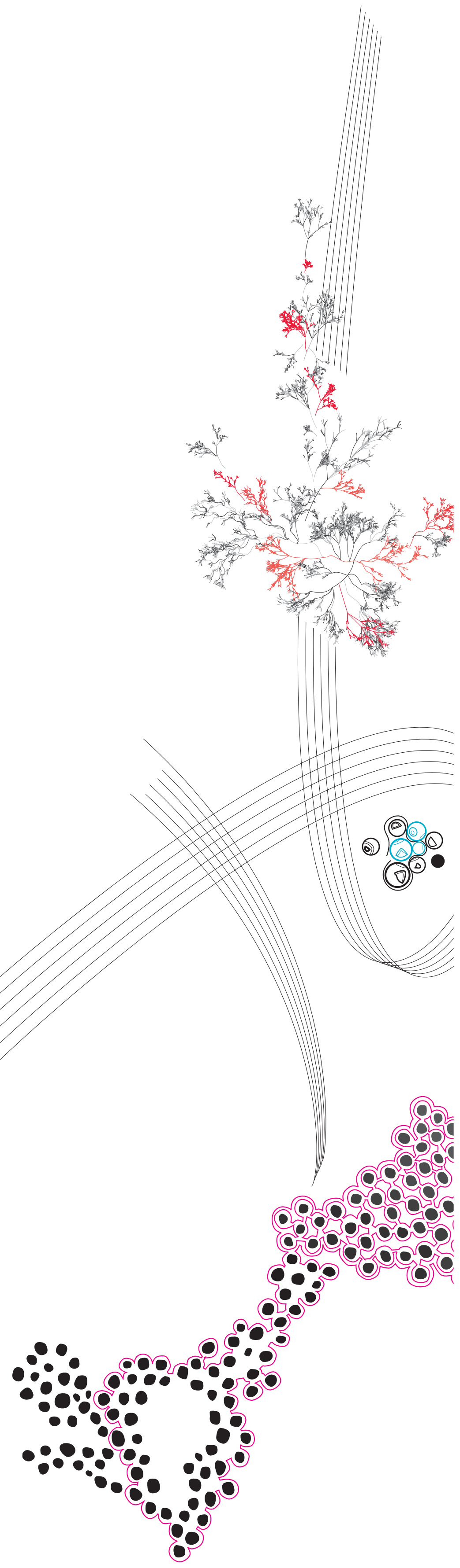
BSc Thesis Applied Mathematics and Technical Computer Science



A Simplified Study on Accelerated Particle Simulation Using the Fast Multipole Method

Mayank Thakur

Supervisor: C.Pérez Arancibia & H. Moritz

August, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

Firstly, I would like to express my heartfelt gratitude to both of my supervisors, Dr. Morritz and Dr. Pérez Arancibia. Their guidance and support were instrumental in ensuring the success of my project and keeping me on the right track. They both ensured that the mathematical and programming aspects were thoroughly integrated, acknowledging my status as a double degree student. Writing this thesis was a transformative experience, imparting valuable skills that extend beyond the thesis material and will stay with me throughout my life.

Additionally, I owe my deepest appreciation to my parents. Without their unwavering support and encouragement, I would not have been able to reach this milestone. Their belief in me has been a constant source of strength. I am also grateful to my friends, who stood by me during challenging times and provided the emotional support and companionship that kept my spirits high. Their presence made the difficult moments more bearable and their encouragement kept me motivated throughout this journey.

A Simplified Study on Accelerated Particle Simulation Using the Fast Multipole Method

Mayank Thakur[*]

August, 2024

Abstract

The Fast Multipole Method (FMM) represents a groundbreaking algorithm in computational physics, drastically reducing the computational complexity of potential calculations from $O(nm)$ to $O(n+m)$, for a system with n sources and m target points. This thesis explores the mathematical foundations and practical implementation of the FMM for accelerated particle simulations. We begin by formulating the problem of calculating potentials generated by a set of sources and review the limitations of naive algorithms. Through a detailed examination of the multipole and local expansions and their associated error bounds and translation operators, we establish the theoretical underpinnings of the FMM. Our study involves implementing and testing simplified versions of the FMM, focusing on configurations where sources and targets are well-separated, thereby avoiding the need for a quadtree data structure. The results confirm the FMM's expected linear complexity and its ability to trade accuracy for speed with high granularity. We also investigate the method's performance with randomly distributed particles.

Keywords: Fast Multipole Method, Algorithmic Complexity, Particle Simulation, Computational Physics, Multipole Expansion, Logarithmic Potential

[*]Email: m.thakur@student.utwente.nl

Contents

1 Introduction

1.1 Problem Formulation and Naive Solution

The calculation of the potentials created by a set of sources is a very common problem in physics and arises in a wide variety of situations. Some of the most notable are within electromagnetism and celestial mechanics [5, 3]. If there are a large number of objects that interact with each other by means of a slowly decaying potential, and one would like to simulate them over time, there is most likely a potential calculation is involved. However, this problem can be quite difficult to solve using standard algorithms, especially if there are a large number of objects involved.

In the most general sense, (assuming the potential is well defined and integrable) the potential is the derivative of the function which describes the net force at a point. Furthermore, the calculation of the potential can take many forms, but the scope of this thesis is limited to potentials in two-dimensions that may be calculated using a logarithm.

The following is a formal statement of the kind of problems we consider in this thesis. Let $\{q_j\}_{j=1}^n \subset \mathbb{R}$ be a set of sources located at $\{x_j\}_{j=1}^n \subset \mathbb{C}$. We consider the numerical evaluation of the following potential:

$$\phi(z) := \sum_{j=1}^{n} q_j \log(z - x_j), \qquad z \neq x_j,\ j = 1, \dots, n \tag{1}$$

at a set of target point location given by $\{y_k\}_{k=1}^m \subset \mathbb{C}$. Such potential calculations arise in many different fields. In celestial mechanics for instance, the "charge" q_j can be interpreted as the mass of a planet located at $\mathbf{x}_j = (\Re x_j, \Im x_j) \in \mathbb{R}^2$. The potential ϕ in (1) then encodes the gravitational force field produced by a collection of n planets at locations $\mathbf{x}_j = (\Re x_j, \Im x_j)$, $j = 1, \dots, n$. Indeed, the gradient $\nabla\Phi$, where $\Phi(\mathbf{r}) = \Re\{\phi(z)\}$, $\mathbf{r} = (\Re z, \Im z) \in \mathbb{R}^2$, is proportional to gravitational force field at $\mathbf{r}$.

Note that, in general, charges and coordinates are expressed using scalars and vectors in $\mathbb{R}^2$. However, the potential function, which in this case is a logarithm, is harmonic. Therefore, it can be studied as an analytic function on the complex plane, where the vectors of $\mathbb{R}^2$ are represented as complex numbers, and only the real part of the potential is reported as the potential [7]. In the context of this paper, the actual reported value is not relevant, which is why we will be working with complex potentials, as is shown in the Naive algorithm below.

At first glance, even a naive programmer would be able to come up with a simple algorithm to compute the sum (1). Such an algorithm would look like this:

Algorithm 1 Naive Potential Calculation

```
Require: $\{x_i\}_{i=0}^n \subset \mathbb{C}$, $\{q_i\}_{i=0}^n \subset \mathbb{R}$, Sources
Require: $\{y_i\}_{i=0}^m \subset \mathbb{C}$, Targets

1: potentials ← {}                                  ▷ Initialize an empty list for potentials
2: for each target point $y_j$ do
3:     $\phi(y_j) \leftarrow 0$                       ▷ Initialize the potential at $y_j$ to 0
4:     for each source point $x_i$ ($x_i \neq y_j$) do
5:         $\phi(y_j) \leftarrow \phi(y_j) + q_i \log(y_j - x_i)$
6:     end for
7:     potentials[$j$] ← $\phi(y_j)$
8: end for
```

It easy to see that this algorithm has complexity $O(nm)$, as there are $n-1$ operations for each of the m potentials. The rest of the operations in this algorithm are constant, as they are just storage or initialization steps.

1.2 Fast Multipole Method

The Fast Multipole Method (FMM) was introduced by V. Rokhlin and L.Greengard [4], and was created to calculate sums like (1), quickly. In fact, they were able to reduce the from $O(nm)$ to $O(n+m)$. Essentially, the algorithm can take as input a set of n sources and m target points, and return the potential evaluated at each of the target points, in one run. In this thesis, we will be exploring the theoretical underpinning of this method, and verifying the CPU time performance of the algorithms, through Python, in a simple source-target point configuration.

The main idea behind the FMM is to compress the information of a large number of particles into a suitable expansion, which can be used to quickly find the potential at the target points. By information, we mean the pointwise interactions between charges and potentials at the target points. In this thesis, particle will be synonymous with charge and location, or the tuple (q_i, z_i).

Essentially, we take a large number of particles, which can be distributed arbitrarily over the 2D plane, or the complex plane, and then transform them into a smaller set of equivalent particles, with different "charges", such that the calculation of the potential is relatively unchanged (i.e. the calculation is still accurate). In this case, the term "charges" is used very loosely to refer to the coefficients in a multipole expansion, because it can be used to find the potential created by a number of sources. For example, the potential at a point created by 5 sources can be found via naive calculation, or by creating a multipole expansion and evaluating that at the target point. This idea will be further explored in Section 3.

Perhaps, one of the reasons that FMM was one of the top 10 algorithms [2] of the 20th century is that it allows the user to trade accuracy for speed, with high granularity. The purpose of [4] is to develop a proper mathematical formulation of this idea. The complete algorithm makes use of a data structure called a quadtree, which provides a method for sectioning two-dimensional space into smaller spaces such that sources and potentials can share the same space. Due to complexity in the implementation of such a data structure, we will be considering particle configurations which don't need a quadtree.

In this thesis, we will examine the main features of the FMM (by selecting a smaller subset from all the mathematical components that make up the FMM) and present a straightforward implementation to demonstrate some of the main algorithm's capabilities over simple particle configurations. We also adapt the notation and the language of the algorithm used in the original Greengard-Rokhlin paper [4] to be closer to the modern equivalent.

2 Mathematical Background

In this section we will derive and cover some basic results which underpin the Fast Multipole Method. We will provide some intuitive explanations for how one might understand these formulas. Some of these formulas will be used in a later section to develop and test some simpler versions of the Fast Multipole Method.

2.1 Multipole Expansion

As shown in [4], we can generate a multipole expansion of the potential (1) at a point, z, using the charges $\{q_i, i = 1, \cdots n\}$, which are located at the points $\{x_i, i = 1, \cdots n\}$ (With the property $|x_i| \leq r$, for all i, for some $r > 0$). Such expansion is given by

$$\phi(z) = M_0(0)\log(z) + \sum_{k=1}^{\infty} \frac{M_k(0)}{z^k}, \qquad \text{(Multipole Expansion at 0)}$$

where

$$M_0(0) = \sum_{i=1}^{m} q_i \quad \text{and} \quad M_k(0) = \sum_{i=1}^{m} \frac{-q_i x_i^k}{k}. \qquad (2)$$

This expansion is valid only for $|z| > r$. This will be referred to as the **multipole expansion of the charges about the origin**, with respect to the variable z. In this formulation of the [4], each of the logarithmic terms in the potential (1) is expanded in a Taylor series, and then the terms are regrouped conveniently. For the multipole expansion, we are also provided with an error estimate of the expansion,

$$\left|\phi_0(z) - M_0(0)\log(z) - \sum_{k=1}^{p} \frac{M_k(0)}{z^k}\right| \leq \left(\frac{A}{c-1}\right)\left(\frac{1}{c}\right)^p, \qquad \text{(Expansion Error)}$$

where

$$A := \sum_{i=1}^{m} |q_i|, \qquad \text{and} \qquad c := \left|\frac{z}{r}\right| > 1,$$

which is just the truncation error of the sum. The number of terms kept in the truncated sum, p, is called the **order of the multipole expansion**. The right hand side of (Expansion Error) is termed the accuracy of the expansion. Now, if we fix z and r, and let p vary, it is easy to see that the accuracy, as a function of the order, is decreasing. We can also see that as long as $|z| > r$, the accuracy will increase as well.

The multipole expansion, paired with the local expansion, form the core computation of the FMM. As seen in (Multipole Expansion at 0), the multipole expansion cannot be used to find potentials within the circle in which the sources are located. This is problematic becuase it restricts the particle configurations which can be used with FMM. The local expansion is used to solve that issue, as it can only be used within the boundary in which it is defined. Both the multipole and local expansions are formed in equivalent ways, through perturbations, the only difference being the relative location the perturbation is carried out from. Now that we have motivated the need for these formulae and how they are to be used, we can derive them.

2.2 Moment and Local Expansion of the potential

The core idea in the derivations presented in this section is to expand the logarithm in terms of its Taylor series. Therefore, we start by taking a logarithm term as it would appear in (1), and perturb it to obtain a taylor expansion:

$$\log(z - x_j) = \log(z - z_0 + z_0 - x_j).$$

Now, we assume that $|z_0 - x_j| < |z - z_0|$. Then, using the Taylor expansion of the logarithm, as presented in Appendix A (reformulated here for notational convinience),

$$\log(w + \Delta w) = \log(w) - \sum_{k=1}^{\infty} \frac{(-\Delta w)^k}{k w^k},$$

which is valid for as long as $|\Delta w| \leq |w|$, we can choose $\Delta w = z_0 - x_j$ and $w = z - z_0$ to obtain the multipole expansion:

$$\begin{aligned}
\phi(z) &= \sum_{j=1}^{n} q_j \log(z - x_j) \\
&= \sum_{j=1}^{n} q_j \left(\log(z - z_0) - \sum_{k=1}^{\infty} \frac{(-(z_0 - x_j))^k}{k(z - z_0)^k} \right) \\
&= \left(\sum_{j=1}^{n} q_j \right) \log(z - z_0) + \sum_{k=1}^{\infty} \frac{1}{(z - z_0)^k} \sum_{j=1}^{n} \frac{-q_j (x_j - z_0)^k}{k} \\
&= M_0(z_0) \log(z - z_0) + \sum_{k=1}^{\infty} \frac{M_k(z_0)}{(z - z_0)^k},
\end{aligned}$$

where

$$M_k(z_0) = \begin{cases} \sum_{j=1}^{n} q_j, & k = 0, \\ \sum_{j=1}^{n} \frac{-q_j (x_j - z_0)^k}{k}, & k \geq 1. \end{cases} \qquad \text{(Moments)}$$

Not only does this explain the notation presented in (Multipole Expansion at 0), it yields a more general version of it. The expression,

$$M_0(z_0) \log(z - z_0) + \sum_{k=1}^{\infty} \frac{M_k(z_0)}{(z - z_0)^k},$$

is termed the **the multipole expansion of the chages about the** z_0. The expansion just found is centered at some arbitrary point z_0 instead of the origin. Indeed, if we let $z_0 = 0$, we recover (Multipole Expansion at 0). The coefficients $M_k(z_0)$ are called **the Moments of a set of charges about the point** z_0. Now, since the choice of w and Δw was arbitrary, we can pick them in the opposite way, to yield a local expansion. Choosing $\Delta w = z - z_0$ and $w = z_0 - x_j$ we obtain:

$$\begin{aligned}
\phi(z) &= \sum_{j=1}^{n} q_j \log(z - x_j) \\
&= \sum_{j=1}^{n} q_j \left(\log(z_0 - x_j) - \sum_{k=1}^{\infty} \frac{(-(z - z_0))^k}{k(z_0 - x_j)^k} \right) \\
&= \sum_{j=1}^{n} q_j \log(z_0 - x_j) + \sum_{k=1}^{\infty} (z - z_0)^k \sum_{j=1}^{n} \frac{-q_j}{k(x_j - z_0)^k} \\
&= L_0(z_0) + \sum_{k=1}^{\infty} L_k(z_0)(z - z_0)^k \\
&= \sum_{k=0}^{\infty} L_k(z_0)(z - z_0)^k,
\end{aligned}$$

where the local expansion coefficients are given by

$$L_k(z_0) = \begin{cases} \sum_{j=1}^{n} q_j \log(z_0 - x_j), & k = 0, \\ \sum_{j=1}^{n} \frac{-q_j}{k(x_j - z_0)^k}, & k \geq 1. \end{cases} \quad \text{(Local Expansion Coefficients)}$$

The expansion of the potential as,

$$\phi(z) = \sum_{k=0}^{\infty} L_k(z_0)(z - z_0)^k, \quad \text{(Local Expansion at } z_0\text{)}$$

is referred to as the **local expansion of the potential about** z_0. We do not provide error bounds for this expansion, as this expansion is not used independently in this thesis. We only use a local expansion that is formed from a multipole expansion (as will be described in the following section), which will have an error bound.

2.3 Translation Operators

We know from earlier discussion that the order of the expansion controls the accuracy of the calculation. However, it is not practical to keep increasing the order of the expansion, as that leads to increased computational complexity, as well as increased memory use. In order to make the multipole expansion more useful, [4] developed translation operators. The purpose of the translation operators is to not only shift the center of the expansions, but also to change the bounds of its accuracy. i.e. shifting the locations where the expansion is accurate. This is useful because it allows us to create a multipole expansion which is accurate in one region and then shift it, such that the same property holds elsewhere in the complex plane. One can imagine that this would be useful in the case that the sources are clustered and the targets are evenly distributed over a large area. Using the shifting property, the expansion can be moved to evaluate the potential at all required points, instead of having to do a new calculation for each target point. Together with local expansions and multipole expansion, we have all the mathematical machinery that allows FMM to achieve linear time complexity for arbitrary configurations of particles.

There are 4 kinds of translation operators, Multipole to Multipole (M2M), Multipole to Local (M2L), Local to Multipole (L2M) and Local to Local (L2L) [1]. M2M and L2L are used to shift the centers of the expansions arbitrarily. M2L and L2M are used to exchange between multipole and local expansions. The translation operators are useful because they allow the expansions to interface with the quadtree, thereby allowing it to adapt to any particle configuration. In this thesis, we will only be considering the M2L translation operator. In fact, we will first prove that a multipole expansion from an arbitary location can be transformed into a local expansion at the origin, and then show that this transformation need not be restricted to the origin. The goal being, so show that an arbitrary multipole expansion can be translated to an arbitrary local expansion.

The local expansion can be derived from the multipole expansion by expanding the logarithm and the inverse terms as Maclaurin series in terms of z, and then rearranging:

$$\begin{aligned}
\phi(z) &= M_0(z_0)\log(z-z_0) + \sum_{k=1}^{\infty} \frac{M_k(z_0)}{(z-z_0)^k} \\
&= M_0(z_0)\left(\log(-z_0) - \sum_{\ell=1}^{\infty} \frac{1}{\ell}\left(\frac{z}{z_0}\right)^{\ell}\right) + \sum_{k=1}^{\infty} \frac{M_k(z_0)}{(z-z_0)^k} \\
&= M_0(z_0)\log(-z_0) - M_0(z_0)\sum_{\ell=1}^{\infty} \frac{1}{\ell}\left(\frac{z}{z_0}\right)^{\ell} \\
&\quad + \sum_{k=1}^{\infty} M_k(z_0)\left(\frac{1}{-z_0}\right)^{k} \sum_{\ell=0}^{\infty} \binom{\ell+k-1}{k-1}\left(\frac{z}{z_0}\right)^{\ell} \\
&= M_0(z_0)\log(-z_0) + \sum_{\ell=1}^{\infty}\left(-\frac{M_0(z_0)}{\ell\cdot z_0^{\ell}} + \sum_{k=1}^{\infty} M_k(z_0)\left(\frac{1}{-z_0}\right)^{k}\binom{\ell+k-1}{k-1}\right) z^{\ell} \\
&\quad + \sum_{k=1}^{\infty} M_k(z_0)\left(\frac{1}{-z_0}\right)^{k} \\
&= \sum_{l=0}^{\infty} L_{\ell}(0) z^{l},
\end{aligned}$$

where

$$L_{\ell}(0) = \begin{cases} M_0(z_0)\log(-z_0) + \sum_{k=1}^{\infty} \frac{M_k(z_0)}{z_0^k}(-1)^k & \ell = 0 \\ -\frac{M_0(z_0)}{\ell\cdot z_0^{\ell}} + \frac{1}{z_0^{\ell}}\sum_{k=1}^{\infty} \frac{M_k(z_0)}{z_0^k}\binom{\ell+k-1}{k-1}(-1)^k, & \ell > 0. \end{cases} \tag{M2L}$$

We have taken a multipole expansion centered at at z_0, in the circle $|z_0 - z| \le R$, and expressed it as a local expansion, which is in the form of a power series. Not only that, but we found a relationship between $L_{\ell}(0) and M_k(z_0)$. Above is the derivation of the local expansion that is presented in [4]. However, it is not the most direct way of doing so, as we saw in the derivation of (Local Expansion Coefficients). The current method assumes that the coefficients (Moments) have already been calculated, and uses them to calculate the local coefficients. These methods are equivalent, and yield a local expansion. However, there is one difference, which is in the region of validity. Due to the way it is calculated, the transformation presented by [4] only converges converges on $|z| < R$. Here, R is found using $|z_0| < (c+1)R$, for some $c > 1$. The change in the region of validity is better visualized in Figure 1. On the other hand, the calculation of (Local Expansion Coefficients) only assumes that $|z - z_0| \le |z_o - x_j|$, for each source x_j. This results in the following error bounds:

$$\left|\phi(z) - \sum_{l=0}^{p} L_l(0)\cdot z^l\right| \le \frac{A(4e(p+c)(c+1)+c^2)}{c(c-1)}\left(\frac{1}{c}\right)^{p+1}. \tag{3}$$

As discussed earlier, one of the main computational difficulties of the FMM lies in the shifting of the centers of the various mutipole and local expansions, and converting between them. However, as one might have noticed, all the translations so far seem to go to the origin. The simple remedy to this is a change of the origin. For example, we have a multipole expansion about the point z_0 and we would like to construct a local expansion about z_1. However, the derived formulas only allow us to shift to the origin. We see that

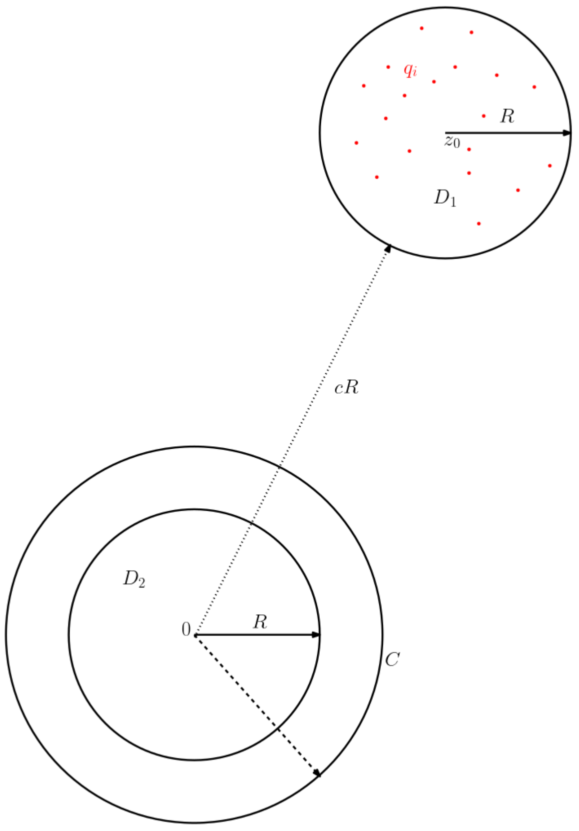


FIGURE 1: Update on the bounds of M2L transformation

$z - z_0 = z - z_1 - (z_0 - z_1)$. Now, if we let $z - z_1 = z'$, we are able to apply the M2L formula as usual to $\phi(z')$, and obtain a local expansion in terms of $z' = z - z_0$, centered at the origin. This is in fact just a local expansion about z_1. Similar logic applies to the other formulas as well, and it means that we are able to now apply arbitrary translations over the complex plane.

Now, we can start to construct some simple particle configurations to test the application and implementation of these tools that we have developed. We will also conduct some complexity analyses to ensure that the implementation is indeed linear, as postulated by [4].

3 Implementation and Complexity Analysis

In this section, we will analyze two simpler versions of the FMM. These versions are simpler in the sense that they only use the multipole expansion or the local expansion, and do not use a quadtree structure that the classic FMM uses to handle source and target points defined within the same region. Recalling the error bounds from (Expansion Error), and (3), we can see that the the target points being distant from the sources means that the multipole expansion converges, and therefore provides an accurate estimate of the potential.

The reason this works is because it represents a well separated set of points, and for such configurations, no quadtree is needed for the FMM to be used. Two sets of points, $\{a_i\}_{i=0}^{m}$, and $\{b_i\}_{i=0}^{n}$ are called well separated if there exists some R, such that both sets are bounded by a circle of radius R, and the circles themselves are separated by a distance of R. This is best visualized in Figure 2, where the red points are a_i, and the blue points are b_i.

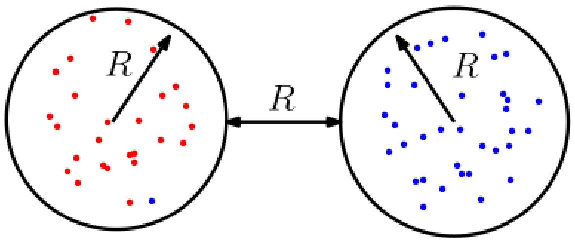


FIGURE 2: Illustration of 2 sets of well-separated points

Before we move on to checking the time efficiency of this algorithm, we will firstly define a measure of the error. One of the most basic ways of measuring the error of the potentials is the absolute error as a function of the order, p, which is the number of terms in the truncated series, and the number of sources, n:

$$\mathrm{E}_n^p(y_i) := \max_{i\in[0,n]} \left| \phi(y_i) - M_0(z_0)\log(y_i - z_0) - \sum_{k=1}^{p} \frac{M_k(z_0)}{(y_i - z_0)^k} \right|. \tag{Absolute Error For Multipole Expansion}$$

Here, $\{y_i\}_{i=0}^n$ is the set of target point, p denotes the order of the multipole expansion that is used, with z_0 the center, and $\phi(y_i)$ is the true value of the potential at that point.

In the error calculations, we compare the actual potential to the potential calculated by FMM. Therefore, in order to make error calculations, we use the naive algorithm. However, this measure by itself does not provide too much insight on the convergence of the method to the correct answer, because the absolute error is not scale invariant. i.e. if all the charges in the system are large, then the error will be large. Therefore, we use the relative error

$$\Delta_n^p(y_i) := \frac{\max_{i\in[0,n]} \left| \phi(y_i) - M_0(z_0)\log(y_i - z_0) - \sum_{k=1}^{p} \frac{M_k(z_0)}{(y_i - z_0)^k} \right|}{\max_{i\in[0,n]} \phi(y_i)}, \tag{Relative Error For Multipole Expansion}$$

instead, which normalizes the absolute error. This is more useful for seeing the accuracy of the FMM. Furthermore, seeing the error bounds defined in (Expansion Error), we postulate that the relative error should decrease exponentially. Similarly, we define a relative error function for local expansions as:

$$\Delta_n^p(y_i) := \frac{\max_{i\in[0,n]} \left| \phi(y_i) - \sum_{k=1}^{p} L_k(z_0)(y_i - z_0)^k \right|}{\max_{i\in[0,n]} \phi(y_i)}. \tag{Relative Error For Local Expansion}$$

Particle Configurations Figures 3 and 4 represent the two particle configurations that were tested for performance and accuracy. The box on the left represents the sources (also shown in red) and the points on the right are the target points (also show in blue). In order to generate points as they are shown in Figure 3, we randomly sampled 2 uniform distributions (ranging from 0 to 1) for the real and imaginary part of the sources, and the same for the targets, except the real part was offset by 4. For the charges of the sources, we sampled a standard normal distribution. Furthermore, the configuration shown in Figure 4 was developed to debug the implementation of the algorithm. In this configuration, the sources are distributed within the unit box (centered at 0) and evenly along the line $\Re z = 0.9$, and the target points are distributed within the unit box (centered at 5) and on the line $\Re z = 4.9$. An easier, linear set of points makes debugging simpler, since the formulas were checked through Google Sheets. Then, the simulation was also run on this configuration, as ideally there should not be any difference in performance or convergence between the two.

Here we use boxes instead of circles, as they are more convenient for the implementation, and more standard within the literature [4, 1].

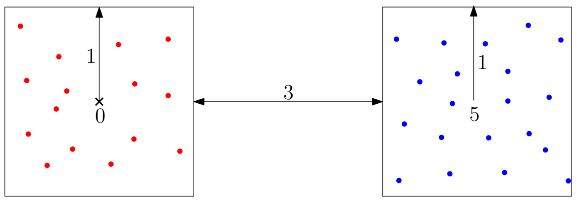


FIGURE 3: Random Sources and Targets setup for time complexity evaluation

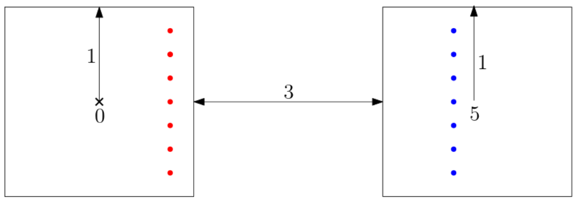


FIGURE 4: Linearly Distributed Sources and Targets for time complexity evaluation

Computation All the algorithms mentioned in this paper were implemented in Python 3.10, and run on the University of Twente computing cloud, called JupyterLab. Since JupyterLab is a shared resource, there are often spikes in performance, but the graphs cannot be generated on a local PC as it takes too long. For each configuration of number of sources, n and the order of the expansion, p, the algorithm potentials were calculated 50 times, in order to ensure a sufficiently large sample size of execution times and relative errors were collected. This was done to increase not only the significance of the data, but also to reduce any spiking in performance that would be caused by the usage of JupyterLab.

3.1 Moment to Potential

Using the principle of well separated points, and a metric for the accuracy of the implementation, we are able to commence the analysis of a version of the FMM which only uses the multipole expansion. For this analysis, we implement the computational boxes as shown in Figure 3 and 4. Using the sources we create a multipole expansion about the origin (marked with a cross). Then we use the expansion to evaluate the potential at the target points. Based off of these configurations, we can develop the following algorithm:

Algorithm 2 Multipole Potential Calculation

```
Require: $\{x_i\}_{i=0}^{n} \subset \mathbb{C}$, $\{q_i\}_{i=0}^{n} \subset \mathbb{R}$, Sources in the unit box
Require: $\{y_i\}_{i=0}^{m} \subset \mathbb{C}$, Targets in the well separated unit box
Require: $p$, order of the expansion

1: $M \leftarrow \{M_k(0)\}_{k=0}^{p}$        ▷ Initialize a list for coefficients, calculated using (2)
2: $\phi(z) \leftarrow M[0]\log(z) + \sum_{k=1}^{p} \frac{M[k]}{z^k}$        ▷ Initialize a function for potential calculation
3: potentials $\leftarrow \{\}$        ▷ Initialize an empty list for potentials
4: for each target point $y_i$ do
5:     potential$[i] \leftarrow (\phi(y_i))$        ▷ Only report real values as potential
6: end for
```

The time taken for this algorithm to run based on different inputs was measured. Furthermore, the values of the potentials generated by this algorithm were also compared with the values obtained by the naive algorithm, and was used in conjunction with (Relative Error For Multipole Expansion). Then, the time and error values are presented below.

Complexity Analysis The goal of this algorithm is to gauge the algorithmic complexity of the M2M formula, which should be linear. Therefore, we do not account for the complexity of generating the random points.

Step	Justification	Complexity
1	There are p coefficients that need to be calculated, and each calculation iterates over all the n sources	$O(np)$
2	Each function call has to evaluate p terms	$O(p)$
4, 5	p terms need to be calculated for each of the m sources	$O(mp)$

Therefore, the total complexity of this algorithm is $O(p(n+m))$ which is linear with respect to the number of points in the system.

3.1.1 Performance

Between both the random and linearly distributed source and target points, the time complexity as well as the error functions are the same. Based on Algorithm 1 and 2, we expected to see a quadratic increase in the complexity of the naive method and a linear increase in the complexity of the Multipole method. Furthermore, we expect the decay in the error function (in terms of the order) to be exponential.

In the following figures, we do not differentiate between randomly and linearly distributed, as they both obtained the same time complexity as well as accuracy results.

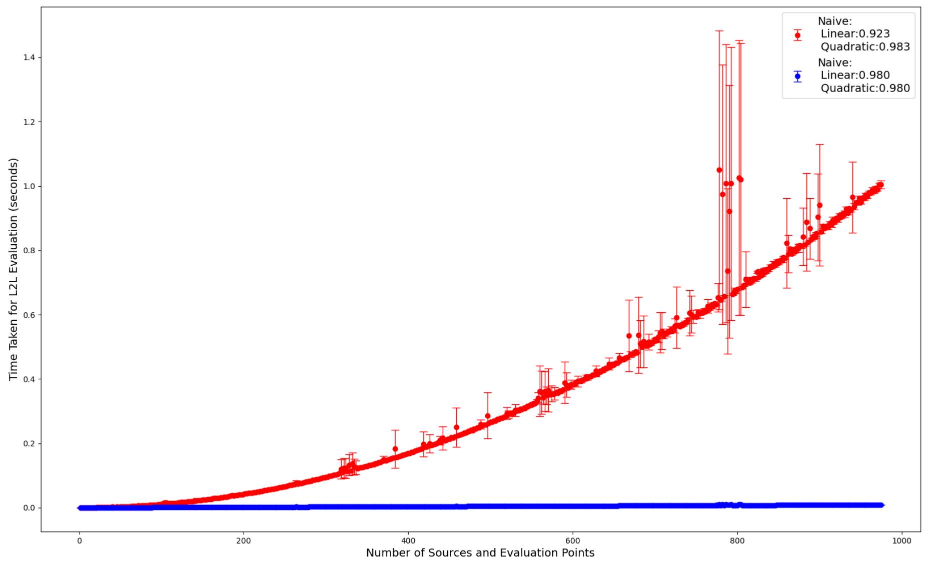


FIGURE 5: Time taken for Naive and Algorithm plotted against the number of sources and target points for Algorithm 2 (made using randomly distributed sources, targets, and charges)

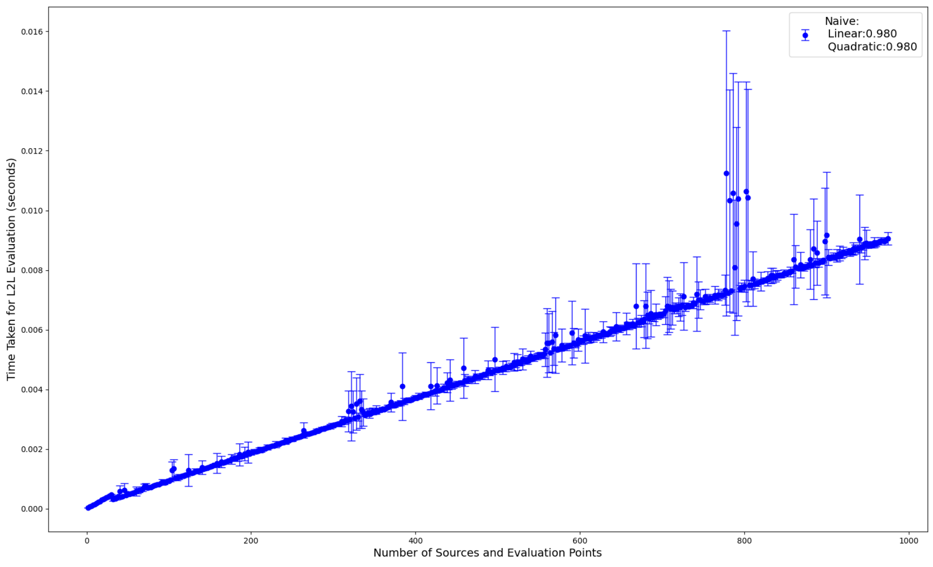


FIGURE 6: Time taken for Algorithm 2 plotted against the number of source and target points (made using randomly distributed sources, targets, and charges)

Here, we observe an (almost) linear trend in the execution times of Algorithm 2 (In Figures 5 and 6), and a non-linear one for the Naive algorithm (In Figure 5) In order to quantify the linearity of the plotted functions, we conducted a linear regression, and observed the R^2 value. In order to check if the function is quadratic, we use an automatic curve fitter and calculate the R^2 value manually. The R^2 values are present in the legends of both figures. Therefore we are certain that the growth of the Naive Algorithm is quadratic and the growth of Algorithm 2 is linear.

That being said, there is a very large amount of noise towards the end of the plot. This is likely caused by JupyterLab, as it was not present in any of the other (shorter) runs. Furthermore, we see that the R^2 values of both linear and quadratic tests for both curves achieved a very high value. This is likely because of the scale of the plot. The x values range from 1 to 1000, and the y values from 0 to 1.4, for the naive calculation, and 0 to 0.012 for Algorithm 2. At this scale, both linear and quadratic models would function relatively well, hence the high R^2 value. However, the statistical tests were merely conducted to confirm the theory, so we can still be certain that the naive algorithm is significantly slower than Algorithm 2, while being quadratic in time complexity, while Algorithm 2 is linear in time complexity.

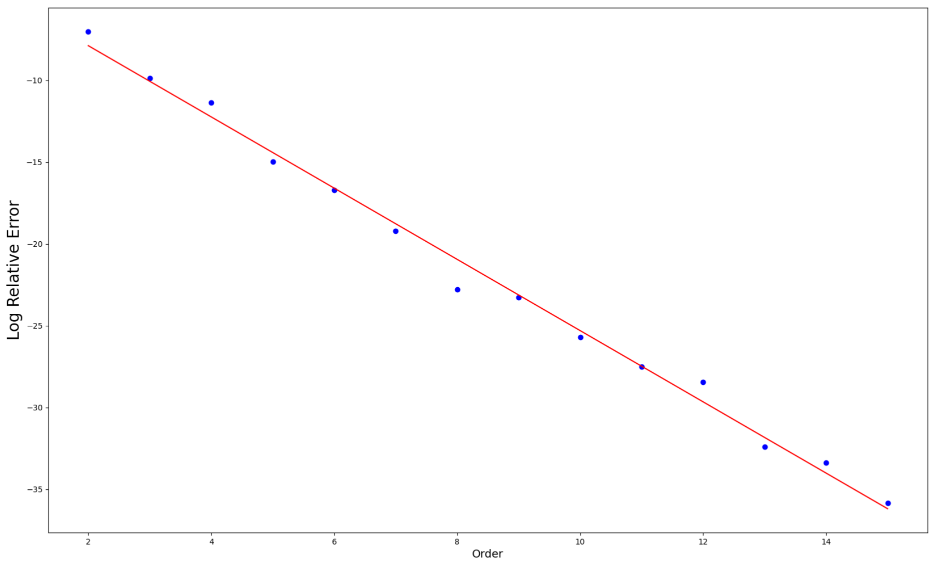


FIGURE 7: Logarithmic Relative error for Algorithm 2 plotted against the order of the expansion used (made using randomly distributed sources, targets, and charges)

It does not take many terms for the relative error to decay down to below machine precision, which is around the order of -16. This is exactly what we expected, knowing that there is an exponential decay in the error function would signify a negatively sloped line in the logarithmic-linear plot as we have in Figure 7.

3.2 Moment - to - Local to Potential

Now we analyze the creation and evaluation of the local expansion from a multipole expansion. Similar to the last algorithm, this is also a simplified version of the FMM, and the same point configurations were used (as shown in Figures 3 and 4). Based on these configurations and the formulas derived earlier, we derive the following algorithm.

Algorithm 3 Local Expansion Potential Calculation

```
Require: $\{x_i\}_{i=0}^{n} \subset \mathbb{C}$, $\{q_i\}_{i=0}^{n} \subset \mathbb{R}$, Sources in the unit box
Require: $\{y_i\}_{i=0}^{m} \subset \mathbb{C}$, Targets in the well separated unit box, centered at $z_0$
Require: $p$, order of the expansion

1: $M \leftarrow \{M_k(0)\}_{k=0}^{p}$        ▷ Initialize a list for coefficients, calculated using (2)
2: $\phi(z) \leftarrow M[0]\log(z) + \sum_{k=1}^{p} \frac{M[k]}{z^k}$        ▷ Initialize a function for potential calculation
3: $\phi(z) \leftarrow \sum_{l=0}^{\infty} L_\ell(4) z^l$,        ▷ Formed using M2L
4: potentials $\leftarrow \{\}$        ▷ Initialize an empty list for potentials
5: for each target point $y_i$ do
6:     potential$[i] \leftarrow (\phi(y_i))$        ▷ Only report real values as potential
7: end for
```

The time taken for this algorithm to run based on different inputs was measured. Furthermore, the values of the potentials generated by this algorithm were also compared with the values obtained by the naive algorithm, and was used in conjunction with (Relative Error For Local Expansion). Then, the time and error values are presented below.

Complexity Analysis The goal of this algorithm is to gauge the algorithmic complexity of the Multipole creation and M2L formula, which should be linear. Therefore, we do not account for the complexity of generating the random points.

Step	Justification	Complexity
1	There are p coefficients that need to be calculated, and each calculation iterates over all the n sources	$O(np)$
2	Each function call has to evaluate p terms	$O(p)$
3	Applying the M2L identity involves iterating over all the existing coefficients, $\{a_k\}_{k=0}^{p}$, and this calculation is carried out p times.	$O(p^2)$
5, 6	p terms need to be calculated for each of the m sources	$O(mp)$

Therefore, the total complexity of this algorithm is $O(p(n+m)+p^2)$. However, the order of the expansions is usually taken to be much smaller than the number of particles in the system, therefore, in this case, the complexity reduces to $O(p(n+m))$ again, which is linear with respect to the number of points in the system.

3.2.1 Performance

Similar to the last case, we expected an exponential decay for the error, and a quadratic and linear trend for the time complexities of the Naive algorithm and Algorithm 3 respectively.

Linearly Distributed Sources and Targets The results for this configuration are identical to results that have already been discussed, so we will omit this discussion.

Randomly Distributed Sources and Targets The simulation for the randomly generated sources and targets produced some unexpected results.

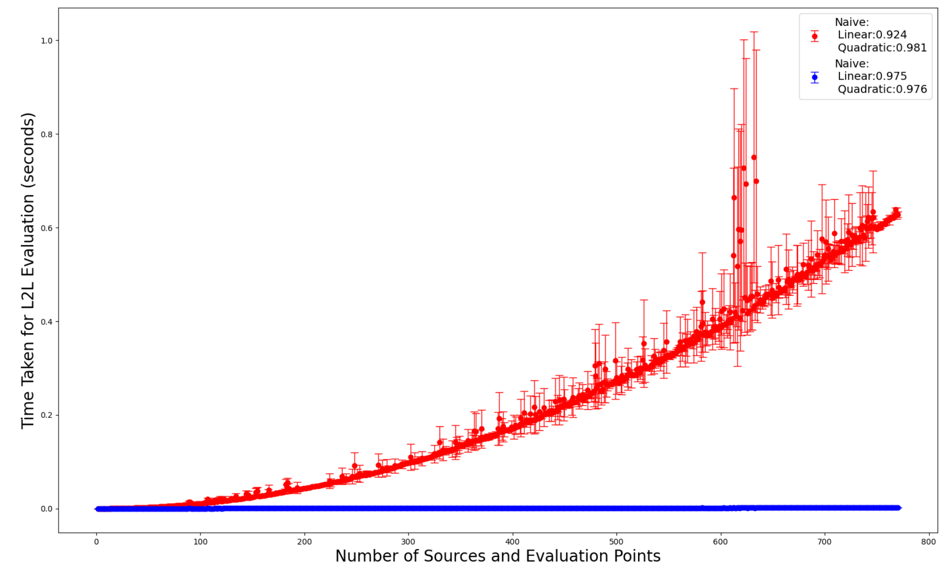


FIGURE 8: Time taken for Naive and Algorithm 3 plotted against the number of sources and target points (made using randomly distributed sources, targets, and charges)

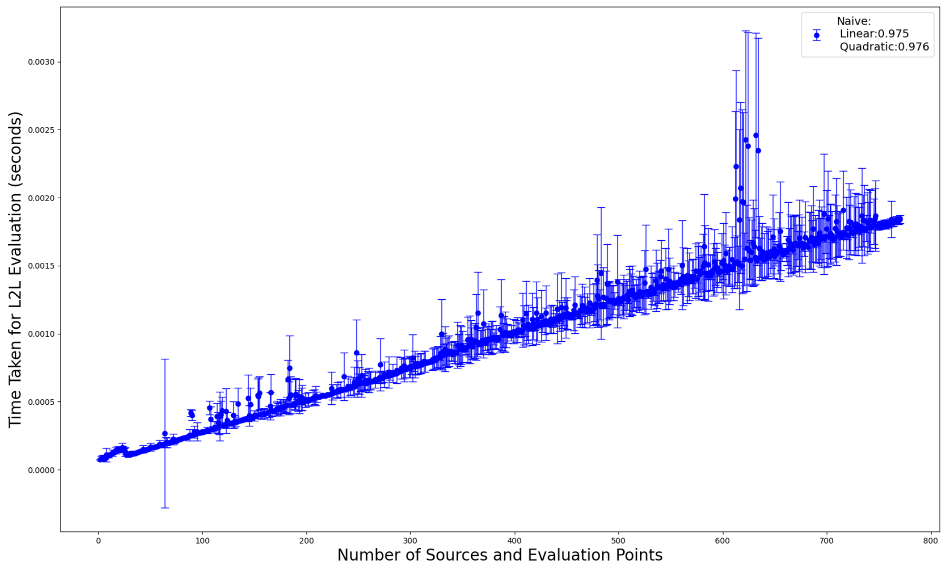


FIGURE 9: Time taken for Algorithm 3 plotted against the number of source and target points (made using randomly distributed sources, targets, and charges)

Apart from a dip in performance towards the end of the calculation, there were no other issues in the generation of this data. Based on the R^2 values reported in the legend 8, the created plots can be seen as both linear and quadratic. However, by the same reasoning as presented in the earlier section, we will conclude that Algorithm 3 scales linearly with the number of sources and targets, and the time complexity of the naive algorithm is quadratic.

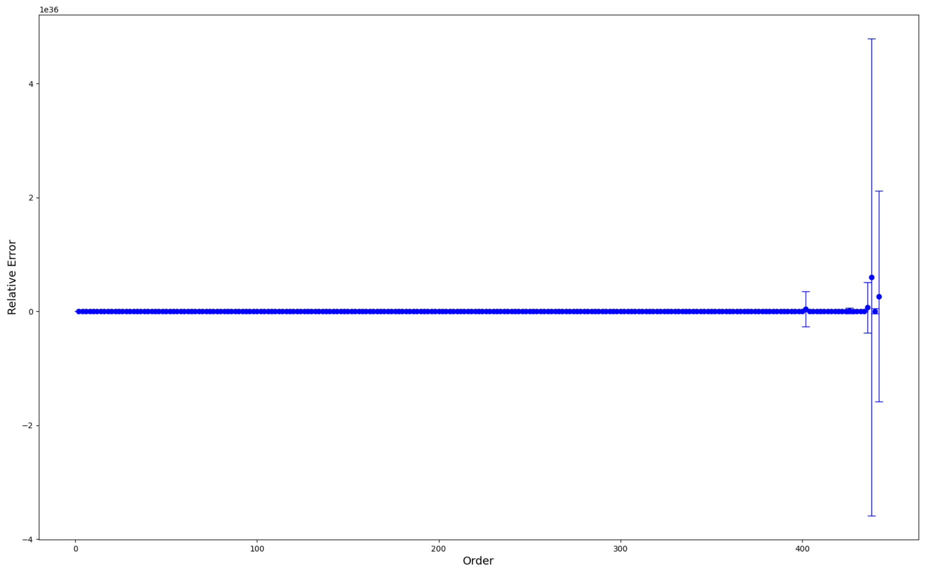


FIGURE 10: Relative error for Algorithm 3 plotted against the order of the expansion used (made using randomly distributed sources, targets, and charges)

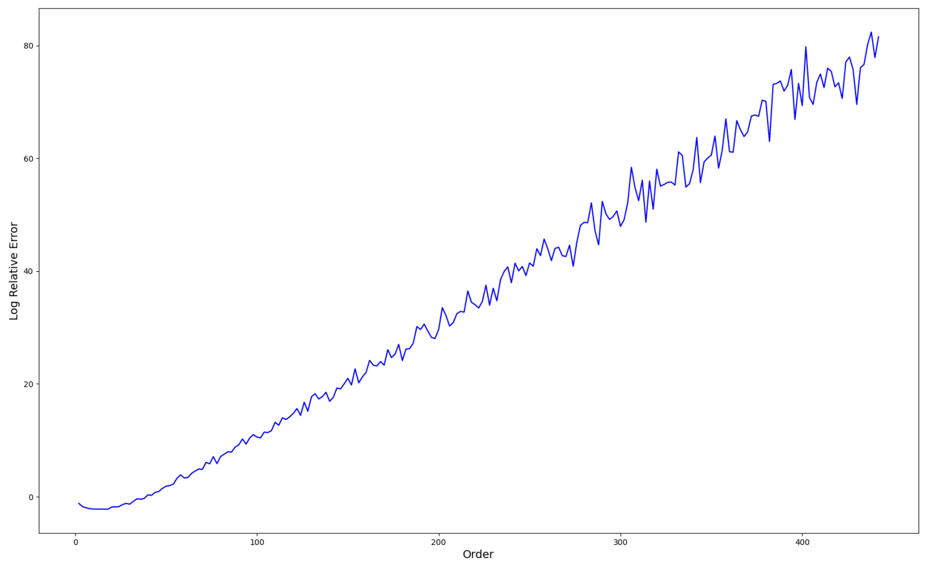


FIGURE 11: Logarithmic Relative error for Algorithm 3 plotted against the order of the expansion used (made using randomly distributed sources, targets, and charges)

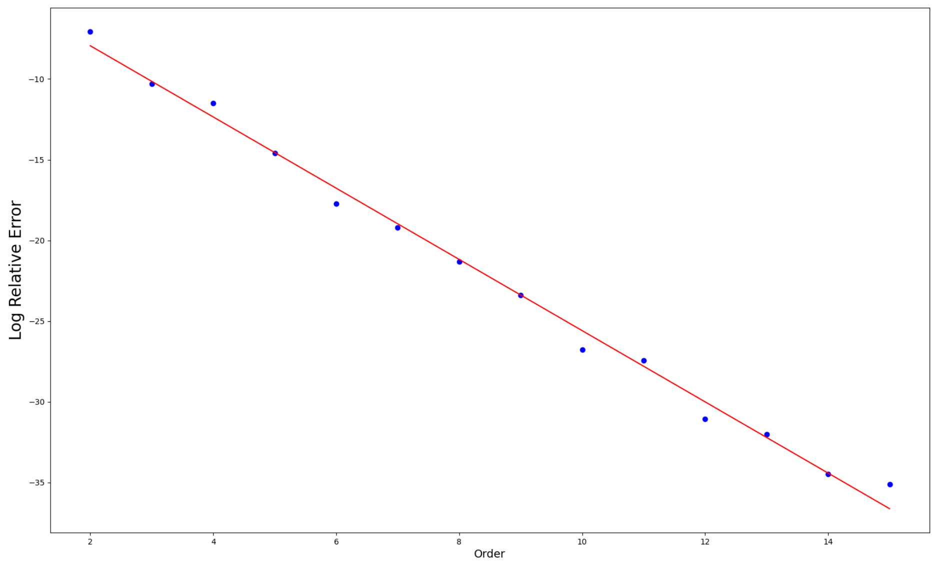


FIGURE 12: Logarithmic Relative error for Algorithm 3 plotted against the order of the expansion used (made using linearly distributed sources, targets, and charges)

However, in Figures 10 and 11 we see the major differences. The first one is that the relative error is massive. On the top left side, we can see that the scale of the graph is 10^{36}. Not only that, but the error seems to increase towards the end, as the order keeps increasing. That is why, the logarithmic plot is also increasing. It seems to suggest that the error is exponential.

That being said, the same does not apply for linearly spaced points (As seen in Figure 12). In that case, we see that the error actually still exponentially decays. Plotting the data on a logarithmic scale, we see the decreasing linear trend we saw in Figure 7. Therefore, we see a difference in the convergence of the method with respect to the configuration of the particles. This is not only unexpected, it indicates that the implementation might not complete.

Overall, the results obtained by the simulation make sense, because the algorithms that we tested do not take into account the distribution of the data at all, only the values of the data. Therefore it is sensible that the time complexity would be only dependent on the number of points and nothing else.

4 Conclusion

In this thesis, we explored the Fast Multipole Method (FMM) and its implementation for efficient particle simulation. Starting with the formulation of the potential calculation problem and the limitations of the naive $O(nm)$ approach, we introduced the FMM as a powerful alternative capable of reducing computational complexity to $O(n+m)$.

We delved into the mathematical foundation of the FMM, the multipole expansion, local expansion, translation formuals, and their associated error bounds. By implementing and analyzing two simplified versions of the FMM (namely Source to Multipole and Multipole to Local), we demonstrated the method's efficacy in both theoretical and practical terms.

Our results confirmed the expected linear complexity of the FMM compared to the quadratic complexity of the naive approach. Additionally, the exponential decay in the relative error highlighted the accuracy of the method, particularly for well-separated particle configurations.

However, unexpected behavior was observed in scenarios with randomly distributed particles, indicating a technical bug, as the theoretical results did not align. Despite this, the FMM's capability to trade accuracy for speed and its significant computational efficiency make it a critical tool in large-scale particle simulations.

References

[1] Rick Beatson and Leslie Greengard. A short course on fast multipole methods. `https://math.nyu.edu/~greengar/shortcourse_fmm.pdf`, 2001. Department of Mathematics and Statistics, University of Canterbury and Courant Institute of Mathematical Sciences, New York University.

[2] Barry A. Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM News*, 33(4), 2000. Published by the Society for Industrial and Applied Mathematics.

[3] R. Fitzpatrick. *An Introduction to Celestial Mechanics*. Cambridge University Press, 2012.

[4] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[5] John David Jackson. *Classical electrodynamics*. John Wiley & Sons, 2021.

[6] Robert Morris. *Combinatorics: Enumeration - Generating Functions, Extended Binomial Theorem*. LibreTexts, 2024. Accessed: 2024-06-30.

[7] E.B. Saff and A.D. Snider. *Fundamentals of Complex Analysis with Applications to Engineering, Science, and Mathematics: Pearson New International Edition*. Pearson Education, 2013.

A Formulas for FMM derivations

Derivation of $\log(z - z_0)$

This derivation follows from the Taylor series of the logarithm.

$$
\begin{aligned}
f(x) &= \log(1-x) \\
f'(x) &= \frac{d}{dx}\log(1-x) = -\frac{1}{1-x} \\
f''(x) &= \frac{d}{dx}\left(-\frac{1}{1-x}\right) = -\frac{d}{dx}\left((1-x)^{-1}\right) = -(-1)(1-x)^{-2} = \frac{1}{(1-x)^2} \\
f'''(x) &= \frac{d}{dx}\left(\frac{1}{(1-x)^2}\right) = 2(1-x)^{-3} \\
f^{(n)}(x) &= (-1)^{n-1}\frac{(n-1)!}{(1-x)^n} \\
f(0) &= \log(1-0) = \log(1) = 0 \\
f'(0) &= -\frac{1}{1-0} = -1 \\
f''(0) &= \frac{1}{(1-0)^2} = 1 \\
f'''(0) &= 2(1-0)^{-3} = 2 \\
f^{(n)}(0) &= (-1)^{n-1}(n-1)! \\
\log(1-x) &= \sum_{n=0}^{\infty}\frac{f^{(n)}(0)}{n!}x^n \\
&= \sum_{n=1}^{\infty}\frac{(-1)^{n-1}(n-1)!}{n!}x^n \\
&= \sum_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}x^n,
\end{aligned}
$$

which converges for $|x| \leq 1$. Using this,

$$
\begin{aligned}
\log(z-z_0) &= \log\left(-z_0\left(1-\frac{z}{z_0}\right)\right) \\
&= \log(-z_0) + \log\left(1-\frac{z}{z_0}\right) \\
\log\left(1-\frac{z}{z_0}\right) &= -\sum_{\ell=1}^{\infty}\frac{1}{\ell}\left(\frac{z}{z_0}\right)^{\ell} \\
\log(z-z_0) &= \log(-z_0) - \sum_{\ell=1}^{\infty}\frac{1}{\ell}\left(\frac{z}{z_0}\right)^{\ell},
\end{aligned}
$$

Which converges for $|z| \leq |z_0|$.

Derivation of $(z - z_0)^{-k}$

This derivation follows from the extended binomial theorem [6].

$$\begin{aligned}
(z - z_0)^{-k} &= \left(\frac{1}{-z_0}\right)^k \left(1 - \frac{z}{z_0}\right)^{-k} \\
\left(1 - \frac{z}{z_0}\right)^{-k} &= \sum_{\ell=0}^{\infty} \binom{\ell + k - 1}{k - 1} \left(\frac{z}{z_0}\right)^{\ell} \\
(z - z_0)^{-k} &= \left(\frac{1}{-z_0}\right)^k \sum_{\ell=0}^{\infty} \binom{\ell + k - 1}{k - 1} \left(\frac{z}{z_0}\right)^{\ell}
\end{aligned}$$