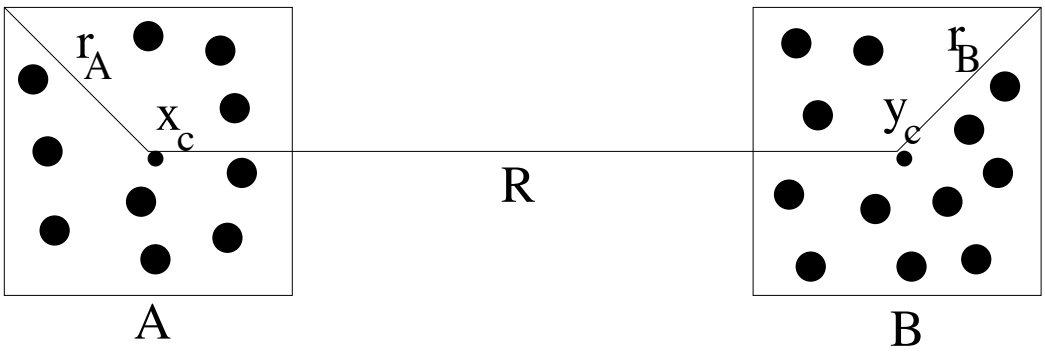


Figure 4.1: Cluster A interacts with cluster B by multipole approximation if the MAC, $\frac{r_A+r_B}{R} \leq \theta$ is satisfied.

potential energy in molecular systems. The cluster-cluster algorithm presented here can be viewed as a hybrid between the Fast Multipole Method approach and the Barnes-Hut [4] approach. Our approach is similar to the approach used in the Parallel Multipole Tree Algorithm (PMTA) [6, 7] although our algorithm employs a Taylor series expansion of the Coulomb potential for far-field expansions, unlike PMTA which we assume uses a spherical harmonic expansion of the potential. The multipole expansions for our algorithm are sped up by using recurrence relations to compute the Taylor coefficients needed for the expansion. PMTA, on the other hand, employs FFT-accelerated operations [6, 7] to speed up the multipole expansions. In addition, PMTA employs the same order of approximation in the two interacting clusters while our algorithm allows for the use of different orders of expansion in the two clusters.

4.1 Development

The cluster-cluster Cartesian Multipole Treecode Algorithm (CCMTA) we have developed is derived from particle-cluster CCMTA.

We consider the interactions between two clusters A and B of particles and focus on the effect of the particles in cluster B on the particles in cluster A. The direct

summation method is a particle-particle interaction of the form,

$$V(\mathbf{x}_i) = q_i \sum_{\mathbf{y}_j \in B} q_j \phi(\mathbf{x}_i - \mathbf{y}_j), \tag{4.1.1}$$

where $\mathbf{x}_i$ and $\mathbf{y}_j$ are particles in cluster A and cluster B respectively and q_i and q_j are their respective charges. The clusters are depicted in Figure 4.1.

4.1.1 Particle-Cluster

The particle-cluster method approximates the effect of a cluster B of far-field particles, $\mathbf{y}_j$, on a target particle i, in cluster A via a three dimensional Taylor approximation of $\phi(\mathbf{x}_i - \mathbf{y}_j)$. The multipole expansion is up to order q and it is centered at $\mathbf{y} = \mathbf{y}_c$, the geometric center of cluster B. The multipole expansion of the potential energy, $V(\mathbf{x}_i)$ at position $\mathbf{x}_i$ is given by

$$\begin{aligned} V(\mathbf{x}_i) &= q_i \sum_{\mathbf{y}_j \in B} q_j \phi(\mathbf{x}_i - \mathbf{y}_j) \\ &\approx q_i \sum_{\mathbf{l}=0}^{q} \frac{1}{\mathbf{l}!} D_{\mathbf{y}}^{\mathbf{l}} \phi(\mathbf{x}_i - \mathbf{y}_c) \sum_{\mathbf{y}_j \in B} q_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}} \\ &= q_i \sum_{||\mathbf{l}||=0}^{q} a_{\mathbf{l}}(\mathbf{x}_i - \mathbf{y}_c) \cdot M_B^{\mathbf{l}}, \end{aligned} \tag{4.1.2}$$

where

$$a_{\mathbf{l}}(\mathbf{x}_i - \mathbf{y}_c) = \frac{1}{\mathbf{l}!} D_{\mathbf{y}}^{\mathbf{l}} \phi(\mathbf{x}_i - \mathbf{y}_c), \tag{4.1.3}$$

is the lth order Taylor coefficients of the potential function $\phi(\mathbf{x})$ and

$$M_B^{\mathbf{l}} = \sum_{\mathbf{y}_j \in B} q_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}} \tag{4.1.4}$$

is the lth multipole moment of cluster B.

The force on particle i due to cluster B is derived as the negative gradient of

$V(\mathbf{x}_i)$ which yields,

$$F_{i,B} = -\nabla_{\mathbf{x}} V(\mathbf{x}_i) \tag{4.1.5}$$

$$F_{i,B} \approx -q_i \sum_{||\mathbf{l}||=0}^{q} \left(\nabla_{\mathbf{x}} a_{\mathbf{l}}(\mathbf{x}_i - \mathbf{y}_c)\right) M_B^{\mathbf{l}}, \tag{4.1.6}$$

$$= -q_i \begin{pmatrix} (l_1+1)a_{\mathbf{l}+\mathbf{e}_1} \\ (l_2+1)a_{\mathbf{l}+\mathbf{e}_2} \\ (l_3+1)a_{\mathbf{l}+\mathbf{e}_3} \end{pmatrix} M_B^{\mathbf{l}}. \tag{4.1.7}$$

4.1.2 Cluster-cluster approximation

The cluster-cluster algorithm proceeds just as particle-cluster by approximating the effect of cluster B on a particle, i , in cluster A as a multipole approximation of order q centered at $\mathbf{y} = \mathbf{y}_c$.

$$V(\mathbf{x}_i) = q_i \sum_{\mathbf{y}_j \in B} q_j \phi(\mathbf{x}_i - \mathbf{y}_j), \tag{4.1.8}$$

$$\approx q_i \sum_{||\mathbf{l}||=0}^{q} \frac{1}{\mathbf{l}!} D_{\mathbf{y}}^{\mathbf{l}} \phi(\mathbf{x}_i - \mathbf{y}_c) \sum_{\mathbf{y}_j \in B} q_j (\mathbf{y}_j - \mathbf{y}_c)^l, \tag{4.1.9}$$

$$= q_i \sum_{||\mathbf{l}||=0}^{q} \frac{1}{\mathbf{l}!} D_{\mathbf{y}}^{l} \phi(\mathbf{x}_i - \mathbf{y}_c) \cdot M_B^{\mathbf{l}}. \tag{4.1.10}$$

From the resultant, Equation 4.1.10, $D_{\mathbf{y}}^{\mathbf{l}} \phi(\mathbf{x}_i - \mathbf{y}_c)$, is in turn Taylor expanded about $\mathbf{x} = \mathbf{x}_c$, the center of cluster A to order p to give

$$V(\mathbf{x}_i) \approx q_i \sum_{||\mathbf{k}||=0}^{p} \frac{1}{\mathbf{k}!} \sum_{||\mathbf{l}||=0}^{q} \frac{1}{\mathbf{l}!} D_{\mathbf{x}}^{\mathbf{k}} D_{\mathbf{y}}^{\mathbf{l}} \phi(\mathbf{x}_c - \mathbf{y}_c) M_B^{\mathbf{l}} (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}. \tag{4.1.11}$$

Observe that

$$D_{\mathbf{x}}^{\mathbf{k}} \phi(\mathbf{x}_c - \mathbf{y}_c) = (-1)^{\mathbf{k}} D_{\mathbf{y}}^{\mathbf{k}} \phi(\mathbf{x}_c - \mathbf{y}_c), \tag{4.1.12}$$

which allows the dimensions of the operator differential operator D to be reduced

from six dimensions to three dimensions resulting in

$$
\begin{aligned}
(4.1.13)\quad V(\mathbf{x}_i) &\approx q_i \sum_{||\mathbf{k}||=0}^{p} \sum_{||\mathbf{l}||=0}^{q} \frac{1}{\mathbf{k}!}\frac{1}{\mathbf{l}!}(-1)^{\mathbf{k}} D_{\mathbf{y}}^{\mathbf{k}+\mathbf{l}}\phi(\mathbf{x}_c-\mathbf{y}_c)M_B^{\mathbf{l}}(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}} \\
&= q_i \sum_{\mathbf{k}=0}^{p} \sum_{||\mathbf{l}||=0}^{q} \frac{(\mathbf{k}+\mathbf{l})!}{\mathbf{k}!\cdot\mathbf{l}!}\frac{(-1)^{\mathbf{k}}}{(\mathbf{k}+\mathbf{l})!} D_{\mathbf{y}}^{\mathbf{k}+\mathbf{l}}\phi(\mathbf{x}_c-\mathbf{y}_c)M_B^{\mathbf{l}}(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}} \\
&= q_i \sum_{||\mathbf{k}||=0}^{p} \sum_{||\mathbf{l}||=0}^{q} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} \frac{(-1)^{\mathbf{k}}}{(\mathbf{k}+\mathbf{l})!} D_{\mathbf{y}}^{\mathbf{k}+\mathbf{l}}\phi(\mathbf{x}_c-\mathbf{y}_c)M_B^{\mathbf{l}}(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}} \\
&= q_i \sum_{||\mathbf{k}||=0}^{p} \left(\sum_{||\mathbf{l}||=0}^{q} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} (-1)^{||\mathbf{k}||} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c-\mathbf{y}_c)M_B^{\mathbf{l}} \right) (\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}} \\
(4.1.14)\qquad &= q_i \sum_{||\mathbf{k}||=0}^{p} b_{\mathbf{k}}((\mathbf{x}_c-\mathbf{y}_c),q,B)(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}},
\end{aligned}
$$

with

$$
(4.1.15)\qquad b_{\mathbf{k}}((\mathbf{x}_c-\mathbf{y}_c),q,B) = (-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=0}^{q} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c-\mathbf{y}_c)M_B^{\mathbf{l}}.
$$

Equation 4.1.14 is an approximation of the effect of cluster B on particle i in cluster A as a power series of order p with coefficients $b_{\mathbf{k}}$ centered at $\mathbf{x}_c$, the center of cluster A.

The force on particle i due to cluster B is again the negative of the gradient of the potential energy, $V(\mathbf{x}_i)$, given in Equation 4.1.14 to yield,

$$
\begin{aligned}
(4.1.16)\qquad F_{i,B} &\approx -q_i \sum_{||\mathbf{k}||=0}^{p} b_{\mathbf{k}}((\mathbf{x}_c-\mathbf{y}_c),q,B)\nabla_{\mathbf{x}}(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}} \\
(4.1.17)\qquad &= -q_i \sum_{\mathbf{k}=0}^{p} b_{\mathbf{k}}((\mathbf{x}_c-\mathbf{y}_c),q,B) \begin{pmatrix} k_1(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}-\mathbf{e}_1} \\ k_2(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}-\mathbf{e}_2} \\ k_3(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}-\mathbf{e}_3} \end{pmatrix}.
\end{aligned}
$$

By using the approximation to $\mathbf{F}_{i,B}$ in Equation 4.1.17, we approximate the far-field virial, V_s^r from Equation 2.4.17 due to the real space potential energy as,

$$V_s^r \approx -\frac{\alpha^2}{\sqrt{\pi}} \sum_{\bigcup_B} \sum_{i=1}^{N} q_i \mathbf{r}_{i,B} \cdot \sum_{||\mathbf{k}||=0}^{p} b_{\mathbf{k}}((\mathbf{x}_c - \mathbf{y}_c), q, B) \begin{pmatrix} k_1(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}-\mathbf{e}_1} \\ \\ k_2(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}-\mathbf{e}_2} \\ \\ k_3(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}-\mathbf{e}_3} \end{pmatrix}. \tag{4.1.18}$$

In short, we have effectively reduced the interactions between particles in cluster A and cluster B to an interaction between the centers of the clusters.

Since the cluster-cluster approach involves more approximation than the particle-cluster we expect the cluster-cluster approach to be less accurate for given truncation orders p in cluster A and q in cluster B. We will now show that for the Coulomb potential

$$\phi(\mathbf{r}) = \frac{1}{\mathbf{r}}, \tag{4.1.19}$$

the cluster-cluster approach has error on the same order as the particle-cluster approach.

4.2 Error Analysis

We restrict the analysis to the error in approximating the Coulomb potential energy

$$V(\mathbf{x}_i) = q_i \sum_{j \in B} q_j \frac{1}{|\mathbf{x}_i - \mathbf{x}_j|} = q_i \sum_{j \in B} q_j \phi(\mathbf{x}_i - \mathbf{x}_j), \tag{4.2.1}$$

of a particle at position $\mathbf{x}_i$, in cluster A due to particles at $\mathbf{y}_j$, in cluster B using a cluster-cluster approximation.

First we look at the error associated with the particle-cluster approximation due to the multipole expansion centered at $\mathbf{y}_c$, the center of cluster B. The multipole expansion to order q in cluster B is given by Equation 4.1.2. Here we use an expansion to order $q-1$. The recurrence relation for the Taylor coefficients, $a_{\mathbf{l}}$, in the Taylor expansion of the potential function $\phi(\mathbf{x})$ in Equation 4.2.1,

$$||\mathbf{l}|||\mathbf{x}|^2 a_{\mathbf{l}} + (2||\mathbf{l}|| - 1)\sum_{i=1}^{3}(x_i - y_{c_i})a_{\mathbf{l}-\mathbf{e_i}} + (||\mathbf{l}|| - 1)\sum_{i=1}^{3} a_{\mathbf{l}-\mathbf{2e_i}} = 0, \tag{4.2.2}$$

with $\mathbf{x} = \mathbf{x}_i - \mathbf{y}_c$, has been derived by several authors [38, 17] with the same procedure used in deriving Equation 2.4.66. Here, the bold type $\mathbf{x}_i$ refers to a vector while the x_i in the refers to a coordinate of $\mathbf{x}_i$. The analysis given here follows closely the development in [37, 38]. With a $(q-1)$st expansion, the sum of the neglected terms from the expansion in Equation 4.1.2 is given as

$$q_i \sum_{||\mathbf{l}||\geq q} a_{\mathbf{l}}(\mathbf{x}_i - \mathbf{y}_c)\cdot M_B^{\mathbf{l}} = q_i \sum_{j\in B} q_j \sum_{n\geq q} A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j), \tag{4.2.3}$$

with

$$A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j) = \sum_{||\mathbf{l}||=n} a_{\mathbf{l}}(\mathbf{x}_i - \mathbf{y}_c)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}}. \tag{4.2.4}$$

We will derive a bound for $A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j)$ which will in turn bound the error in the approximation. First, we find a recurrence relation for A_n.

Multiplying Equation 4.2.2 by $(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}}$ results in

(4.2.5)

$$||\mathbf{l}|||\mathbf{x}|^2 a_{\mathbf{l}}\cdot(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{l}}+(2||\mathbf{l}||-1)\sum_{i=1}^{3}(x_i-y_{c_i})a_{\mathbf{l}-\mathbf{e_i}}\cdot(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{l}}+(||\mathbf{l}||-1)\sum_{i=1}^{3} a_{\mathbf{l}-\mathbf{2e_i}}\cdot(\mathbf{y}_j-\mathbf{y}_c)^{\mathbf{l}} = 0,$$

which is then summed over $||\mathbf{l}|| = n$ to produce the equation

(4.2.6)

$$\sum_{||\mathbf{l}||=n} \{||\mathbf{l}|||\mathbf{x}|^2 a_{\mathbf{l}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}}\} + \sum_{||\mathbf{l}||=n} \left\{ (2||\mathbf{l}|| - 1) \sum_{i=1}^{3} (x_i - y_{c_i}) a_{\mathbf{l}-\mathbf{e_i}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}} \right\}$$
$$+ \sum_{||\mathbf{l}||=n} \left\{ (||\mathbf{l}|| - 1) \sum_{i=1}^{3} a_{\mathbf{l}-\mathbf{2e_i}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}} \right\} = 0.$$

By simplifying we arrive at

$$(4.2.7) \qquad n|\mathbf{x}|^2 \sum_{||\mathbf{l}||=n} a_{\mathbf{l}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}} + (2n-1) \sum_{||\mathbf{l}||=n} \sum_{i=1}^{3} (x_i - y_{c_i}) a_{\mathbf{l}-\mathbf{e_i}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}}$$
$$+(n-1) \sum_{||\mathbf{l}||=n} \sum_{i=1}^{3} a_{\mathbf{l}-\mathbf{2e_i}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}} = 0,$$

which can be rewritten as

$$(4.2.8) \qquad n|\mathbf{x}|^2 A_n + (2n-1) \sum_{i=1}^{3} (x_i - y_{c_i}) \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{e}_i} \sum_{||\mathbf{l}||=n} a_{\mathbf{l}-\mathbf{e_i}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}-\mathbf{e}_i} +$$
$$(n-1) \sum_{i=1}^{3} (\mathbf{y}_j - \mathbf{y}_c)^{2\mathbf{e}_i} \sum_{||\mathbf{l}||=n} a_{\mathbf{l}-\mathbf{2e_i}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}-2\mathbf{e}_i} = 0,$$

where Equation 4.2.4 has been applied. In addition, from Equation 4.2.4 we generate the identity,

$$(4.2.9) \qquad A_{n-k}(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j) = \sum_{||\mathbf{l}||=n-k} a_{\mathbf{l}} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}}$$
$$(4.2.10) \qquad = \sum_{||\mathbf{l}||=n} a_{\mathbf{l}-k\mathbf{e}_i} \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{l}-k\mathbf{e}_i}.$$

Then, by using Equation 4.2.10, Equation 4.2.8 becomes

(4.2.11)

$$n|\mathbf{x}|^2 A_n + (2n-1) A_{n-1} \sum_{i=1}^{3} (x_i - y_{c_i}) \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{e}_i} + (n-1) A_{n-2} \sum_{i=1}^{3} (\mathbf{y}_j - \mathbf{y}_c)^{2\mathbf{e}_i} = 0.$$

Let

$$(4.2.12) \qquad \sum_{i=1}^{3} (x_i - y_{c_i}) \cdot (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{e}_i} = (\mathbf{x}_i - \mathbf{y}_c) \cdot (\mathbf{y}_j - \mathbf{y}_c) = \alpha$$

and

$$(4.2.13)\quad \sum_{i=1}^{3}(\mathbf{y}_j-\mathbf{y}_c)^{2\mathbf{e}_i}=\sum_{i=1}^{3}(\mathbf{y}_{j_i}-\mathbf{y}_{c_i})\cdot(\mathbf{y}_{j_i}-\mathbf{y}_{c_i})=(\mathbf{y}_j-\mathbf{y}_c)\cdot(\mathbf{y}_j-\mathbf{y}_c)=\beta^2,$$

implying that, $\beta=|\mathbf{y}_j-\mathbf{y}_c|$. With these definitions, Equation 4.2.11 then yields the recurrence relation

$$(4.2.14)\qquad n|\mathbf{x}|^2A_n+(2n-1)\alpha A_{n-1}+(n-1)\beta^2A_{n-2}=0\qquad \text{for } n\geq 2.$$

In order to bound A_n, we will first write A_n in terms of the nth Legendre polynomial, $P_n(x)$.

The recurrence relation for the Legendre polynomial in a single dimension is given as [1]

$$(4.2.15)\qquad nP_n(x)-(2n-1)xP_{n-1}(x)+(n-1)P_{n-2}(x)=0\qquad \text{for } n\geq 2,$$

with $P_0(x)=1$, $P_1(x)=x$ and $P_n(x)=(-1)^nP_n(-x)$. Using the last condition, Equation 4.2.15 can be rewritten as

$$(4.2.16)\quad n(-1)^nP_n(-x)-(2n-1)x(-1)^{n-1}P_{n-1}(-x)+(n-1)(-1)^{n-2}P_{n-2}(-x)=0,$$

which results in

$$(4.2.17)\qquad nP_n(-x)+(2n-1)xP_{n-1}(-x)+(n-1)P_{n-2}(-x)=0$$

after multiplying through by $(-1)^n$.

Set $x=\frac{\alpha}{\beta|\mathbf{x}|}$ and $h=\frac{\beta}{|\mathbf{x}|}$. Then multiplying through Equation 4.2.17 by h^n results in

(4.2.18)

$$n\cdot h^nP_n\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right)+(2n-1)\frac{\alpha}{\beta|\mathbf{x}|}\cdot\frac{\beta}{|\mathbf{x}|}\cdot h^{n-1}P_{n-1}\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right)+(n-1)\left(\frac{\beta}{|\mathbf{x}|}\right)^2\cdot h^{n-2}P_{n-2}\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right)=0,$$

which is in turn multiplied by $|\mathbf{x}|^2$ and divided by $\frac{1}{|\mathbf{x}|}$ to arrive at

(4.2.19)
$$n|\mathbf{x}|^2\frac{h^n}{|\mathbf{x}|}P_n\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right)+(2n-1)\alpha\frac{h^{n-1}}{|\mathbf{x}|}P_{n-1}\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right)+(n-1)\beta^2\frac{h^{n-2}}{|\mathbf{x}|}P_{n-2}\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right)=0,$$

for $n \geq 2$.

Comparison of Equation 4.2.19 to Equation 4.2.14 leads to

$$A_n = \frac{h^n}{|\mathbf{x}|}P_n\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right) \qquad \text{for } n \geq 2, \tag{4.2.20}$$

if the equality holds for the $n = 0$ and $n = 1$ terms.

For $n = 0$, from Equation 4.2.4

$$A_n = A_0 = a_0 = \phi(\mathbf{x}) = \frac{1}{|\mathbf{x}|} \tag{4.2.21}$$

and

$$\frac{h^n}{|\mathbf{x}|}P_n\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right) = \frac{h^0}{|\mathbf{x}|}P_0\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right) = \frac{1}{|\mathbf{x}|}. \tag{4.2.22}$$

For $n = 1$ using Equation 4.2.14,

$$A_n = A_1 = -\frac{\alpha}{|\mathbf{x}|^2}A_0 = -\frac{\alpha}{|\mathbf{x}|^3}, \tag{4.2.23}$$

and

$$\frac{h^n}{|\mathbf{x}|}P_n\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right) = \frac{h^1}{|\mathbf{x}|}P_1\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right) = \frac{\beta}{|\mathbf{x}|}\cdot\frac{1}{|\mathbf{x}|}\cdot\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right) = -\frac{\alpha}{|\mathbf{x}|^3}. \tag{4.2.24}$$

Thus,

$$A_n = \frac{h^n}{|\mathbf{x}|}P_n\left(-\frac{\alpha}{\beta|\mathbf{x}|}\right) \qquad \text{for } n \geq 2, \tag{4.2.25}$$

with $A_0 = \frac{1}{|\mathbf{x}|}$ and $A_1 = -\frac{\alpha}{|\mathbf{x}|^3}$. From Equation 4.2.12 and Equation 4.2.13, we get the relation

$$\left|\frac{\alpha}{\beta|\mathbf{x}|}\right| = \left|\frac{(\mathbf{x}_i - \mathbf{y}_c)\cdot(\mathbf{y}_j - \mathbf{y}_c)}{|\mathbf{y}_j - \mathbf{y}_c||\mathbf{x}_i - \mathbf{y}_c|}\right| \leq 1, \tag{4.2.26}$$

and with $P_n(1) = 1$, the bound on A_n is given as

$$|A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j)| = \left| \frac{1}{|\mathbf{x}|} \left(\frac{\beta}{|\mathbf{x}|} \right)^n \right| \leq \frac{1}{|\mathbf{x}|} \left(\frac{|\mathbf{y}_j - \mathbf{y}_c|}{|\mathbf{x}|} \right)^n. \tag{4.2.27}$$

The particle-cluster multipole approximation is only implemented when

$$|\mathbf{y}_j - \mathbf{y}_c| < |\mathbf{x}_i - \mathbf{y}_c|,$$

thus the bound on $|A_n|$ decreases with increasing n.

We will use this result to bound the error for the cluster-cluster method where the order of the expansion is $p-1$ in cluster A and $q-1$ in cluster B.

For the cluster-cluster approximation, the potential energy from Equation 4.1.13 is approximated as

$$V(\mathbf{x}_i) \approx q_i \sum_{||\mathbf{k}||=0}^{p-1} \left((-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=0}^{q-1} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \right) (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}. \tag{4.2.28}$$

The exact potential can also be written as a power series of infinite order in both cluster A and cluster B where

$$\begin{aligned} V(\mathbf{x}_i) &= q_i \sum_{||\mathbf{k}||=0}^{\infty} (-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=0}^{\infty} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \\ &= q_i \sum_{||\mathbf{k}||=0}^{p-1} (-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=0}^{q-1} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \\ &+ q_i \sum_{||\mathbf{k}||=0}^{p-1} (-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=q}^{\infty} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \\ &+ q_i \sum_{||\mathbf{k}||=p}^{\infty} (-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=0}^{\infty} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}. \end{aligned} \tag{4.2.29}$$

The error, $E(\mathbf{x}_i)$ is the difference between Equation 4.2.29 and Equation 4.2.28 and is given as

$$E(\mathbf{x}_i) = q_i \sum_{||\mathbf{k}||=0}^{p-1} (-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=q}^{\infty} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \tag{4.2.30}$$

$$
\begin{aligned}
&+ \quad q_i \sum_{||\mathbf{k}||=p}^{\infty} (-1)^{\mathbf{k}} \sum_{||\mathbf{l}||=0}^{\infty} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \\
&= \quad q_i \sum_{||\mathbf{l}||=q}^{\infty} M_B^{\mathbf{l}} \sum_{||\mathbf{k}||=0}^{\infty} (-1)^{\mathbf{k}} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \\
&+ \quad q_i \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \sum_{||\mathbf{k}||=p}^{\infty} (-1)^{\mathbf{k}} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \cdot (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}}.
\end{aligned}
$$

The inner sum of the first term of the last equality in Equation 4.2.30, runs from $||\mathbf{k}|| = 0$ to $||\mathbf{k}|| = \infty$, which means that the expansion in cluster A is exact. The error is only for the expansion in cluster B. This exactness in the cluster A expansion allows us to write

(4.2.31)
$$
q_i \sum_{||\mathbf{l}||=q}^{\infty} M_B^{\mathbf{l}} \sum_{||\mathbf{k}||=0}^{\infty} (-1)^{\mathbf{k}} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c)(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \equiv q_i \sum_{||\mathbf{l}||=q}^{\infty} a_{\mathbf{l}}(\mathbf{x}_i - \mathbf{y}_c) M_B^{\mathbf{l}}.
$$

From Equation 4.2.3,

$$
\begin{aligned}
(4.2.32) \qquad |q_i \sum_{||\mathbf{l}||=q}^{\infty} a_{\mathbf{l}}(\mathbf{x}_i - \mathbf{y}_c) \cdot M_B^{\mathbf{l}} \quad | &\leq \quad |q_i| \sum_{j \in B} |q_j| \sum_{n \geq q} |A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j)|, \\
&= \quad |q_i| Q \sum_{n \geq q} |A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j)|,
\end{aligned}
$$

and

$$
(4.2.33) \qquad Q = \sum_{j \in B} |q_j|.
$$

Then, the error from Equation 4.2.30 for the cluster-cluster approximation is bounded by

(4.2.34)
$$
|E(\mathbf{x}_i)| \leq |q_i| Q \sum_{n \geq q} |A_n(\mathbf{x}_i, \mathbf{y}_c, \mathbf{y}_j)| + |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \sum_{||\mathbf{k}||=p}^{\infty} \left((-1)^{\mathbf{k}} \binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}} a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c - \mathbf{y}_c) \right) (\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \right|.
$$

The first term is the error we get from a particle-cluster approximation which for the cluster-cluster algorithm is performed in cluster B and whose magnitude depends on

q. The second term is the error that comes from expanding the resultant from the particle-cluster approximation about the center of cluster A to obtain the $(p-1)$st order power series. As such, the magnitude of the second term depends on both p and q. The presence of the second term shows that the error for the cluster-cluster approximation is worse than the error for the particle-cluster approximation for a given p and q.

We will now derive a bound for the error and show that $|E(\mathbf{x}_i)| \to 0$ when $p, q \to \infty$. We first derive a bound for the first term. We make use of the bound on $|A_n|$ in Equation 4.2.27. Then

(4.2.35)

$$\begin{aligned} |q_i|Q\sum_{n\geq q}|A_n(\mathbf{x}_i,\mathbf{y}_c,\mathbf{y}_j)| &\leq |q_i|Q\sum_{n\geq q}\frac{1}{|\mathbf{x}_i-\mathbf{y}_c|}\left(\frac{|\mathbf{y}_j-\mathbf{y}_c|}{|\mathbf{x}_i-\mathbf{y}_c|}\right)^n, \\ &= \frac{|q_i|Q}{|\mathbf{x}_i-\mathbf{y}_c|}\frac{1}{\left(1-\frac{|\mathbf{y}_j-\mathbf{y}_c|}{|\mathbf{x}_i-\mathbf{y}_c|}\right)}\left(\frac{|\mathbf{y}_j-\mathbf{y}_c|}{|\mathbf{x}_i-\mathbf{y}_c|}\right)^q, \\ &= \frac{|q_i|Q}{|\mathbf{x}_i-\mathbf{y}_c|-|\mathbf{y}_j-\mathbf{y}_c|}\left(\frac{|\mathbf{y}_j-\mathbf{y}_c|}{|\mathbf{x}_i-\mathbf{y}_c|}\right)^q. \end{aligned}$$

To derive the bound on the second term in Equation 4.2.34, we note that for a finite q

(4.2.36)

$$\begin{aligned} & |q_i|\left|\sum_{||\mathbf{l}||=0}^{q-1}M_B^{\mathbf{l}}\sum_{||\mathbf{k}||=p}^{\infty}\left((-1)^{\mathbf{k}}\binom{\mathbf{k}+\mathbf{l}}{\mathbf{l}}a_{\mathbf{k}+\mathbf{l}}(\mathbf{x}_c-\mathbf{y}_c)\right)(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}}\right| \\ \equiv\ & |q_i|\left|\sum_{||\mathbf{l}||=0}^{q-1}M_B^{\mathbf{l}}\right|\left|\sum_{||\mathbf{k}||=p}^{\infty}(-1)^{\mathbf{k}}a_{\mathbf{k}}(\mathbf{x}_c-\mathbf{y}_c)(\mathbf{x}_i-\mathbf{x}_c)^{\mathbf{k}}\right|. \end{aligned}$$

Recall from Equation 4.1.3 that

$$\begin{aligned} a_{\mathbf{k}}(\mathbf{x}_c - \mathbf{y}_c) &= \frac{1}{\mathbf{k}!} D^{\mathbf{k}}_{\mathbf{y}} \phi(\mathbf{x}_c - \mathbf{y}_c) \\ &= (-1)^{\mathbf{k}} \frac{1}{\mathbf{k}!} D^{\mathbf{k}}_{\mathbf{y}} \phi(\mathbf{y}_c - \mathbf{x}_c) \\ &= (-1)^{\mathbf{k}} a_{\mathbf{k}}(\mathbf{y}_c - \mathbf{x}_c). \end{aligned} \tag{4.2.37}$$

Then from Equations (4.2.36), (4.2.4) and (4.2.27),

(4.2.38)

$$\begin{aligned} |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right| \left| \sum_{||\mathbf{k}||=p}^{\infty} (-1)^{\mathbf{k}} a_{\mathbf{k}}(\mathbf{x}_c - \mathbf{y}_c)(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \right| \\ = |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right| \left| \sum_{||\mathbf{k}||=p}^{\infty} (-1)^{2\mathbf{k}} a_{\mathbf{k}}(\mathbf{y}_c - \mathbf{x}_c)(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \right|, \\ = |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right| \left| \sum_{||\mathbf{k}||=p}^{\infty} a_{\mathbf{k}}(\mathbf{y}_c - \mathbf{x}_c)(\mathbf{x}_i - \mathbf{x}_c)^{\mathbf{k}} \right|, \\ = |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right| \left| \sum_{n \geq p} A_n(\mathbf{y}_c, \mathbf{x}_c, \mathbf{x}_i) \right|, \\ \leq |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right| \sum_{n \geq p} \frac{1}{|\mathbf{y}_c - \mathbf{x}_c|} \left(\frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|} \right)^p, \\ = |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right| \frac{1}{|\mathbf{y}_c - \mathbf{x}_c|} \frac{1}{\left(1 - \frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|}\right)} \left(\frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|} \right)^p \\ = |q_i| \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right| \frac{1}{|\mathbf{y}_c - \mathbf{x}_c| - |\mathbf{x}_i - \mathbf{x}_c|} \left(\frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|} \right)^p, \\ = \frac{|q_i| Q_{MB}}{|\mathbf{y}_c - \mathbf{x}_c| - |\mathbf{x}_i - \mathbf{x}_c|} \left(\frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|} \right)^p, \end{aligned}$$

with

$$Q_{MB} = \left| \sum_{||\mathbf{l}||=0}^{q-1} M_B^{\mathbf{l}} \right|. \tag{4.2.39}$$

Then from Equation 4.2.35 and Equation 4.2.38, the bound on $|E(\mathbf{x}_i)|$ is given by

(4.2.40)
$$|E(\mathbf{x}_i)| \leq \frac{|q_i|Q}{|\mathbf{x}_i - \mathbf{y}_c| - |\mathbf{y}_j - \mathbf{y}_c|} \left(\frac{|\mathbf{y}_j - \mathbf{y}_c|}{|\mathbf{x}_i - \mathbf{y}_c|} \right)^q + \frac{|q_i|Q_{MB}}{|\mathbf{y}_c - \mathbf{x}_c| - |\mathbf{x}_i - \mathbf{x}_c|} \left(\frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|} \right)^p.$$

Taking the limit of $|E(\mathbf{x}_i)|$ with $p, q \to \infty$ yields

(4.2.41)
$$\begin{aligned} \lim_{p,q\to\infty} |E(\mathbf{x}_i)| &\leq \frac{|q_i|Q}{|\mathbf{x}_i - \mathbf{y}_c| - |\mathbf{y}_j - \mathbf{y}_c|} \lim_{q\to\infty} \left(\frac{|\mathbf{y}_j - \mathbf{y}_c|}{|\mathbf{x}_i - \mathbf{y}_c|} \right)^q \\ &+ \frac{|q_i|Q_{MB}}{|\mathbf{y}_c - \mathbf{x}_c| - |\mathbf{x}_i - \mathbf{x}_c|} \lim_{p\to\infty} \left(\frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|} \right)^p, \\ &= 0, \end{aligned}$$

since

(4.2.42)
$$\frac{|\mathbf{y}_j - \mathbf{y}_c|}{|\mathbf{x}_i - \mathbf{y}_c|} < 1.$$

and

(4.2.43)
$$\frac{|\mathbf{x}_i - \mathbf{x}_c|}{|\mathbf{y}_c - \mathbf{x}_c|} < 1.$$

Thus, $|E(\mathbf{x}_i)| \to 0$ as $p, q \to \infty$.

4.3 Algorithm Description

We will now provide a detailed description of the cluster-cluster algorithm (CCMTA). We have developed two versions of CCMTA, *leaf-cluster*, and *leaf-leaf*. The *leaf-cluster* algorithm proceeds as particle-cluster with the leaves (the last and smallest clusters of a branch of the tree) acting as particles. A leaf interacts with other leaves and other bigger clusters in the tree. The interaction in the *leaf-leaf* algorithm is similar to that of direct summation. For *leaf-leaf*, interactions are limited to between leaves so there is no interaction between a leaf and a bigger cluster. The *leaf-cluster*

algorithm is faster than the *leaf-leaf* algorithm in our present implementation of the two algorithms. As such the results presented in this chapter were generated with the *leaf-cluster* algorithm. In chapter VI, we will give a brief outline of a possible way to speed up the *leaf-leaf* algorithm.

Both algorithms begin by creating the same hierarchical oct-tree with a recursive subroutine. The subroutine takes in a number of particles, N, and the coordinates of the box that contains these particles. It then checks whether $N > N_o$, where N_o is the maximum number of particles in allowed in a leaf. If $N > N_o$, the cluster of particles is divided into at most eight children, however if $N \leq N_o$, then the cluster is labelled as a leaf. An alternative way for deciding when to divide a cluster is to specify the maximum number of levels in a tree. Then a cluster is divided only if it is on a level higher level than the level specified for leaves.

Each newly created cluster computes and stores its geometric center, moments and the coordinates of its bounding box. In addition, if the cluster is also a leaf, then it stores its unique leaf number used to identify that leaf. The unique leaf numbers range from 1 to the total number of leaves. The initial call to the subroutine is with the full system with all the particles.

Equation 4.1.14 and Equation 4.1.17 give the prescription for computing the potential on a particle at $\mathbf{x}_i$ and the force on the particle respectively. The criterion for determining whether a particular leaf-cluster interaction shown in Figure 4.1 is far-field is chosen to be,

$$\frac{\mathbf{r_A} + \mathbf{r_B}}{R} \leq \theta, \tag{4.3.1}$$

where $\mathbf{r_A}$ and $\mathbf{r_B}$ are the radii of leaf A and cluster B respectively, R is the distance between the centers of A and B and $\theta < 1$. This criterion, also used in [17], was found empirically to yield better efficiency for CCMTA than the original particle-cluster

criterion, $\frac{\mathbf{r_B}}{R}$.

We have found CCMTA to be at about twice as fast as a particle-cluster multipole treecode algorithm for the free space Coulomb potential with comparable accuracy. This decrease in cpu time for CCMTA can be attributed to several efficiencies inherent to the algorithm.

- For CCMTA, the $a_{\mathbf{k}}(\mathbf{x}_c - \mathbf{y}_c)$ coefficients are computed only once per leaf unlike particle-cluster where the coefficients are computed for each particle in a cluster.
- The $\mathbf{b_k}$ coefficients in Equation 4.1.14 and defined in Equation 4.1.15 are the same for each particle in a leaf. Hence, these $\mathbf{b_k}$'s are summed up over all of a leaf's multipole interactions before computing the power series in Equation 4.1.14 for each particle. Say a leaf interacts via multipole approximation with Nm clusters. By summing up the $\mathbf{b_k}$ coefficients for all Nm clusters, we only compute one power series for each particle instead of $N \cdot m$. Each pth order power series has $O(p^3)$ multiplications, hence we perform $O(p^3)$ operations instead of $O(N \cdot m \cdot p^3)$ operations.
- The power series in Equation 4.1.14 cost $O(p^3)$ to evaluate because it is efficiently computed with a 3-dimensional Horner's rule. A naive computation will cost $O(p^4)$.
- We are also able to employ Newton's 3rd law, that action and reaction are equal and opposite, for the direct summation in the near-field which is not possible for particle-cluster.

4.3.1 Leaf-Cluster Algorithm

The subroutine `Compute_Interaction` computes the potential energy and the force on particles.

```
Subroutine Compute_Interaction
   Do i = 1, Number of leaves
     b_k = 0
     Call Compute_b_k(leaf.i,root,b_k) to compute b_k coefficients
     Compute Interactions between particles in leaf.i directly
     Call Compute_Power_Series(leaf.i,b_k) to compute potential energy
     and forces using power series.
   End Do
   Total Potential = 1/2 Total Potential
 End Subroutine Compute_Interaction
```

Figure 4.2: The subroutine for computing interactions in the *leaf-cluster* algorithm.

The subroutine `Compute_`$\mathbf{b_k}$`(leaf.A,cluster.B,`$\mathbf{b_k}$`)` is a recursive routine that takes in the target leaf A, the cluster B and the coefficients $\mathbf{b_k}$. It sums up the $\mathbf{b_k}$ over all the multipole interactions of a leaf. It also computes the potential energy and forces in the near-field.

```
Recursive Subroutine Compute_b_k(leaf.A,cluster.B,b_k)
   If multipole acceptability criterion is satisfied
     Compute b_k due to cluster.B and add to the present b_k.
   else
     if cluster.B has no children
     If leaf number (leaf.A) < leaf number (cluster.B)
       Compute interaction between leaf.A and cluster.B by direct
       summation
     else
       Do i = 1, number of children of cluster.B
         Call Compute_b_k(leaf.A,cluster.B.child(i),b_k)
       End Do
     end if
   End if
End Subroutine Compute_b_k
```

Figure 4.3: The subroutine for computing the $b_{\mathbf{k}}$ coefficients for the power series in the *leaf-cluster* algorithm.

When `Compute_`$\mathbf{b_k}$ decides that a cluster.B is a leaf, we perform a further check `leaf number of leaf.A` $<$ `leaf number of cluster.B` to ensure that we do not compute the direct interactions twice. This is because if `leaf number (leaf.A)`

$>$ `leaf number (leaf.B)`, then we would have accounted for the interactions already using Newton's third law, since the loop in Compute_Interaction goes from the lowest leaf number to the highest. We can use Newton's third law because our system is uniformly distributed. If a target leaf A interacts with a leaf B during the subroutine call `Compute_`$\mathbf{b_k}$`(leaf.A,root,`$\mathbf{b_k}$`)` in Figure 4.2, then the call `Compute_`$\mathbf{b_k}$`(leaf.B,root,`$\mathbf{b_k}$`)` where leaf B is the target leaf will result also result in an interaction between leaf A and leaf B. To avoid duplicating the interactions we check the leaf numbers of the leaves. The interaction is performed at the first instance when it occurs and avoided at the second instance. The case when `leaf number of leaf.A = leaf number of cluster.B` is handled inside Compute_Interaction where the interactions between particles in a leaf are computed directly, also making use of Newton's third law.

The part of potential energy and forces contributed by the far-field multipole interactions are computed using power series of order p, by the subroutine `Compute_Power_Series` which employs a three-dimensional Horner's rule. The power series for the force follows the same pattern as the potential energy with the order of the power series for the force in Equation 4.1.17 one less than the order of power series for the potential energy in Equation 4.1.14. In order to be able to use the same loop for both the potential energy and the force, when $k_1 = 0$, or $k_2 = 0$, or $k_3 = 0$, $x_{i_1} - x_{c_1}$, or $x_{i_2} - x_{c_2}$, or $x_{i_3} - x_{c_3}$ is set to 1 respectively. This also avoids potential division by small numbers which can cause severe numerical instabilities. `Compute_Power_Series` takes in a leaf and the summed up $\mathbf{b_k}$ coefficients of the leaf where the particles in the leaf range from $i = ileaf1$ to $ileaf2$.

```
Subroutine Compute_Power_Series(leaf.i,b_k)
   Do i =ileaf1, ileaf2
    dx = x_{i_1} − x_{c_1} ; dy = x_{i_2} − x_{c_2} ; dz = x_{i_3} − x_{c_3}
    dx0 = dx; dy0 = dy; dz0 = dz
    tbk = b_{0,0,p} ; peng = tbk
    f(1) = 0; f(2) = 0; f(3) = p · tbk
    Do k_3 = p − 1,0,-1
     tbk = b_{0,p−k_3,k_3} ; ty = tbk
     xf2 = 0; yf2 = (p − k_3) · tbk; zf2 = k_3 · tbk
      Do k_2 = p − 1 − k_3,0,-1
       tbk = b_{p−k_3−k_2,k_2,k_3} ; tx = tbk
       xf1 = (p − k_3 − k_2) · tbk; yf1 = k_2 · tbk; zf1 = k_3 · tbk
        Do k_1 = p − k_3 − k_2,0,-1
         If (k_1 = 0) set dx0 = 1
         tx = dx · tx + b_k
         xf1 = dx0 · xf1 + k_1 · b_k
         yf1 = dx · yf1 + k_2 · b_k
         zf1 = dx · zf1 + k_3 · b_k
         dx0 = dx
        End Do
       If (k_2 = 0) set dy0 = 1
       ty = dy · ty + tx
       xf2 = dy · xf2 + xf1
       yf2 = dy0 · yf2 + yf1
       zf2 = dy · zf2 + zf1
       dy0 = dy
      End Do
     If (k_3 = 0) set dz0 = 1
     peng = dz · peng + ty
     f(1) = dz · f(1) + xf12
     f(2) = dz · f(2) + yf2
     f(3) = dz0 · f(3) + zf2
    End Do
    V(x_i) = V(x_i) + q_i · peng
    F_{i,B_x} = F_{i,B_x} − q_1 · f(1)
    F_{i,B_y} = F_{i,B_y} − q_1 · f(2)
    F_{i,B_z} = F_{i,B_z} − q_1 · f(3)
   End Do
End Subroutine Compute_Power_Series
```

Figure 4.4: The subroutine for evaluating the power series using 3-D Horner's rule *leaf-cluster* algorithm.

4.3.2 Leaf-Leaf Algorithm

For the *leaf-leaf* algorithm, `Compute_Interaction` and `Compute_`$\mathbf{b_k}$ are slightly different from that of the *leaf-cluster* algorithm.

```
Subroutine Compute_Interaction
   Do i = 1, Number of leaves
     b_k = 0
     Do j = i + 1, Number of leaves
      Call Compute_b_k(leaf.i,leaf.j,b_k^i,b_k^j) to compute b_k^i and b_k^j
      coefficients
     End Do
     Compute Interactions between particles in leaf.i directly
     Call Compute_Power_Series(leaf.i,b_k) to compute potential energy
     and forces.
   End Do
   Total Potential = 1/2 Total Potential
End Subroutine Compute_Interaction
```

Figure 4.5: The subroutine for computing the interactions for the *leaf-leaf* algorithm.

The recursive subroutine `Compute_`$\mathbf{b_k}$ takes in the two interacting leaves and their respective $\mathbf{b_k}$ arrays.

```
Recursive Subroutine Compute_b_k(leaf.A,leaf.B,b_k^A,b_k^B)
   If multipole acceptability criterion is satisfied
     Compute b_k^A due to leaf.B and add to the present b_k^A.
     Compute b_k^B due to leaf.A and add to the present b_k^B.
   else
     Compute interaction between leaf.A and leaf.B by direct
     summation
   End if
End Subroutine Compute_b_k
```

Figure 4.6: The subroutine for computing $b_{\mathbf{k}}$ coefficients for the *leaf-leaf* algorithm.

4.4 Implementation and Comparison with Particle-Cluster

To validate the new cluster-cluster algorithms, we applied the algorithms to compute the Coulomb potential energy and the forces on different sized systems for

different orders of approximation and compared the result to that of particle-cluster. We did computations for both the free-space Coulomb potential, and the Coulomb potential in periodic boundary conditions using a system of CH_3Cl molecules.

4.4.1 Coulomb Potential - Free Space

We tested on system sizes

$$N \in \{4096, 21952, 39304, 54872, 74088, 125000, 250000, 500000\}$$

each for orders $p = q \in \{6, 8\}$. The multipole acceptance criterion for the particle-cluster was set to

$$\theta = \frac{r}{R} = 0.7 \tag{4.4.1}$$

which provides an efficient regime for the algorithm. The cluster-cluster algorithms were tested for

$$\theta = \frac{r_A + r_B}{R} = 0.8,\ \ 0.9. \tag{4.4.2}$$

The tree used for the computation in free space was generated by specifying the number of levels and recursively dividing the system until the specified level was attained. This implementation was for both the particle-cluster and the cluster-cluster algorithms.

We show in Figure 4.7 to Figure 4.11 comparisons between the particle-cluster and the cluster-cluster algorithms for relative error (Rel. Err.) in energy, root mean square error (RMSE) in force and cpu time. The exact results for computing these errors are from direct summation computation.

Figure 4.7 and Figure 4.8 compare the particle-cluster and cluster-cluster algorithms for $(\theta, p, q) = (0.8, 6, 6)$ and $(\theta, p, q) = (0.8, 8, 8)$ respectively. The cluster-

cluster algorithms have errors on the order of the errors of the particle-cluster algorithm for all the systems sizes considered. The comparisons of cpu time are shown in the bottom graph of both figures and more clearly in Figure 4.11. They show that the *leaf-cluster* algorithm is about twice as fast as the particle-cluster algorithm.

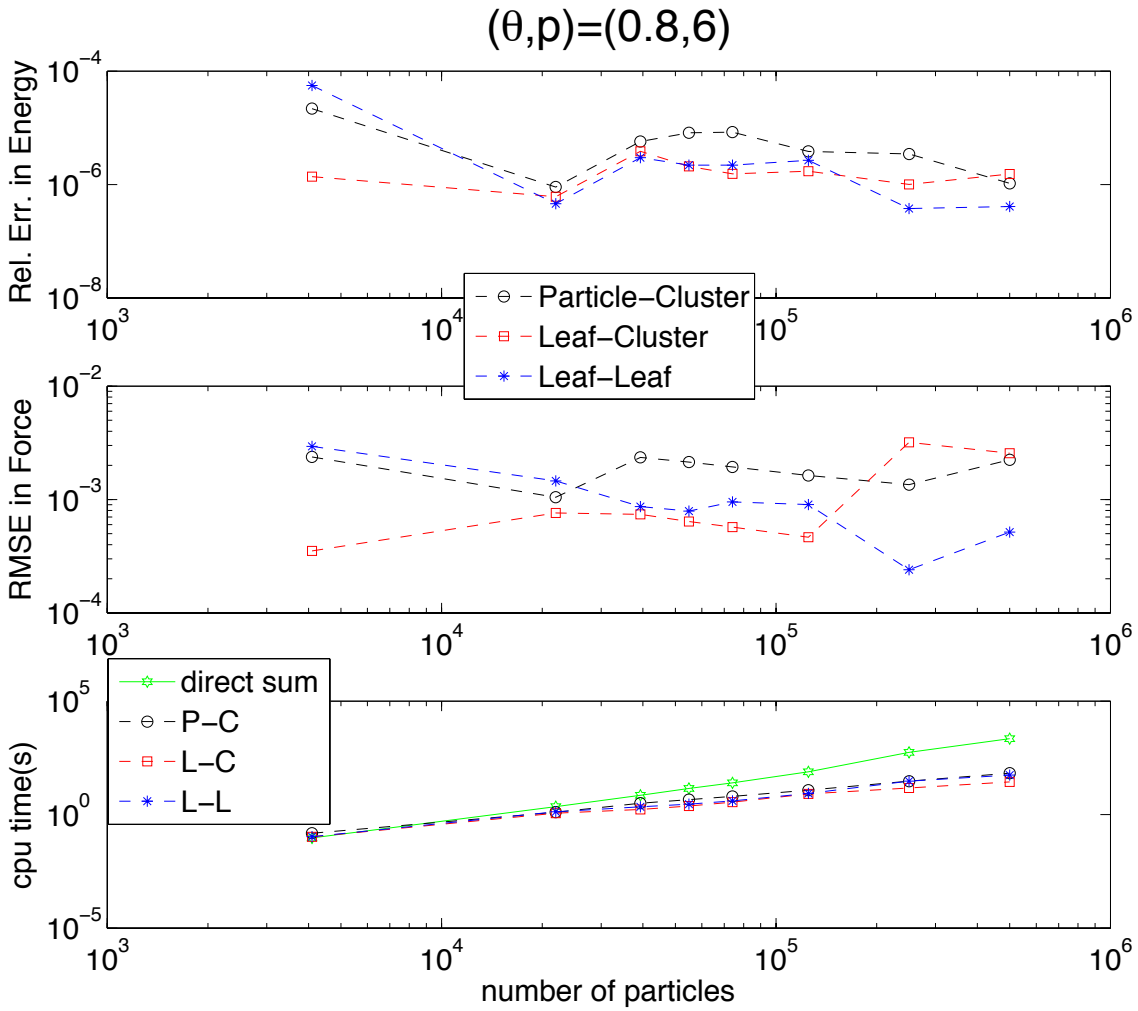


Figure 4.7: Comparison of the particle-cluster algorithm to the *leaf-cluster* and *leaf-leaf* algorithms. The cluster-cluster algorithms are comparable in accuracy to the particle-cluster algorithm and the *leaf-cluster* algorithm is faster than either of the other two.

Figure 4.9 and Figure 4.10 compare the particle-cluster and cluster-cluster algorithms for $(\theta, p, q) = (0.9, 6, 6)$ and $(\theta, p, q) = (0.9, 8, 8)$ respectively. Again, the cluster-cluster algorithms have errors on the order of the errors of the particle-cluster

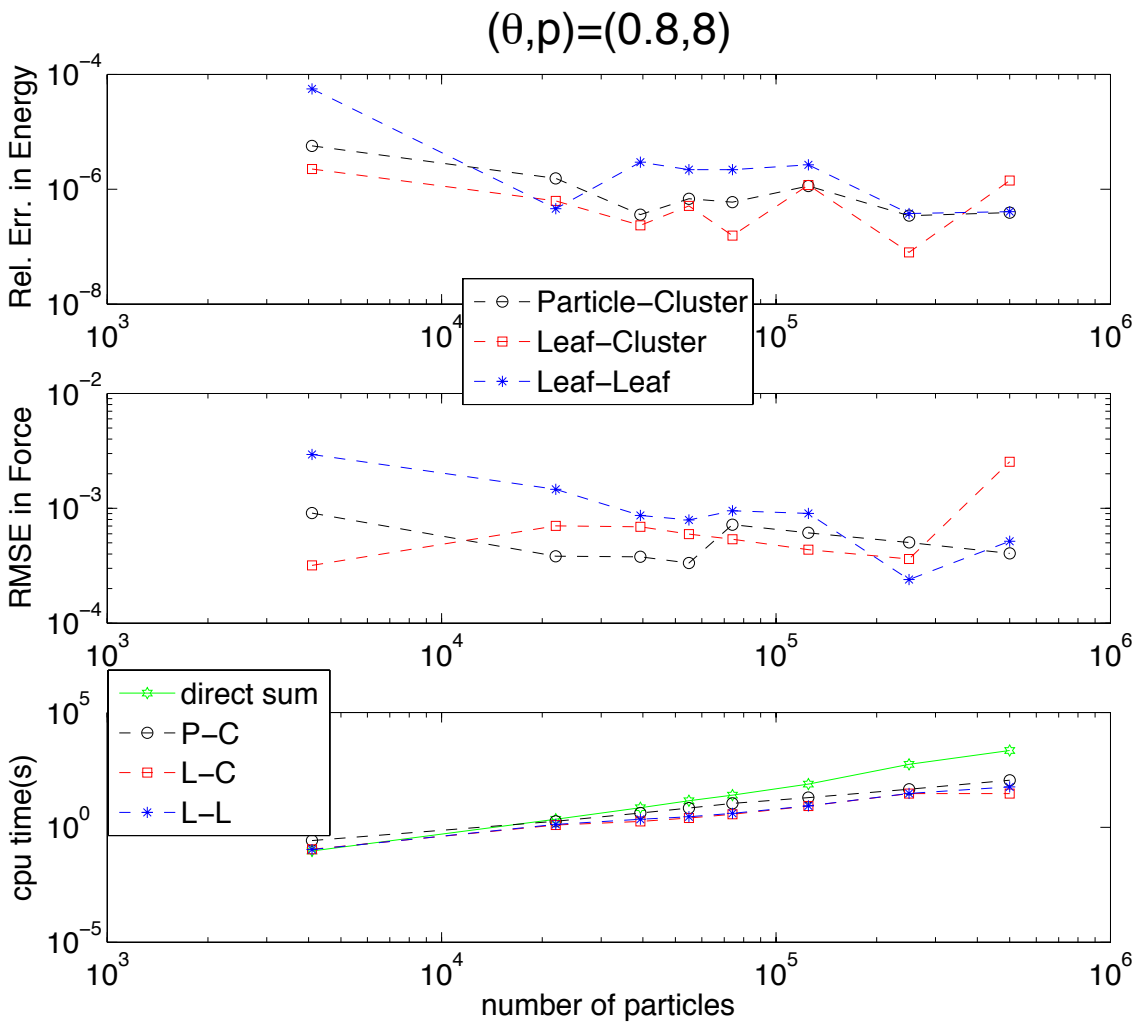


Figure 4.8: Comparison of the particle-cluster algorithm to the *leaf-cluster* and *leaf-leaf* algorithms. The cluster-cluster algorithms are comparable in accuracy to the particle-cluster algorithm and the *leaf-cluster* algorithm is faster than either of the other two.

algorithm for all the systems sizes considered. The comparisons of cpu time are shown in the bottom graph of both figures and more clearly in Figure 4.11. They show that, in this regime, both the *leaf-cluster* and *leaf-leaf* algorithms are faster than the particle-cluster algorithm.

The major result of this chapter is that for the free-space Coulomb potential, the cluster-cluster algorithms, in particular the *leaf-cluster* algorithm, provide *comparable accuracy* as the particle-cluster algorithm but in about *half the time* of the

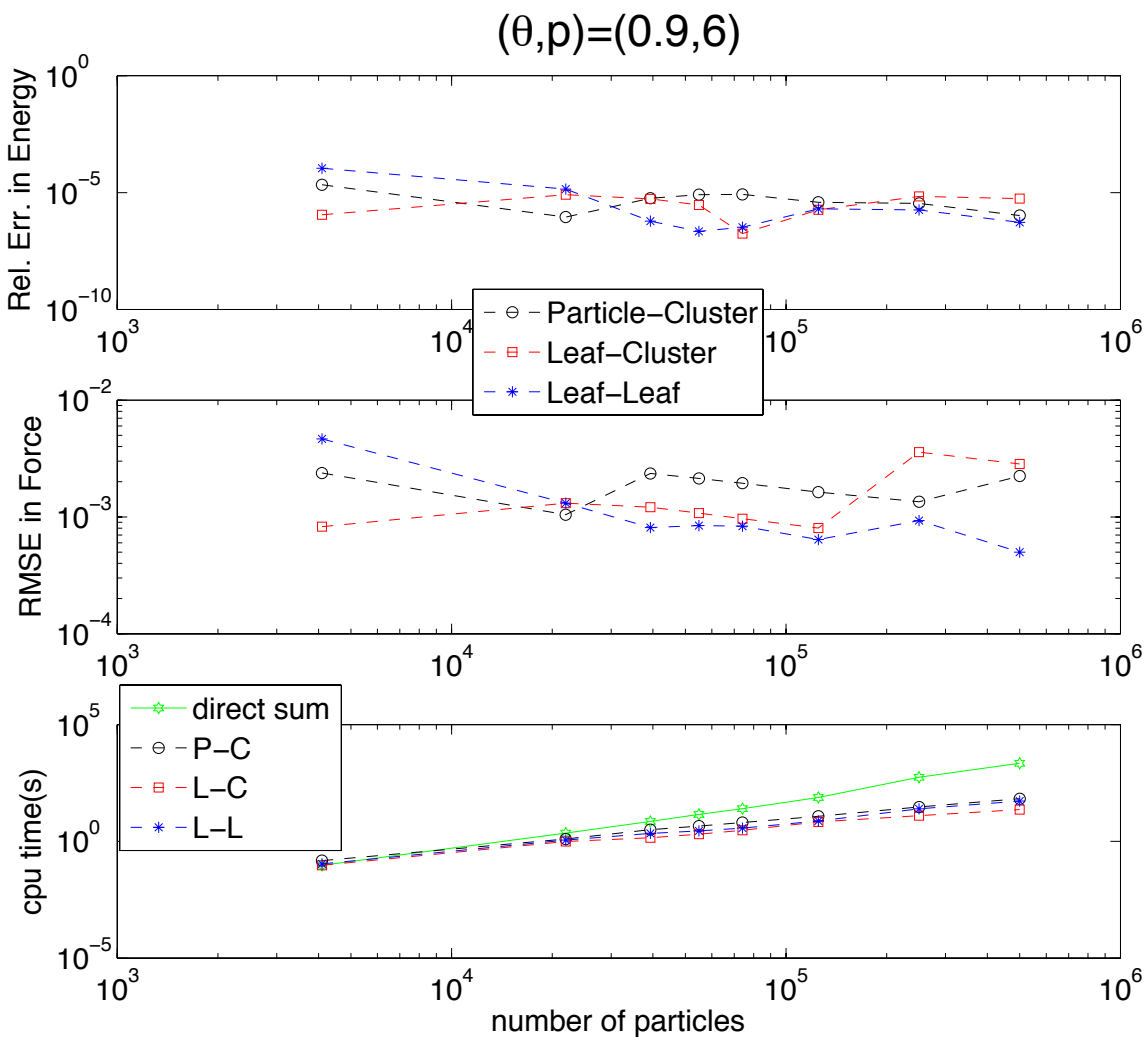


Figure 4.9: Comparison of the particle-cluster algorithm to the *leaf-cluster* and *leaf-leaf* algorithms. The cluster-cluster algorithms are comparable in accuracy to the particle-cluster algorithm and the *leaf-cluster* algorithm is faster than either of the other two.

particle-cluster algorithm.

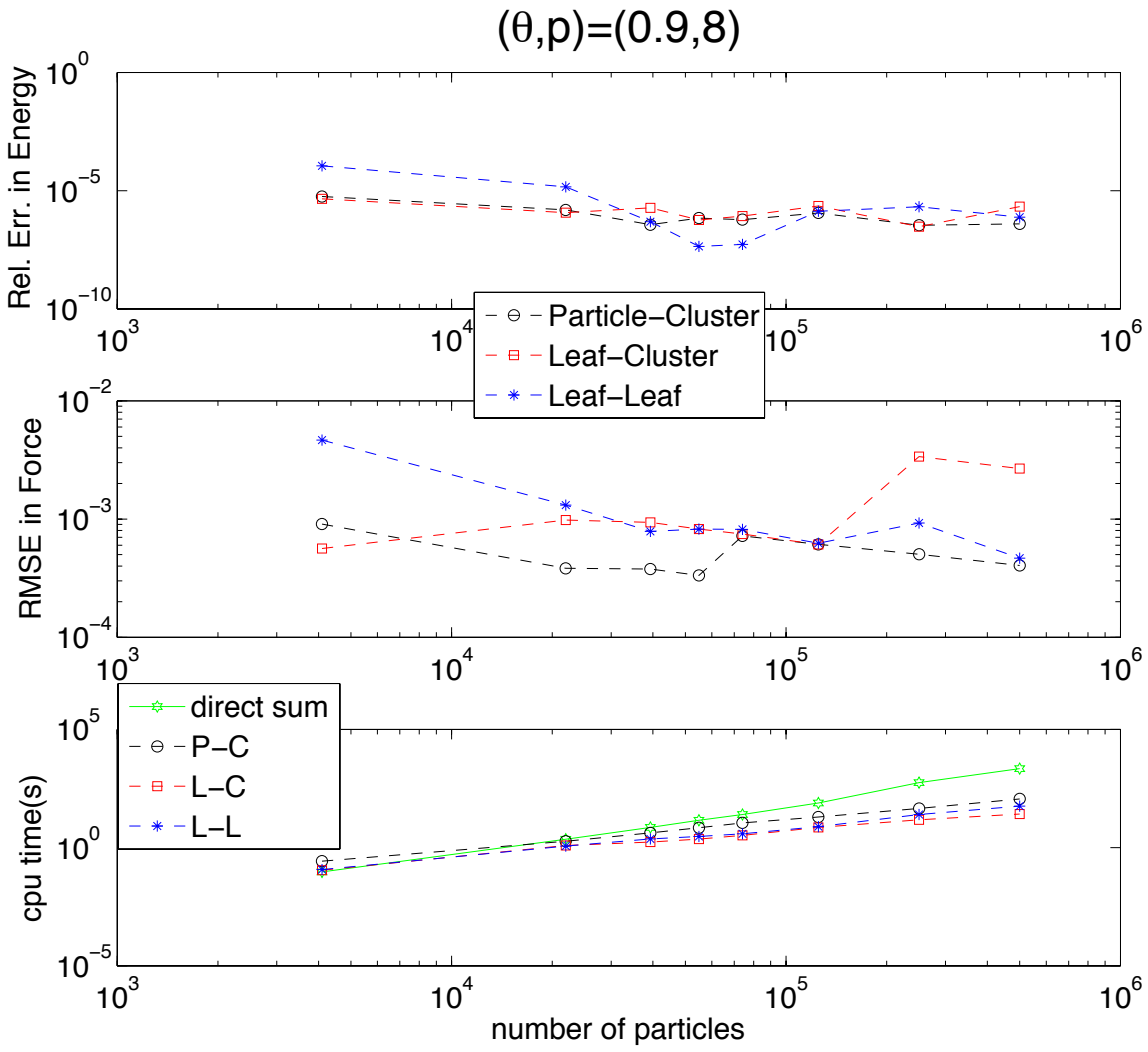


Figure 4.10: Comparison of the particle-cluster algorithm to the *leaf-cluster* and *leaf-leaf* algorithms. The cluster-cluster algorithms are comparable in accuracy to the particle-cluster algorithm and the *leaf-cluster* algorithm is faster than either of the other two.

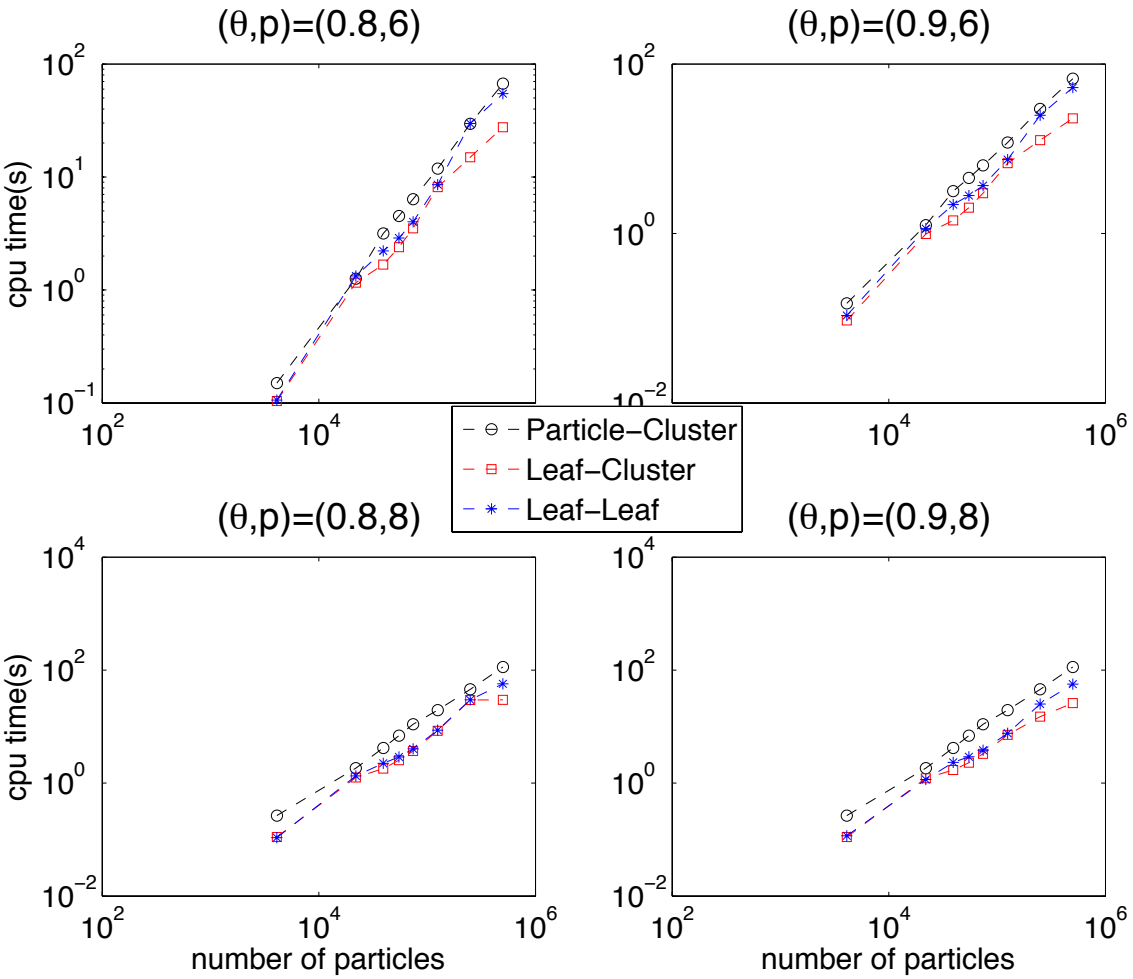


Figure 4.11: These are the same cpu time plots in figures 4.7 to 4.10. They show clearly that the *leaf-cluster* algorithm is faster than the particle-cluster and *leaf-leaf* algorithms for the current implementations.

4.4.2 Coulomb Potential - Periodic Boundary Conditions-Ewald Sum

Here, we present results for the *leaf-cluster* algorithm implemented for the Ewald sum and compare the results to the particle-cluster Cartesian Treecode Ewald method and the Particle Mesh Ewald method. For both methods, we employ a multipole acceptability criterion of

$$\theta = \frac{r_A + r_B}{R} = 0.7, \tag{4.4.3}$$

for the same orders (p, q) of the multipole approximation with $q = p$.

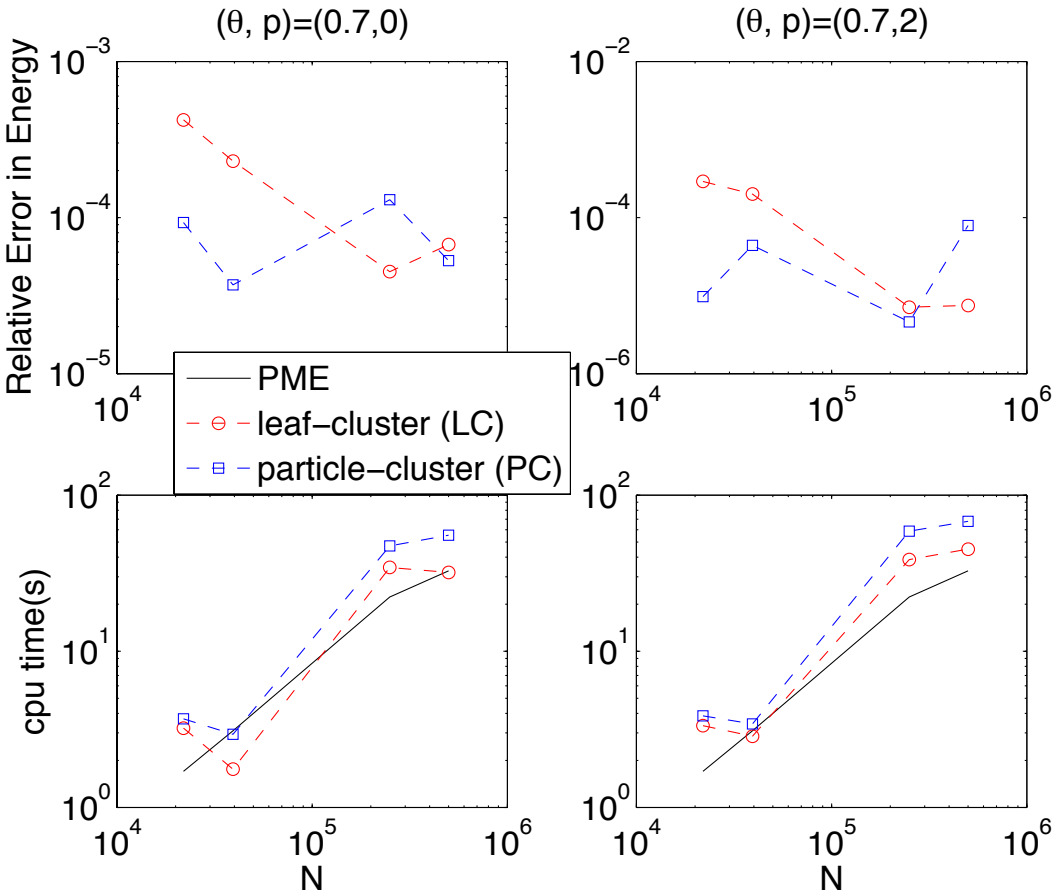


Figure 4.12: The top row compares the relative error in energy of the two algorithms and the bottom row compares the cpu times of the two algorithms with the cpu times for PME. The left column results are for multipole order 0 and the right column for multipole order 2. For both orders the *leaf-cluster* algorithm is comparable to the particle-cluster algorithm in accuracy but faster. The *leaf-cluster* algorithm compares more favorably with PME in cpu time than the particle-cluster algorithm.

Figure 4.12 to Figure 4.13 show that the for the Ewald sum, the cluster-cluster algorithm is slightly faster than the particle-cluster algorithm for comparable accuracy. For low order multipoles $p = 0$ and $p = 2$, the cluster-cluster methods are competitive with the particle mesh Ewald method. Further improvements in the cluster-cluster methods might lead to better cpu comparisons with the PME method for higher orders. Chapter VI, will provide some ideas about improving the cluster-cluster method.

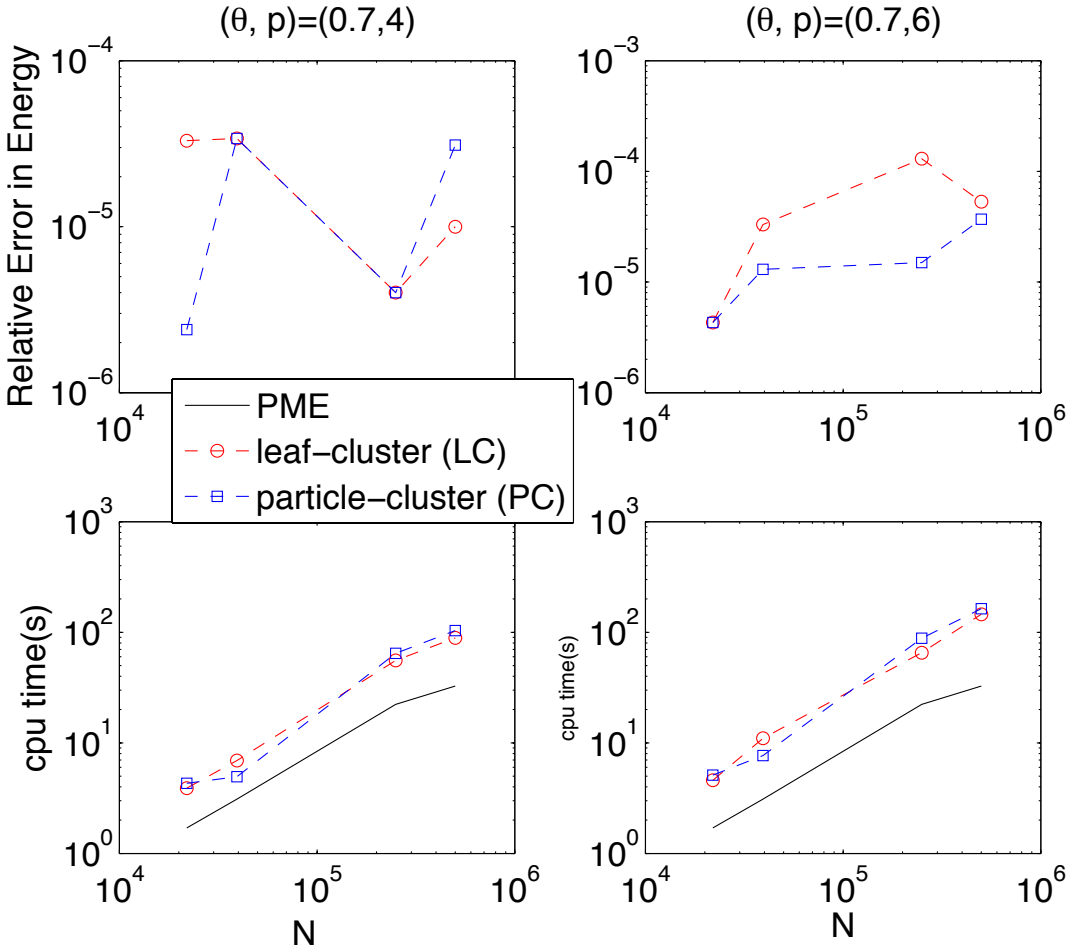


Figure 4.13: The top row compares the relative error in energy of the two algorithms and the bottom row compares the cpu times of the two algorithms with the cpu times for PME. The left column results are for multipole order 4 and the right column for multipole order 6. For both orders the *leaf-cluster* algorithm is comparable to the particle-cluster algorithm in accuracy but faster.

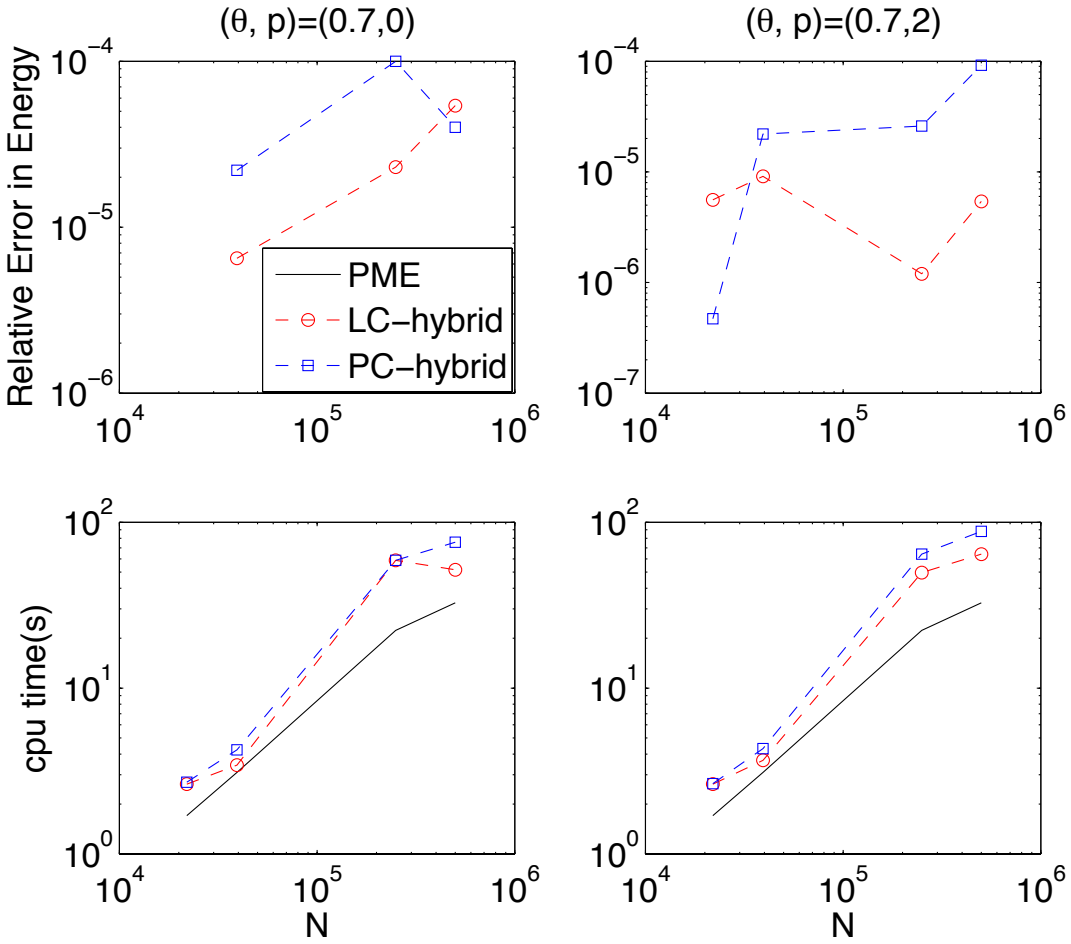


Figure 4.14: Comparison of particle-cluster CTE-PME hybrid and leaf-cluster CTE-PME hybrid. The top row compares the relative error in energy of the two algorithms and the bottom row compares the cpu times of the two algorithms with the cpu times for PME. The left column results are for multipole order 0 and the right column for multipole order 2. For both orders the *leaf-cluster* algorithm is comparable to the particle-cluster algorithm in accuracy but faster. The *leaf-cluster* algorithm compares more favorably with PME in cpu time than the particle-cluster algorithm.

Figure 4.14 and Figure 4.15 show results for the cluster-cluster CTE-PME hybrid algorithm compared to PME and the particle-cluster CTE-PME hybrid method. Both CTE variants are slower than the PME algorithm.

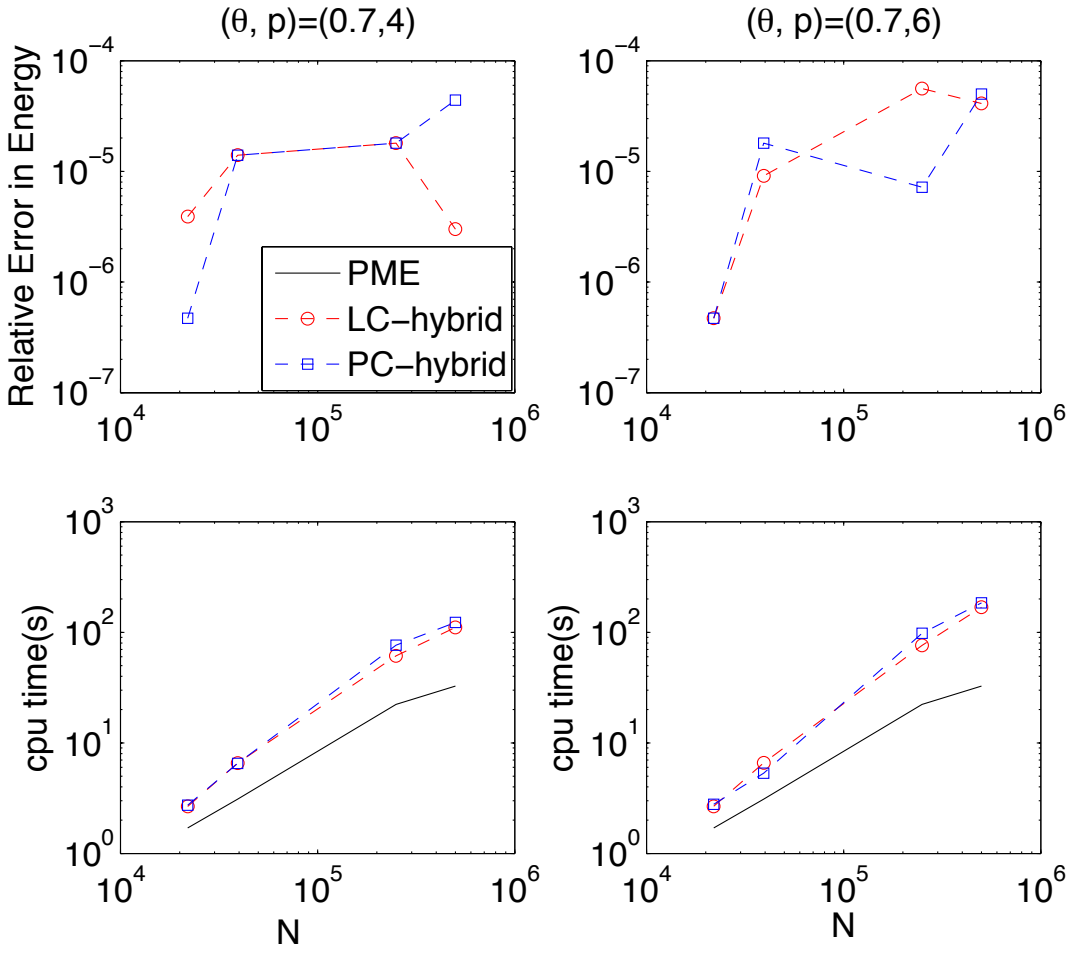


Figure 4.15: Comparison of particle-cluster CTE-PME hybrid and leaf-cluster CTE-PME hybrid. The top row compares the relative error in energy of the two algorithms and the bottom row compares the cpu times of the two algorithms with the cpu times for PME. The left column results are for multipole order 4 and the right column for multipole order 6. For both orders the *leaf-cluster* algorithm is comparable to the particle-cluster algorithm in accuracy but faster.

4.5 Conclusion

We have developed the *leaf-cluster* and *leaf-leaf* algorithms in this chapter. Both algorithms are variants of a cluster-cluster formulation. We have given a proof that the cluster-cluster method has comparable error with particle-cluster for the kernel $\phi(\mathbf{x}) = \frac{1}{|\mathbf{x}|}$ and that the error goes to zero as the orders, (p, q) of the approximation increase. We have also provided computation results which show that the cluster-cluster algorithm can achieve the same medium accuracy as the particle-cluster algorithm in half the computation time of the particle-cluster algorithm.

In the next chapter we will turn our attention to parallel methods for treecode algorithms. In particular, we will focus on parallelizing the particle-cluster Cartesian Treecode Ewald (CTE) Method presented in chapter II and chapter III for DL_POLY_2 and DL_POLY_3. These parallelizations are also suitable for the cluster-cluster algorithms.

CHAPTER V

Parallel Cartesian Multipole Treecode Algorithms

This chapter presents our work on parallelizing the Cartesian treecode method. The parallelization is suitable for both periodic and non-periodic systems, however, we present results for only the periodic system, i.e. Ewald sums.

The literature on parallel algorithms for particle methods based on hierarchical clustering is extensive. The Fast Multipole Method has been parallelized on distributed systems with load balancing achieved via the replicated data strategy [31] and Hilbert space filling curves [34]. Several parallelizations of the Barnes-Hut treecode have used orthogonal recursive bisection [48, 20, 58] to load balance the computation and communication costs.

The load balancing in our parallel treecode algorithms is restricted to the approaches taken in DL_POLY_2 and DL_POLY_3 since the goal was to incorporate the algorithm into these two packages.

All the results presented here are from runs on a Quad-core Intel Xeon cluster with Gigabit Ethernet connection using the Intel Fortran compiler.

5.1 Replicated Data Strategy - DL_POLY_2

The first parallel treecode algorithm we implemented was for DL_POLY_2 which uses the replicated data strategy (RDS) to load balance. The replicated data strategy

refers to the fact that each processor stores a copy of all the atoms in the system.

A variant of our replicated data strategy based algorithm is presented in [39]. The main difference in our implementation for DL_POLY_2 from the implementation in [39] is in load balancing. In [39], load balancing is achieved by grouping the atoms using Morton order to form a 1-D list. The groups are split over the processors in order to achieve similar computational load for each processor based on the computational cost from the previous timestep. Our approach is much simpler since our focus was on liquids which have uniform distribution. For the particle-cluster algorithm, we split the atoms equally over all the processors with the understanding that the computational load will be the same for each atom precisely because of the uniform distribution.

For the *leaf-cluster* algorithm, we split the leaves equally over the processors. This also results in roughly equal work for each processor since the leaves have roughly the same number of atoms.

Each of the N atoms in DL_POLY is labeled with a unique number $iatm$ where $iatm \in \{1, 2, ..., N\}$. For a distributed parallel system of P processors where each processor has a rank $np \in \{0, 1, 2, .., P-1\}$, the algorithm proceeds as follows.

- Build a full tree on each processor. This is possible because each processor stores a copy of all the atoms.

- Assign each processor a subset of the atoms. A processor p is assigned the atoms labeled from $iatm = \frac{np*N}{P}$ to $iatm = \frac{(np+1)*N}{P}$. For the *leaf-cluster* algorithm, if the total number of leaves formed from the tree is LT, then processor p is assigned the leaves labeled from $ileaf = \frac{np*LT}{P}$ to $ileaf = \frac{(np+1)*LT}{P}$.

- For each processor, traverse the tree for each atom or leaf in the subset assigned

to the processor in order to compute the potential energy and the forces on the atoms. The computational work is load balanced because each processor has the same number of atoms.

- After the interactions, the algorithm performs a global sum over all processors to compute the total potential.

The replicated data strategy is simple to code, easily adaptable from serial code and easy to reduce to a serial code [59]. The parallelization is independent of distribution of atoms and thus the efficiency does not depend on whether the system is isotropic or not.

Despite these advantages, the replicated data strategy has quite significant drawbacks [59, 54]. The major limitations are

- An inability to handle large systems because of large memory requirements for storing a copy of all atoms.

- The global summation also reduces efficiency for large number of processors on the order of $P > 10^2$.

The inability of the replicated data strategy to handle large systems necessitated the development of the domain decomposition [44] approach to load balancing which will be explained shortly.

We now present the scaling results for the parallel Cartesian treecode method load balanced via the replicated data strategy.

5.1.1 Results

We parallelized the Cartesian treecode Ewald method introduced earlier in Chapter II and run tests of a few systems with $\theta = 0.7$ and the cutoff $r_c = \frac{L}{2}$. Figure 5.1

and Figure 5.2 show the scaling for the particle-cluster and *leaf-cluster* variants respectively for $N = 21952$ atoms on up to 64 processors. The figures show that both parallel algorithms have almost perfect scaling up to 64 processors.

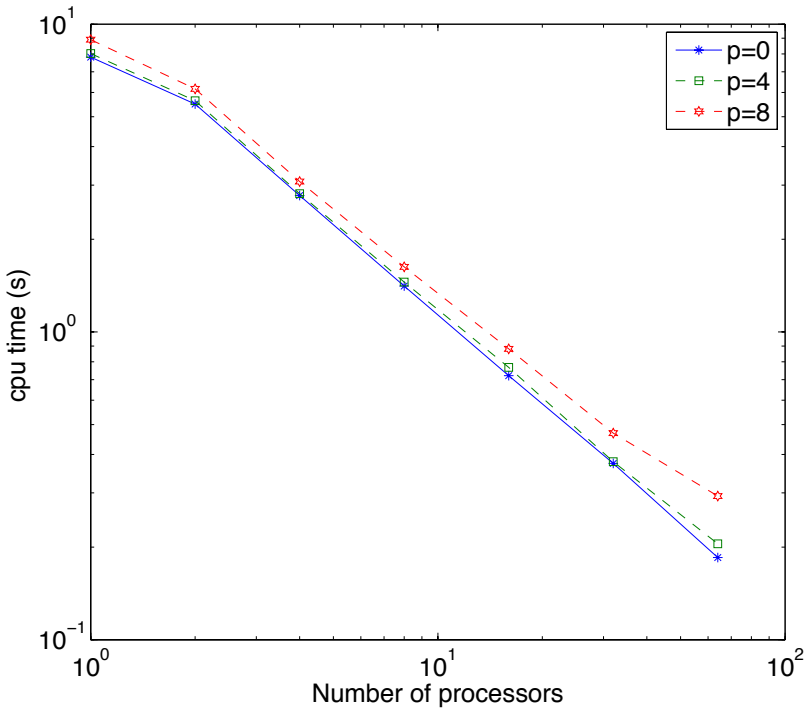


Figure 5.1: Scaling of parallel particle-cluster CTE using the RDS method for $N = 21952$ atoms, $\theta = 0.7$ and $r_c = \frac{L}{2}$. The figure shows almost perfect scaling for up to 64 processors.

We expect that the scaling will deteriorate for increasing number of processors for a constant system size because of an increase in communication.

5.2 Domain Decomposition - DL_POLY_3

Domain decomposition (DD) [44, 59, 54, 27] is the basis for load balancing in DL_POLY_3. It is a strategy for parallelizing the linked cell method of Hockney and Eastwood [29]. Domain decompostion is appropriate for systems with potentials which are short range. For DD the simulation box is divided into P equal subrvolumes for each of the P processors. Each processor works on its assigned subregion and

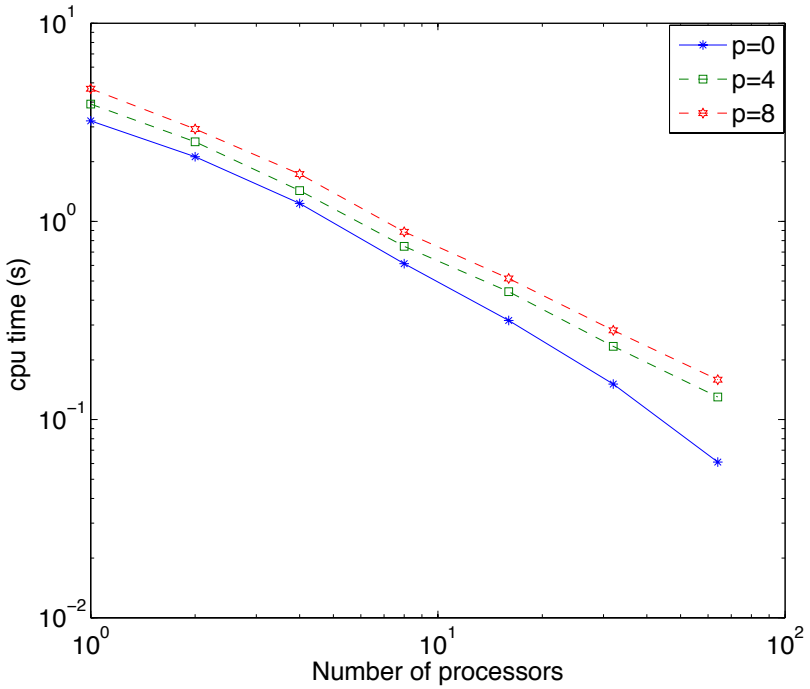


Figure 5.2: Scaling of parallel *leaf-cluster* CTE using the RDS method for $N = 21952$ atoms, $\theta = 0.7$ and $r_c = \frac{L}{2}$. The figure shows almost perfect scaling for up to 64 processors.

there is minimal communcation between processors. The major advantages of domain decomposition are

- Low memory requirements since each processor stores only a subset of the atoms.
- Ability to handle much larger systems on the order of 10^7 because of the low memory requirement.
- Less communication overhead because of the limited all-to-all communication leading to better efficiency.

Domain decomposition however is not very efficient for systems with long-range potentials and it is quite complex to code. In addition, DD is not very portable since different parallel architectures requre different algorithms for subdividing and assigning subregions to processors. Finally, non-isotropic systems might lead to poor load balancing for DD. This is because division of the simulation box might result

in significant disparities in the number of atoms in each subregion as a result of the non-isotropy.

5.2.1 Domain Decomposition with a Linked Cell Method

The linked cell method enables the rapid and efficient computation of potentials with a short interaction range compared to the simulation box length.

The simulation box is divided into subcells of length equal to or slightly greater than the cutoff radius. Then interactions of the atoms in a subcell are restricted to the subcell and the immediate neighbors of the subcell since these are the only atoms which will be within the cutoff radius of the atoms. A linked list points to each subcell and its immediate neighbors while providing access to the particles in each subcell. If periodic boundaries are employed, the neighbors of subcells at the edge of the simulation box include the periodic extensions.

DL_POLY_3 employs the domain decomposition method due to Rapaport [44] in order to subdivide the simulation box into the number of processors. The P processors are assumed to correspond to a space that is subdivided into $P = P_x \times P_y \times P_z$ subregions. Processor np $(0 \leq np \leq P-1)$ corresponds to the region $\{np_x, np_y, np_z\}$ with

$$np_x = \lfloor np/(P_y P_z) \rfloor, \tag{5.2.1}$$

$$np_y = \lfloor np/P_z \rfloor \bmod P_y, \tag{5.2.2}$$

$$np_z = np \bmod P_z. \tag{5.2.3}$$

As an example for $P = 16$ and $(P_x, P_y, P_z) = (4, 2, 2)$ the Table below shows the subregion for each processor np.

$np = 0$	$\{0, 0, 0\}$	$np = 4$	$\{1, 0, 0\}$	$np = 8$	$\{2, 0, 0\}$	$np = 12$	$\{3, 0, 0\}$
$np = 1$	$\{0, 0, 1\}$	$np = 5$	$\{1, 0, 1\}$	$np = 9$	$\{2, 0, 1\}$	$np = 13$	$\{3, 0, 1\}$
$np = 2$	$\{0, 1, 0\}$	$np = 6$	$\{1, 1, 0\}$	$np = 10$	$\{2, 1, 0\}$	$np = 14$	$\{3, 1, 0\}$
$np = 3$	$\{0, 1, 1\}$	$np = 7$	$\{1, 1, 1\}$	$np = 11$	$\{2, 1, 1\}$	$np = 15$	$\{3, 1, 1\}$

Table 5.1: Triplets indicating the subregion for processor np.

5.2.2 Parallel Cartesian Treecode Algorithm with Domain Decomposition

As we pointed out earlier, domain decomposition is suitable for systems with short range potential. The Coulomb potential on the other hand is long range and as such to compute it accurately requires interactions between all atoms. An explicit atom-atom interaction with DD will then require that all processors send all the atoms native to the processor to all other processors. This of course will severely degrade the efficiency of the algorithm because of the communication cost involved. Our algorithm attempts to decrease the communication cost by aggregating the information from each processor and sending this information in bulk.

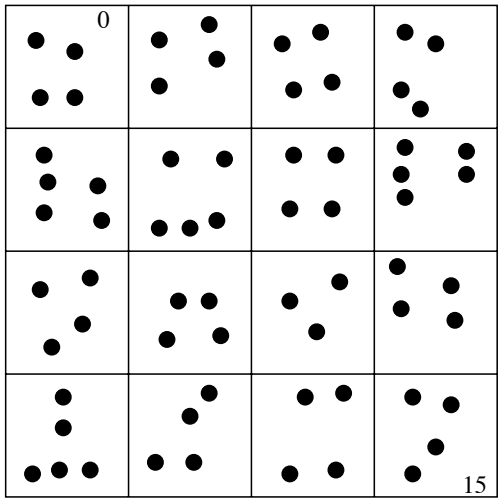
Our algorithm is a first attempt at parallelizing the Cartesian treecode method with the domain decomposition method. It is by no means the final product. In fact, we will point out in chapter VI a few ideas to improve the present algorithm. It is worth noting that the particle-mesh Ewald method has already been parallelized for DL_POLY_3 with the domain decomposition method [9, 55, 56].

The parallelization of the Cartesian treecode method in DL_POLY_3 is inspired by approaches taken in [48, 20, 58]. The major hurdle for the algorithm is providing each processor with information about all the atoms in the system in order for the processor to accurately compute the interactions on its native atoms. Naively sending every atom to every processor eliminates the limited communication advantage of the DD method and results in large memory requirements as well.

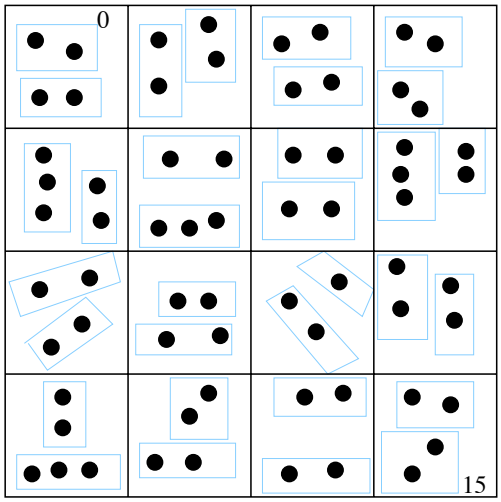
This first implementation of our algorithm uses all-to-all communication between

processors. We aggregate the coordinates and charge of native atoms in the moments and center of leaves of subtrees and send this information instead to reduce the communicaton overhead.

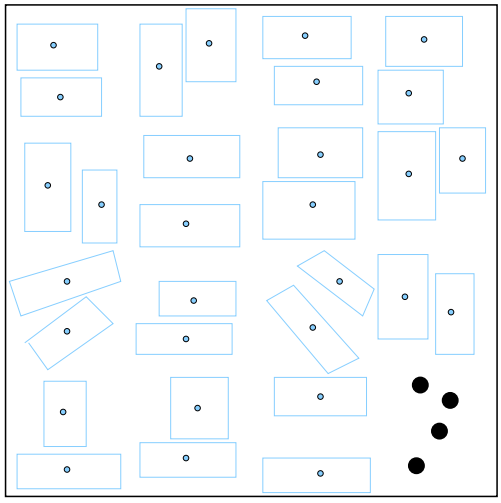
The parallel algorithm proceeds as depicted by Figure 5.3 where $P = 16$.



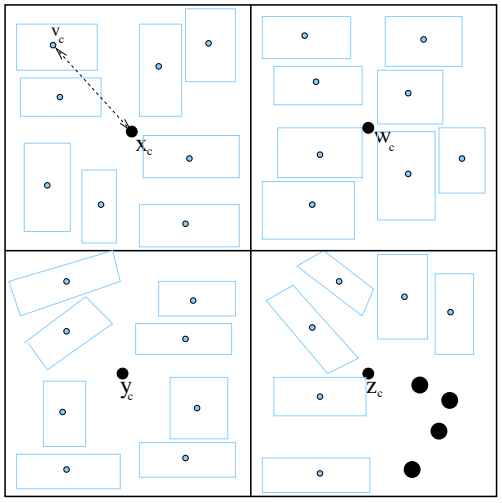
(a) DL_POLY_3 updates the atoms on each processor so that each processor has about equal number of native atoms.



(b) Build subtrees. Store geometric center (gc) and buffered moments of leaves (blue boxes).



(c) Each processor sends gc and buffered moments to every other processor. Processor, $p = 15$ after its received all moments and gc.



(d) Each processor builds a full tree. Leaves from other processors act as atoms. Traverse full tree to compute potential and forces on native atoms.

Figure 5.3: Figures (a), (b), (c) and (d) describe the parallel algorithm with DD.

The algorithm as shown in Figure 5.3 proceeds by

- First building a subtree on each processor with the atoms assigned to the processor as shown in Figure 5.3 (b)

- The geometric center (gc) and the buffered moments $\tilde{M}^{\mathbf{k}}$ of the leaves of each subtree are then computed. The buffered moments are different from the moments introduced earlier in Chapter II and will be defined shortly.

- An all-to-all communication is performed to send the leaves of the subtree of each processor to every other processor. Figure 5.3 (c) shows processor $np = 15$ with the native atoms and the leaves from the other processors after the all-to-all communication.

- Build a full tree on each processor consisting of the native atoms and the leaves from the other processors. The leaves from the other processors serve as pseudo-particles with their positions given by the geometric center. Figure 5.3 (d) shows the full tree in processor $np = 15$ at only the first level.

- Loop over atoms native to the processor and traverse the tree to compute the potential and forces.

As shown in Figure 5.3 (d), building the full tree requires computing the moments of clusters which contain leaves from the other processors. The procedure to handle this is straightforward and we explain it presently. As an example, consider cluster **X** shown in Figure 5.4 which contains several clusters including cluster **V**.

Then the **k**th moment of cluster **X**, $M_X^{\mathbf{k}}$ is a sum of the contribution of cluster **V** and the contributions of the other particles in cluster **X** which is given as

$$M_X^{\mathbf{k}} = \sum_{i \in \mathbf{X}-\mathbf{V}} q_i (x_i - x_c)^{\mathbf{k}} + \sum_{i \in \mathbf{V}} q_i (x_i - x_c)^{\mathbf{k}} \tag{5.2.4}$$

$$= M_{\mathbf{X}-\mathbf{V}}^{\mathbf{k}} + M_{\mathbf{V}}^{\mathbf{k}} \tag{5.2.5}$$

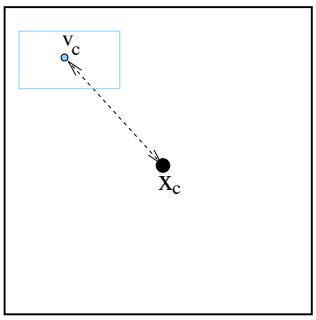


Figure 5.4: Cluster $\mathbf{V}$ inside $\mathbf{X}$.

Then

$$M_{\mathbf{V}}^{\mathbf{k}} = \sum_{i \in \mathbf{V}} q_i [(x_i - v_c) + (v_c - x_c)]^{\mathbf{k}} \tag{5.2.6}$$

$$= \sum_{i \in \mathbf{V}} q_i \sum_{||\mathbf{j}||=0}^{||\mathbf{k}||} \binom{\mathbf{k}}{\mathbf{j}} (x_i - v_c)^{\mathbf{k}-\mathbf{j}} (v_c - x_c)^{\mathbf{j}} \tag{5.2.7}$$

$$= \sum_{||\mathbf{j}||=0}^{||\mathbf{k}||} (v_c - x_c)^{\mathbf{j}} \binom{\mathbf{k}}{\mathbf{j}} \sum_{i \in \mathbf{V}} q_i (x_i - v_c)^{\mathbf{k}-\mathbf{j}} \tag{5.2.8}$$

$$= \sum_{||\mathbf{j}||=0}^{||\mathbf{k}||} (v_c - x_c)^{\mathbf{j}} \tilde{M}_{\mathbf{V}}^{\mathbf{k}} \tag{5.2.9}$$

where the buffered moment of cluster $\mathbf{V}$, $\tilde{M}_{\mathbf{V}}^{\mathbf{k}}$, is defined as

$$\tilde{M}_{\mathbf{V}}^{\mathbf{k}} = \binom{\mathbf{k}}{\mathbf{j}} \sum_{i \in \mathbf{V}} q_i (x_i - v_c)^{\mathbf{k}-\mathbf{j}}. \tag{5.2.10}$$

The buffered moments for the leaves in the subtrees of each processor have to be computed in their native processors because the particles in cluster $\mathbf{V}$ are only present in the processor to which they are assigned. The algorithm gets access to $\mathbf{x}_c$ only after building the full tree.

The moment $M_{\mathbf{V}}^{\mathbf{k}}$ is multidimensional and given as

$$M_{\mathbf{V}}^{\mathbf{k}} = \sum_{||\mathbf{j}||=0}^{||\mathbf{k}||} (v_{c1} - x_{c1})^{j_1} (v_{c2} - x_{c2})^{j_2} (v_{c3} - x_{c3})^{j_3} \binom{k_1}{j_1} \binom{k_2}{j_2} \binom{k_3}{j_3} \tag{5.2.11}$$
$$\times \sum_{i \in \mathbf{V}} q_i (x_{i1} - v_{c1})^{k_1 - j_1} (x_{i2} - v_{c2})^{k_2 - j_2} (x_{i3} - v_{c3})^{k_3 - j_3}.$$

We store $M_{\mathbf{V}}^{\mathbf{k}}$ in a one dimensional array. For a pth order approximation, the length of $M_{\mathbf{V}}^{\mathbf{k}}$ is given as

$$length(M_{\mathbf{V}}^{\mathbf{k}}) = \frac{1}{6!}\prod_{i=1}^{6}(i+p). \tag{5.2.12}$$

This means that our algorithm is guaranteed to reduce the communication overhead of sending all atoms to all processors only if for a cluster size of N_0,

$$length(M_{\mathbf{V}}^{\mathbf{k}}) < 4N_0. \tag{5.2.13}$$

The term $4N_0$ is the representative cost of sending the 3 coordinates and charge of each atom of a cluster. For large systems with large N_0, the condition in (5.2.13) is easily attainable.

5.2.3 Results

We present here preliminary results for the domain decomposition based particle cluster Cartesian treecode Ewald method for system sizes $N \in \{35000, 125000, 500000\}$.

	Relative Error in Energy					Cpu time (seconds)				
	np					np				
N	1	2	4	8	16	1	2	4	8	16
35000	3e-3	4e-2	5e-2	6e-2		11.0	6.10	4.97	5.25	
125000	6e-4	2e-3	6e-3	1e-2		105	47.1	26.3	19.8	
500000	9e-5			8e-3	1e-2	1611			206	98.1

Table 5.2: Scaling and accuracy results for parallel CTE with $(p, \theta) = (6, 0.7)$. The number of tree levels was fixed at 4 for all system sizes. We see that the algorithm scales well but the accuracy decays with increasing number of processors.

Table 5.2 shows the good scaling of the algorithm up to 16 processors but a deterioration in the accuracy of the method with an increase in the number of processors. This decay in accuracy is because the current method employs more approximations with more processors. A particular processor only receives leaves with moments from

all other processors. As a result, atoms which will ordinarily be in the near field of a target atom will be treated by multipole approximation if these atoms happen to be in another processor. This phenomenon increases when the number of processors increase since a lot more of the neighboring atoms of a target atom will be in other processors. Of course an increase in the number of atoms results in less deterioration in the accuracy with increasing number of processors because a target atom will have more of its neighboring atoms in the same processor as the target atom. This suggests that for very large systems, the deterioration in accuracy might not be significant and the algorithm might prove useful in its current form.

The next iteration of this algorithm will focus on refining the communication step such that atoms in neighboring processors which fall within the near-field of each other will be treated exactly. A similar idea has been implemented in [20, 48].

We next looked at the communication time involved in the algorithm to study whether we can achieve comparable communication time for different system sizes. Figure 5.5 shows results for up to 8 processors. For this limited sample, the communication times for $N = 35000$ and $N = 125000$ are roughly comparable. This is because we picked the same number of levels for both systems and this produces the same number of leaves in each subtree and hence the same communication overhead.

5.3 Conclusion

We have presented in this section two parallel methods for the Cartesian treecode multipole algorithm. The first method used a replicated data strategy to load balance the algorithm since that is the strategy of choice in DL_POLY_2. We implemented the algorithm for the particle cluster and *leaf-cluster* variants of the Cartesian treecode Ewald method and incorporated both into DL_POLY_2. We have shown that both

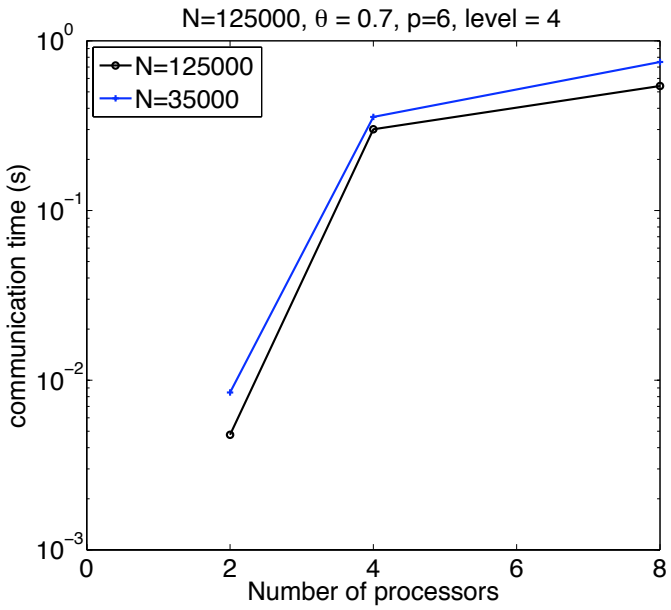


Figure 5.5: Comparable communication overhead for $N = 35000$ and $N = 125000$ for maximum tree level 4 and $(p, \theta) = (6, 0.7)$. The communication cost seems to be $O(np)$.

variants of the parallel CTE algorithm scale well up to 64 processors. The algorithm is limited though because of the large memory requirements.

The second method employed the domain decomposition method used in DL_POLY_3. This algorithm is in the preliminary stages. The initial results suggest promising speedup and a good control of the communication cost. However, there is more work to be done to solve the problem of deterioration in accuracy with increasing number of processors.

The next section presents a summary of the thesis and possible directions for further research.

CHAPTER VI

Conclusion

6.1 Summary

The first project of this thesis validated the Cartesian Treecode Ewald method (CTE) as a viable method for performing molecular dynamics simulations. We performed MD simulations of liquid methyl chloride and reproduced the radial distribution, velocity auto-correlation and force auto-correlation functions. These were compared to results from the Particle Mesh Ewald method (PME) and found to agree for varying sets of parameters. We also devised a method for handling periodic boundary conditions for the Cartesian Treecode Ewald method.

We have also done a systematic comparison of the CTE method, PME method and a CTE-PME hybrid method for computational time and for accuracy in energy, forces and the virial. These comparisons were performed for system sizes ranging from 20000 to 500000. We have ascertained that the cpu time for the current implementation of the CTE method is within an order of magnitude of the PME method in cpu time for comparable accuracy. We have also observed that the CTE-PME hybrid method is faster than the CTE method for small systems but slower for large systems. The present implementations of both methods are however slower than the PME method.

The second project developed two versions of a cluster-cluster algorithm. These are a *leaf-cluster* algorithm and a *leaf-leaf* algorithm. We have shown that these algorithms can achieve the accuracy of the particle-cluster algorithm. The *leaf-cluster* algorithm is found to be about twice as fast as the particle-cluster algorithm with comparable accuracy when both are used to compute the Coulomb potential in free space. For the Ewald sum, the *leaf-cluster* algorithm is about 1.5 times faster than the particle-cluster algorithm.

The third project parallelized the CTE method for DL_POLY_2 and DL_POLY_3. Because DL_POLY_2 uses a replicated data strategy (RDS) method for load-balancing, we designed our algorithm to make use of the RDS method. The parallelization is similar to the parallelization in [39] with the main difference in the approach to splitting the particles over the processors. We parallelized both the particle-cluster and cluster-cluster CTE methods and found both to scale well up to 64 processors from our test on a system of 21952 atoms.

The parallel CTE algorithm for DL_POLY_3 was designed to make use of the domain decomposition method which is the method used for load-balancing in DL_POLY_3. We tested the scaling of this algorithm and the accuracy. Although the algorithm scales well for up to the 16 processors that we tested, the accuracy decays with increasing number of processors. This is because the proportion of multipole approximations versus direct summation increases with increasing number of processors. This algorithm is in the preliminary design stage and we provide ideas about improving the algorithm in the next section.

6.2 Future Directions

Several open problems remain from the work done in this thesis. One such problem is a comparison of the Cartesian Treecode Ewald method (CTE) to the Particle Mesh Ewald method (PME) for non-uniform particle distributions. MD simulations are heavily used in biophysics to study large biomolecules. These biomolecules usually have non-uniform distribution and yet their electrostatic interactions are handled by the PME method which is known to be less efficient for non-uniform distributions [27]. The CTE method might prove to be better suited for MD simulation of biomolecules than the PME method.

There is also the possibility of using the Cartesian treecode method to compute interactions of Gaussian charge distributions in electron structure calculations as in the Continuous Fast Multipole Method [60]. This possibility arises because Equation 2.2.46 gives the potential, $\frac{\mathrm{erf}(\alpha r)}{r}$, due to a Gaussian charge distribution for which a suitable recurrence relation can be derived similar to the recurrence relation for the Gaussian-screened Coulomb potential, $\frac{\mathrm{erfc}(\alpha r)}{r}$, used in the Ewald method in Equation 2.4.67.

Another possible direction is a combination of the particle-cluster and cluster-cluster algorithms. The effect of far-field particles on a target particle decreases with distance for pair potentials like the Coulomb potential. As a result, the effect of far-field clusters on a target particle can be computed differently depending on the distance of the cluster from the target particle. For example, a variable multipole acceptance criterion (MAC) can be implemented. Then the clusters that are farthest from the target particle and satisfy a MAC say with

$$\frac{r_A + r_B}{R} \leq 0.5, \tag{6.2.1}$$

are handled by cluster-cluster approximation and clusters that are closer and satisfy a MAC with

$$0.5 < \frac{r_B}{R} < 0.8, \tag{6.2.2}$$

are handled by the particle-cluster algorithm. This combined algorithm has the potential to be faster while achieving high accuracy.

Another open question from this thesis is a possible refinement of the *leaf-leaf* algorithm in order to speed up the algorithm. Let us suppose Figure 6.1 shows all the leaves from the hierarchical clustering of a two dimensional system. Suppose leaf

$\mathbf{1}^{F}$	$\mathbf{2}^{F}$	$\mathbf{1}^{G}$	$\mathbf{2}^{G}$	$\mathbf{1}^{H}$	$\mathbf{2}^{H}$
$\mathbf{4}^{F}$	$\mathbf{3}^{F}$	$\mathbf{4}^{G}$	$\mathbf{3}^{G}$	$\mathbf{4}^{H}$	$\mathbf{3}^{H}$
$\mathbf{1}^{E}$	$\mathbf{2}^{E}$	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{1}^{A}$	$\mathbf{2}^{A}$
$\mathbf{4}^{E}$	$\mathbf{3}^{E}$	$\mathbf{4}$	$\mathbf{3}$	$\mathbf{4}^{A}$	$\mathbf{3}^{A}$
$\mathbf{1}^{D}$	$\mathbf{2}^{D}$	$\mathbf{1}^{C}$	$\mathbf{2}^{C}$	$\mathbf{1}^{B}$	$\mathbf{2}^{B}$
$\mathbf{4}^{D}$	$\mathbf{3}^{D}$	$\mathbf{4}^{C}$	$\mathbf{3}^{C}$	$\mathbf{4}^{B}$	$\mathbf{3}^{B}$

Figure 6.1: The figure shows the leaves from a hierachical clustering of a two dimensional system.

$\mathbf{1}^F$ interacts with leaf $\mathbf{1}^G$ via multipole approximation then leaf pairs like $(\mathbf{2}^F, \mathbf{2}^G)$, $(\mathbf{1}^G, \mathbf{1}^H)$, $(\mathbf{4}^E, \mathbf{4})$ and several others also interact via multipole approximation. The present algorithm computes the Taylor coefficients $a_{\mathbf{k}}$, where

$$a_{\mathbf{k}} = \frac{1}{\mathbf{k}!} D^{\mathbf{k}}_{\mathbf{y}} \phi(\mathbf{x}_A - \mathbf{y}_B) \tag{6.2.3}$$

which depends on the vector between the centers of each pair, $(\mathbf{x}_A - \mathbf{y}_B)$, for each pair. However, all the pairs given above and several more will have the same vector

between them and as such the Taylor coefficients need only be computed once and used for all of these pairs. Similar procedures for several other pairs for example $(\mathbf{1}^E, \mathbf{4}^D)$ can also be implemented. This will undoubtedly lead to a sharp decrease in the computation time of the *leaf-leaf* algorithm.

In [39] the spatial locality of data is preserved by Morton ordering. One open problem is to speed up the treecode algorithm by utilizing the spatial locality of the data. A loop over particles in the order of particle numbers might be inefficient. This is because consecutive particles in the loop might actually be spatially far apart in memory. Accessing data that are far apart in memory leads to large memory access times and thus slower computation time. Data access from memory is done in blocks. Thus when the data of a particle is accessed, the data for particles that are spatially close are also retrieved and placed in cache. A loop over particles that are spatially close will then lead to fast memory access since the data will be readily available in cache. Morton order is one such procedure for localizing data spatially. Another procedure is the Hilbert order which preserves spatial locality better than the Morton order [28].

For the parallel CTE method which uses domain decomposition, there is still a lot of improvement possible for the algorithm.

- One such improvement is reducing the communication cost by avoiding the all-to-all communication. This can be achieved if each processor has knowledge of coordinates of the sub-volume in every other processor. Then the communication between processors that are more than the cutoff distance apart will be eliminated since interactions between their native particles are not significant.

- For processors with sub-volumes which lie within the cutoff, the MAC criterion can be applied between processors since each processor has knowledge of the

sub-volume of other processors. If the MAC is satisfied between two processors, then these processors will exchange the leaves of their respective subtrees. If the MAC is not satisfied, then the two processors exchange their particles. This will improve the accuracy of the algorithm by reducing the number of multipole approximations.

Variants of the ideas for improving the parallel algorithm are detailed in [48, 20, 58, 27]. The algorithm is aimed at large systems of size $N > 10^6$ and for large number of processors $2^7 \leq P \leq 2^{10}$. Testing the algorithms on systems of these sizes will be a worthy project as well.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M Abramowitz and I. A. (Eds.) Stegun. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, chapter 8 and 22, pages 331–339 and 771–802. Dover, New York, 1972.

[2] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, New York, 1987.

[3] A. W. Appel. An efficient program for many-body simulation. *SIAM J. Sci. Stat. Comput.*, 6:85–103, 1985.

[4] J Barnes and P Hut. A hierarchical O(NlogN) force calculation algorithm. *Nature*, 324:446–449, 1986.

[5] J. E. Barnes. A modified tree code: don't laugh; it runs. *J. Comput. Phys.*, 87(1):161–170, 1990.

[6] J. A. Board, Jr., Z. S. Hakura, W. D. Elliot, D. C. Gray, W. J. Blanke, and J. F. Leathrum. Scalable implementations of multipole-accelerated algorithms for molecular dynamics. In *Scalable High-Performance Computing Conference (SHPCC94)*, pages 87–94, Washington, DC, 1994. IEEE Computer Society Press.

[7] J. A. Board, Jr, Z. S. Hakura, W. D. Elliot, and W. T. Rankin. Scalable Variants of Multipole-Accelerated Algorithms for Molecular Dynamics Applications. Technical Report 94-006, Duke University, Department of Electrical Engineering, Durham, NC, 1994.

[8] F. R. Bouchet and L. Hernquist. Cosmological simulations using the hierarchical tree method. *Ap. J. Suppl.*, 68:521–538, December 1988.

[9] I. J. Bush, I. T. Todorov, and Smith W. A DAFT DL_POLY distributed memory adaptation of the Smoothed Particle Mesh Ewald method. *Comp. Phys. Comm.*, 175(323-329), 2006.

[10] B. J. C. Cabral, J. L. Rivail, and B. J. Bigot. A Monte Carlo simulation study of a polarizable liquid: Influence of the electrostatic induction on its thermodynamic and structural properties. *J. Chem. Phys.*, 86:1467–1473, February 1987.

[11] J Carrier, L Greengard, and V Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Stat. Comput.*, 9(4):669–686, 1988.

[12] T. Darden, D. York, and L. Pedersen. Particle mesh ewald: An NlogN method for Ewald sums in large systems. *J. Chem. Phys.*, 98(12):10089–92, 1993.

[13] S. W. de Leeuw, J. W. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. i. lattice sums and dielectric constants. *Proc. Roy. Soc. Lond.*, 373(1752):27–56, 1980.

[14] H. Q. Ding, N. Karasawa, and W. A. Goddard III. Atomic level simulations on a million particles: The cell multipole method for coulomb and london nonbond interactions. *J. Chem. Phys.*, 97(6):4309–4314, 1992.

[15] H. Q. Ding, N. Karasawa, and W. A. Goddard III. The reduced cell multipole method for coulomb interactions in periodic systems with million-atom unit cells. *Chem. Phys. Letts.*, 196(1,2):6–10, 1992.

[16] Z. H. Duan and R. Krasny. An ewald summation based multipole method. *J. Chem. Phys.*, 113(9):3492–3495, 2000.

[17] Z. H. Duan and R. Krasny. An adaptive treecode for computing nonbonded potential energy in classical molecular systems. *J. Comput. Chem.*, 22(2):184–195, 2001.

[18] Z. H. Duan and R. Krasny. Treecode algorithms for computing nonbonded particle interactions. *Procs. 3rd International Workshop on Algorithms for Macromolecular Modeling*, 2002.

[19] Z. H. Duan and R. Krasny. A treecode algorithm for computing ewald summation of dipolar systems. *Procs. 2003 ACM Symposium on Applied Computing*, pages 172–177, 2003.

[20] J Dubinski. A parallel tree code. *New Astronomy*, 1:133–147, 1996.

[21] U. Essman, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. Pedersen. A smooth particle mesh ewald method. *J. Chem. Phys.*, 103(19):8577–93, 1995.

[22] P. P. Ewald. Evaluation of optical and electrostatic lattice potentials. *Ann. Phys.*, 64:253–87, 1921.

[23] D. A. Fedosov and G. E. Karniadakis. Triple-decker: Interfacing atomistic-mesoscopic-continuum flow regimes. *J. Comput. Phys.*, 228:1157–1171, 2009.

[24] D. Fincham. Leapfrog rotational algorithms. *Mol. Simul.*, 8(5):165–178, February 1982.

[25] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, 2 edition, November 2001.

[26] L Greengard and V Rokhlin. A fast algorithm for particle simulation. *J. Comput. Phys.*, 73:325, 1987.

[27] M. Griebel, S. Knapek, and G. Zumbusch. *Numerical Simulation in Molecular Dynamics*. Springer, Berlin, 2007.

[28] G. Hjaltason, G. R. Hjaltason, and H. Samet. Speeding up construction of quadtrees for spatial indexing. *VLDB*, 11:2002, 1999.

[29] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. McGraw-Hill, New York, 1981.

[30] K. Kholmurodov, W. Smith, K. Yasuoka, T. Darden, and T. Ebisuzaki. A smooth-particle mesh ewald method for dl_poly molecular dynamics simulation package on the fujitsu vpp700. *J. Comput. Chem.*, 21(13):1187–1191, 2000.

[31] A. Kia, D. Kim, and E. Darve. Fast electrostatic calculation on parallel computer clusters. *J. Comput. Phys.*, 227(8551-8567), 2008.

[32] C. Kittel. *Introduction to Solid State Physics*. Wiley, New York, 8th edition, 2004.

[33] J. Kolafa and J. W. Perram. Cutoff errors in the Ewald summation formulae for point charge systems. *Mol. Simul.*, 9(5):351, 1992.

[34] J. Kurzak and B. M. Pettitt. Massively parallel implementation of a fast muiltipole method for distributed memory machines. *J. Par. and Distr. Comp.*, 65:870–881, 2205.

[35] B. M. Ladanyi and R. M. Stratt. On the role of dielectric friction in vibrational energy relaxation. *J. Chem. Phys.*, 111(5):2008, 1999.

[36] P. Li, H. Johnston, and R. Krasny. A Cartesian treecode for screened coulomb interactions. *J. Comput. Phys.*, 228:3858–3868, 2009.

[37] K. Lindsay. *A Three-Dimensional Cartesian Tree-Code and Applications to Vortex Sheet Roll-Up.* PhD thesis, Dept. of Math., University of Michigan, Ann Arbor, MI, 1997.

[38] K. Lindsay and R. Krasny. A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *J. Comput. Phys.*, 172:879–907, 2001.

[39] D. Liu, Z.H. Duan, R. Krasny, and J. Zhu. Parallel implementation of the treecode ewald method. volume 18. Intl. Par. and Distr. Proc. Symp., 2004.

[40] F. F. Martins, B. J. Cabral, and F. M. S. Silva Fernandes. Computer simulation of liquid methyl chloride. *J. Phys. Chem*, 97:9470–9477, 1993.

[41] J. W. Perram, H. G. Petersen, and S. W. de Leeuw. An algorithm for the simulation of condensed matter which grows as the 3/2 power of the number of particles. *Mol. Phys.*, 65(4):875–893, 1988.

[42] H. G. Petersen. Accuracy and efficiency of the particle mesh ewald method. *J. Chem. Phys.*, 103:3668, 1995.

[43] M. R. Pincus and H. A. Scheraga. An approximate treatment of long-range interactions in proteins. *J. Phys. Chem*, 81(16):1579–1583, 1977.

[44] D. C. Rapaport. Multi-million particle molecular dynamics. ii. design considerations for distributed processing. *Comp. Phys. Comm.*, 62:217–228, 1991.

[45] D. C. Rapaport. *The Art of Molecular Dynamics Simulation.* Cambridge University Press, Cambridge, UK, 2nd edition, 2004.

[46] F. Reif. *Fundamentals of Statistical and Thermal Physics.* McGraw-Hill, 1 edition, June 1965.

[47] J. K. Salmon and M. S. Warren. Skeletons from the treecode closet. *J. Comput. Phys.*, 111:136, 1994.

[48] J. K. Salmon, M. S. Warren, and G. S. Winckelmans. Fast parallel tree codes for gravitational and fluid dynamical n-body problems. *Intl. J. of Sup. App.*, 8:129–142, 1994.

[49] K.E. Schmidt and M.A. Lee. Implementing the fast multipole method in three dimensions. *J. Stat. Phys.*, 63:1223, 1991.

[50] K.E. Schmidt and M.A. Lee. Ewald sums for the fast multipole method. *J. Stat. Phys.*, 89:411, 1997.

[51] Q Shi and E Geva. A relationship between semiclassical and centroid correlation functions. *J. Chem. Phys.*, 118(18):8173, 2003.

[52] W. Smith and T. R. Forester. The DL_POLY molecular simulation package. *http://www.cse.clrc.ac.uk/msi/software/DL_POLY/*, 2006.

[53] W. Smith, T. R. Forester, I. T. Todorov, and M. Leslie. *The DL_POLY_2 User Manual.* CCLRC Daresbury Laboratory, Daresbury, Warrington WA4 4AD, Cheshire, UK, 2.17 edition, December 2006.

[54] I. T. Todorov and W. Smith. DL_POLY_3: The CCP5 national UK code for molecular-dynamics simulations. *Phil. Trans.: Mathematical, Physical and Engineering Sciences*, 362(1822):1835–1852, September 2004.

[55] I. T. Todorov and W. Smith. *The DL_POLY_3 User Manual.* CCLRC Daresbury Laboratory, Daresbury, Warrington WA4 4AD, Cheshire, UK, 3.06 edition, March 2006.

[56] I. T. Todorov, W. Smith, K. Trachenko, and M. T. Dove. Dl_poly_3: new dimensions in molecular simulations via massive parallelism. *J. Mater. Chem., Roy. Soc. Chem.*, 16:1911–1918, 2006.

[57] A. Y. Toukmaji and J. A. Board. Ewald summation techniques in perspective: a survey. *Comp. Phys. Comm.*, 95:73–92, 1996.

[58] R Valdarnini. Parallelization of a treecode. *New Astronomy*, 8:691–710, 2003.

[59] Smith W. Molecular dynamics on distributed memory (mimd) parallel computers. *Theor. Chim. Act.*, 84:385–398, 1993.

[60] C. A. White, B. G. Johnson, P. M. W. Gill, and M. Head-Gordon. The Continuous Fast Multipole Method. *Chem. Phys. Letts.*, 230:8–16, November 1994.