Proceedings of Science and Mathematics
Faculty of Science, Universiti Teknologi Malaysia
Vol. 1, 2023, Page 77-84

Application of Fast Multipole Method in Potential Calculations

Johann Zukifle, Yeak Su Hoe*
Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia
*Corresponding author: s.h.yeak@utm.my

Abstract
The purpose of this study is to investigate the application of Fast Multipole Method (FMM) in potential calculation. FMM is a mathematical algorithm used to speed up the calculation of long-range interactions in large-scale simulations involving a large number of particles or fields. The FMM makes efficient use of multipole expansions by shifting them to the centers of appropriate cells in the computational domain and converting them to local Taylor expansions for particle evaluation inside cells with truncation errors. The truncation error is related to the approximation of interactions using truncated Taylor series expansions which utilizes multipole and local expansions to approximate the influence of particles or sources on distant and nearby cells. The calculation of FMM against Direct Method (DM) was studied and the speed up, accuracy and error of FMM were highlighted. As for the result, FMM performed better than DM in all criteria. Initially, for a small number of particles, the direct method might seem efficient. However, as the number of particles grows, the FMM's logarithmic complexity allows it to surpass the direct method in efficiency. In addition, the use of FMM produced least percent total errors when the number of terms increased. The multipole expansions are truncated after a certain number of terms to balance accuracy and computational efficiency. The percent total errors decreased until a certain number of terms and the error started to become constant after the certain number of terms. Therefore, it appears that the use of FMM is highly accurate compare to the DM based on the complexity and the truncation errors.
Keywords: Fast Multipole Method; Direct Method

Introduction

Attention of a growing number of researchers. The fast multipole method (FMM) is a mathematical algorithm used to speed up the calculation of long-range interactions in large-scale simulations. Its primary use is in solving problems that involve a large number of particles or fields, such as n-body gravitational simulations or electrostatic field calculations. The FMM algorithm divides a space into multiple subspaces and provides a framework to construct interactions between particles in each of these smaller subspaces.

Thus, in this research, because it is one of the top ten algorithms in the world, the fast multipole method has been thoroughly studied. The use of the Fast Multipole Method in calculating the potential of the particle is very important and should be studied further because this method is faster than other methods because there may be a large amount that needs to be calculated faster. FMM calculations involve parameters, a set of points in a square in the plane, sources, and potentials. In many branches of physics, pair potential analysis is crucial. The evaluation of the Laplace potential is the source of the classic N-body issue, which also gave rise to tree-algorithms, the fast multipole method (FMM), and kernel independent techniques. Due to the ability to create highly scalable parallel versions of these methods, FMM for Laplace potential has had a significant impact on a variety of fields throughout the years (P. Ling et al., 2022). Fluid dynamics, electromagnetics, and acoustics can all benefit from N-body algorithms (Wang et al., 2021).

Despite its obvious advantages, the FMM algorithm suffers from a few problems that make its implementation and use challenging. The primary problem of FMM is the complexity of the algorithm. The FMM algorithm requires a complex set of mathematical operations that can be time-consuming to

implement. The computational complexity of the FMM increases exponentially with the number of particles in simulation, making it difficult to scale the algorithm to handle large simulations.

Another issue with the FMM algorithm is that it is sensitive to numerical precision. The accuracy of the FMM algorithm largely depends on the numerical precision used to represent the input data. The FMM algorithm does not work well when the numerical precision is low, leading to inaccurate results. Another problem with FMM is the challenge of implementation. The implementation of the FMM algorithm is non-trivial, which means that it requires a significant amount of time and effort to implement and test. This factor can be a hindrance for researchers and scientists attempting to use the FMM algorithm in their simulations. Finally, it is difficult to parallelize the FMM algorithm effectively. Parallel computing can help speed up computations, but the FMM algorithm's complex nature makes it difficult to decompose and parallelize the problem effectively. Improper parallelization can lead to race conditions and synchronization issues, which can cause inaccurate results.

The main purpose of this study is to implement the FMM in potential calculation, develop a Python code for FMM and compare speed up, accuracy and error for FMM with DM. the theory of the Fast Multipole Method involve the position of the potentials and charges and the expansion of mathematical method, Taylor series expansion to become the formula wanted.

Fast Multipole Method

The FMM simplifies the computation of n-body interactions which is algorithm for 2D electrostatic interactions. The multipole method works by expressing the potential as a multipole expansion, then shifting and converting these expansions to local expansions for evaluation of the potential at each particle location. The FMM makes efficient use of multipole expansions by shifting them to the centers of appropriate cells in the computational domain and converting them to local Taylor expansions for particle evaluation inside cells. The mathematical equation for FMM and DM:

$$u_i = \sum_{j=1}^{N} \log(x_i - y_i)\, q_j \tag{1}$$

$$u(x) = \log(x - c) \cdot \hat{q}_0 + \sum_{p=1}^{P} \frac{1}{(x-c)^p} \hat{q}_p \tag{2}$$

Where,

$$\hat{q}_0 = \sum_{j}^{n} q_j \tag{3}$$

$$\hat{q}_p = -\frac{1}{p} \sum_{j=1}^{N} \left(y_j - c\right)^p q_j \tag{4}$$

Based on the equation obtain previously, use the equation based on condition need in the calculation, Sources and Targets are separated.

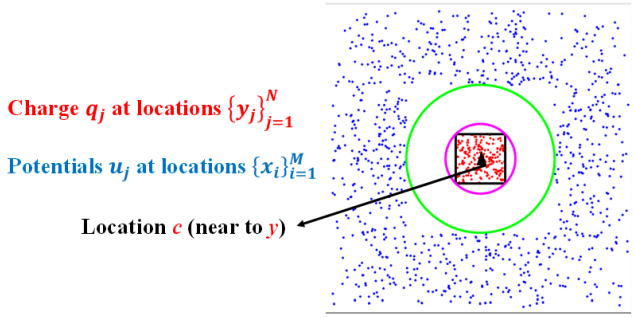

Figure 1 Source and Target Separable

Where the red dots represent the charges and the blue dots represent the potentials. The dots are separate with coloured circle to show the positions. By referring to equation (3.3), then considering the error of the calculation by multipole expansion truncated to the terms P+1. To find the approximation error :

$$u(x) = \log(x-c)\cdot\hat{q}_0 + \sum_{p=1}^{P} \frac{1}{(x-c)^p}\hat{q}_p + \frac{1}{(x-c)^{P+1}}\hat{q}_{P+1} + \cdots \qquad (5)$$

Approximation error E_p scales as $E_{P+1} \sim \left(\frac{r}{R}\right)^{P+1} = \left(\frac{a\sqrt{2}}{3a}\right)^{P+1}$, $r = a\sqrt{a}$ is the radius of magenta circle, $R = 3a$ is the radius of green circle.

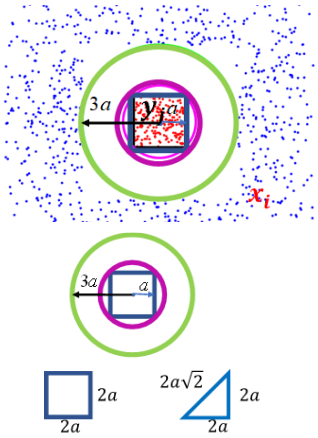

Figure 2 Radius Region For the Particle

Complexity

The complexity of FMM will follow. Flop theory is required to demonstrate the intricacy of FMM. A measure of computer performance known as flops, or floating point operations per second, is helpful in scientific computations that call for floating-point calculations. It is demonstrated how to estimate the running time of Python programming by referring to the definitions. For instance, by contrasting the failures of applying Gauss Elimination and Cramer's Rule. The time required to solve problems with enormous dimensions of numbers may take hundreds of years. In comparison, the number of flops for Gauss Eliminations are smaller than Cramer's Rule and can be solved quickly. To calculate the number of flops in an operation, the flops of each basic operation is assume to be one.

To prove the complexity for DM,

$$u_i = \sum_j^N \log(x_i - y_j)\, q_j,\ i = 1, \cdots, M \qquad (6)$$

Let, flops $+, -, \times, \div$ = 1, log( ) = k.

$$u_i = \log(x_i - y_1)\, q_1 + \cdots + \log(x_i - y_N)\, q_N \rightarrow complexity\ (1 + k + 1) \times N + N$$

Total complexity cost $M \times ((3 + k)N \approx O(MN)$

To prove the complexity for FMM,

$$\hat{q}_p = \sum_{j=1}^{N} \left[-\frac{q_j}{p} (y_j - c)^p \right], p = 1 \cdots P \tag{7}$$

Let number of flops for $\hat{q}_p$, $+, -,\ \times, \div$, power = 1, log( ) = k. Thus we will get number of flops for $\hat{q}_p$

$$\hat{q}_p = \sum_{j=1}^{N} [4\ flop], = 4N\ flop, p = 1 \cdots \tag{8}$$

Since $\hat{q}_p$ is pre-calculated,

Where N indicates the summation to the N-term and P indicates the power of P.

$$u(x_i) = \log(x_i - c) \cdot \hat{q}_0 + \sum_{p=1}^{P} \frac{1}{(x_i - c)^p} \hat{q}_p, i = 1 \cdots M \tag{9}$$

$$\sum_{p=1}^{P} \hat{q}_p = [4N\ flop] + \sum_{p=1}^{P} + = 4N + P\ flop \tag{10}$$

$$\sum_{p=1}^{P} \frac{1}{(x_i - c)^p} \hat{q}_p = [4N] + \sum_{p=1}^{P} \frac{\times}{(x_i - c)^p} = [4N] + \sum_{p=1}^{P} 3 = 4N + 3P \tag{11}$$

Thus we will get number of flops for $\hat{q}_p = 4NP$
Where N indicates the summation to the N-term and P indicates the power of P.

$$\hat{q}_p = \sum_{j=1}^{N} \left[-\frac{q_j}{p} (y_j - c)^p \right], p = 1 \cdots P \tag{12}$$

Complexity $\hat{q}_p, \hat{q}_0 = 4N(P + 1)$

$$u(x_i) = \log(x_i - c) \cdot \hat{q}_0 + \sum_{p=1}^{P} \frac{1}{(x_i - c)^p} \hat{q}_p, i = 1 \cdots M \tag{13}$$

After the calculation of $\hat{q}_p, \hat{q}_0$, the complexity for $u(x)$ is $4N(P + 1) + (2 + k) + (3)P$
The total complexity for $u(x_i), i = 1, \cdots M$ is

$$4N(P + 1) + [(2 + k) + 3P]M \approx O(P(M + N)) \tag{14}$$

Single-Level and Multi-Level Barnes-Hut

In the Fast Multipole Method (FMM), the single-level Barnes-Hut algorithm is used as a key component for approximating particle interactions. The FMM is a hierarchical method that combines the ideas of the Barnes-Hut algorithm with multipole expansions to achieve efficient calculations. In the FMM, the system is divided into a hierarchy of cells, similar to the quadtree or octree structure used in the Barnes-Hut algorithm. The cells at each level of the hierarchy represent different levels of detail in the system. At the lowest level, each cell contains a group of particles or a single particle. By referring to the diagram, let τ as the center in each level to indicates the interaction between particle, x_j and center, τ.

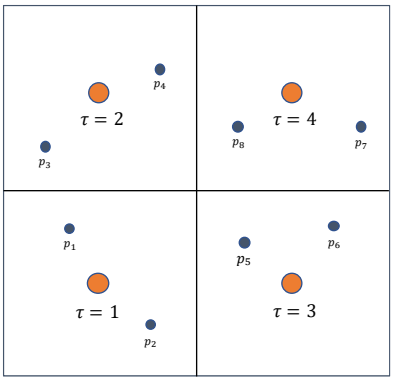

Figure 3 Single-Level Barnes-Hut

Speed Up

Computational Complexity, the FMM significantly reduces the computational complexity from $O(N^2)$ to $O(N)$, where N is the number of particles. This reduction is achieved by approximating interactions between distant particles using multipole expansions, resulting in a more efficient algorithm. Efficient Approximation, the FMM exploits the concept of multipole expansions and local expansions, allowing for a rapid calculation of interactions. By grouping distant particles into multipole expansions, the FMM achieves computational efficiency by avoiding pairwise interactions for all particles. Hierarchical Structure, the FMM utilizes a hierarchical tree structure, enabling efficient traversal and computation by focusing on nearby particles. This hierarchical approach further accelerates the calculation process and improves overall speed.

Accuracy

Control of Error, the FMM allows for controlling the error based on the user-defined parameter called the "accuracy parameter" or "theta." By adjusting this parameter, users can balance the accuracy and efficiency of the FMM. Smaller theta values provide higher accuracy at the cost of increased computational time, while larger theta values sacrifice accuracy for faster computation. Adaptive Refinement, the FMM can be combined with adaptive refinement techniques, where cells with higher potential error or discrepancy are refined further to improve accuracy in specific regions of interest. This adaptivity ensures accurate results while maintaining computational efficiency. Analytical Precision, the FMM uses analytical approximations, such as multipole expansions and local expansions, which are mathematically precise and provide accurate results within the defined error bounds.

Error

Approximation Error, the FMM introduces an approximation error by using multipole expansions to approximate interactions between distant particles. However, this error can be controlled by adjusting the accuracy parameter (theta). Smaller theta values result in lower approximation errors. Truncation Error, in FMM implementations, truncation errors can arise due to finite multipole and local expansion orders. Increasing the expansion order can reduce the truncation error, but it comes at the cost of increased computational complexity. Round-off Error, as with any numerical computation, round-off errors may occur due to finite precision arithmetic. However, the impact of round-off errors can be minimized through careful implementation and appropriate numerical techniques.

Result and Discussion

Implementation of Fast Multipole Method

A pseudocode for the Fast Multipole Method (FMM) outline the step-by-step process involved in implementing the algorithm. While the specific details of the flowchart may vary depending on the implementation and problem being solved, the general overview of the key components that would typically be included:

Firstly, Input Data. The flowchart starts with the input data required for the FMM, such as the coordinates and properties of particles, the desired level of accuracy, and any other parameters specific

to the problem. Secondly, Initialization. The FMM initialization step involves setting up the necessary data structures, including the construction of the hierarchical tree (quadtree) that represents the spatial subdivision of the particle domain. Then, Tree Construction. This step involves constructing the hierarchical tree structure by recursively subdividing the particle domain into smaller boxes or cells. Each box represents a group of particles and is assigned a level based on its position in the tree.
Next, Multipole Expansion. The FMM performs multipole expansions at higher levels of the tree. This involves computing and storing the multipole moments for each box, which capture the interactions between groups of particles within that box.

After that, Downward Pass. During the downward pass, the FMM traverses the tree from higher to lower levels, applying the multipole-to-local translations. This step involves calculating the local expansions for each box based on the multipole moments of its parent boxes. Besides that, Local Interaction. The FMM performs local interactions between particles within the same box or between neighboring boxes at the same level. This step involves computing the interactions between individual particles using the local expansions. Upward Pass. In the upward pass, the FMM traverses the tree from lower to higher levels, applying the local-to-multipole translations. This step involves aggregating the local expansions from child boxes to compute the multipole moments of parent boxes.

As for Far-field Interaction, the FMM computes the interactions between distant boxes (far-field interactions) using the multipole moments. This step involves approximating the interactions between groups of particles at a higher level based on the multipole expansions. Error Control. To ensure accuracy, the FMM may include error control mechanisms. These mechanisms evaluate the accuracy of the approximation at each step and adjust the level of detail or precision as needed. Next, Output. The final output of the FMM includes the computed potentials, forces, or other quantities of interest for the particles. Finally, Termination. The flowchart concludes with the termination of the FMM algorithm, marking the end of the computation.

Speed Up based on the Complexity for FMM

The FMM used is to reduce the time complexity needed to compute pair potential for many particle systems. To calculate the complexity for the direct method and FMM. The used of Python programming is used. The Python programming is related to time complexity testing of the Fast Multipole Method (FMM) and the Direct Method. It measures the runtime of these methods for different numbers of particles and plots the results.

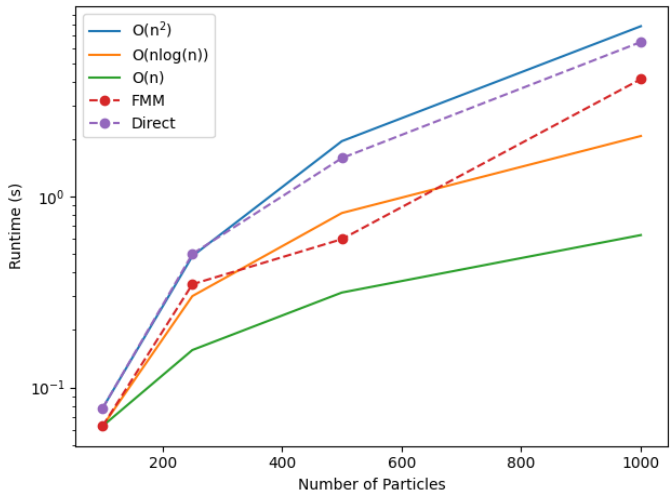

Figure 4 Graph Time complexity scaling for FMM and direct calculation of potential.

In terms of complexity, the FMM has an advantage over the direct method for problems with a large number of particles. The FMM's complexity scales more favorably with the number of particles, allowing it to handle larger problems efficiently. On the other hand, the direct method becomes computationally expensive as the number of particles increases due to its quadratic complexity. By using FMM shows that the runtime to calculate the number of particles is faster compared to DM as the

number of particles is increasing which are 1000 particles. Initially, for a small number of particles, the direct method might seem efficient. However, as the number of particles grows, the FMM's logarithmic complexity allows it to surpass the direct method in efficiency. This is because the FMM constructs a hierarchical data structure, which reduces the number of pairwise interactions required.

Accuracy and Error on Fast Multipole

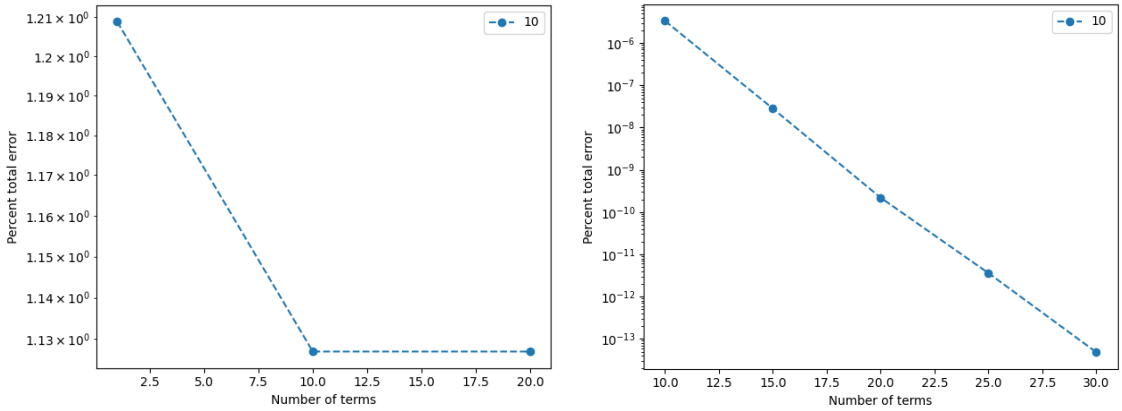

Figure 5 Graph Truncation Error against Number of Terms (1000 random particles)

Figure 5 shows that the percent total error for random number of particles which are 1000 particles against the number of terms decreasing at error of 1.21% to 1.13% from the term 1st to term 10th and the error start to constant from term 10th to the upcoming terms. By referring to the complexity test on previous subtopic stated that the FMM complexity grows more slowly than the direct method as the number of particles increases. This makes the FMM a more efficient choice for problems with a large number of particles. Thus this shown why did the error started to be constant as the number of terms is increasing and there are slightly errors on the early terms. Based on the Figure 5, the graphs show that the percent total error for random number of particles against the number of terms from the tenth terms can be stated to be significantly constant since the error different between the axis is completely the same.

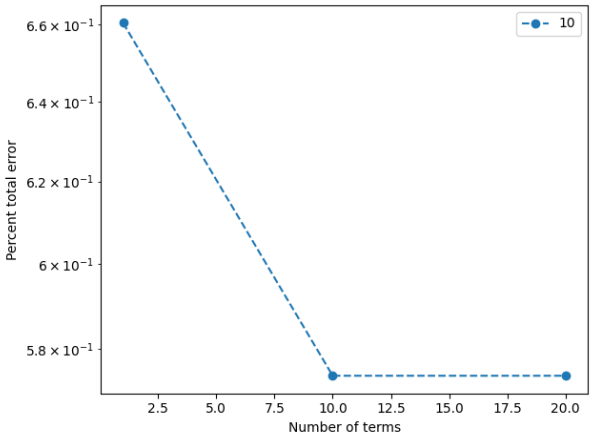

Figure 6 Graph Truncation Error against Number of Terms (2000 random particles)

Conclusion

In conclusion, FMM perform better in complexity and produce less error and more accurate compare to DM since FMM is a powerful algorithm for efficiently computing interactions between particles in N-body problems. Based on the complexity, FMM perform well compared to DM although DM perform better involving small number of particles but FMM perform well when involved large number of particles. The FMM is a fundamental and highly efficient algorithm for N-body simulations and its impact on

computational science and engineering cannot be overstated, providing a powerful tool for tackling complex problems and advancing our understanding of various phenomena.

Acknowledgement

I wish to express my sincere gratitude to all who have contributed throughout the course of this work

References

Baczewski, A. D., & Ying, L. (2019). *A Review of the Fast Multipole Method: Recent Advances and Future Directions.* Archives of Computational Methods in Engineering, 26(1), 1-61.

Barroso-Luque, L. (2018). *A Python Implementation of the Fast Multipole Method*.

Beatson, R., & Greengard, L. (n.d.). *A short course on fast multipole methods*.

Chemistry. Chemical Reviews, 114(1), 28-57.

Deserno, M. (1998). *Fast Multipole Methods for the Simulation of Biomolecular Systems.* Computer Physics Communications, 117(1-2), 1-10.

Greengard, L., & Rokhlin, V. (1987). *A Fast Algorithm for Particle Simulations.* Journal of Computational Physics, 73(2), 325-348.

Greengard, L., & Wandzura, S. (1998). *Fast Multipole Methods*. Courant Institute of Mathematical Sciences.

Jiang, W., et al. (2020). *Accelerating the Fast Multipole Method on GPUs and MICs: A Comparative Study.* Journal of Computational Science, 42, 101124.

L. Greengard and V. Rokhlin. *On the Efficient Implementation of the Fast Multipole Algorithm.* Department of Computer Science Research Report 602, Yale University (1988).

Lutz, M. (2013). *Python Pocket Reference (5th ed.).* O'Reilly Media.

Marouane, M. (2020). *Python for Data Science: A Step-by-Step Guide to Master the Basics of Data Science in Python*. Packt Publishing.

Martinsson, G. (2014). *The Fast Multipole Method*. The University of Colorado.

P. Ling, Michael, et al. *High Performance Evaluation of Helmholtz Potentials Using the Multi-Level Fast Multipole Algorithm*. Dec. 2022, p. 1.

Ramalho, L. (2015). *Fluent Python: Clear, Concise, and Effective Programming.* O'Reilly Media.

Reitz, K. (2020). *The Hitchhiker's Guide to Python: Best Practices for Development.* O'Reilly Media.

Van Rossum, G. (2007). *The Python Language Reference Manual.* Network Theory Ltd.

Wang, Tingyu, et al. *ExaFMM: A High-Performance Fast Multipole Method Library with C++ and Python Interfaces*. 30 May 2021.

White, J. A., & Head-Gordon, T. (2014). *Fast Multipole Methods in Quantum*