

Old Dominion University

ODU Digital Commons

Electrical & Computer Engineering Theses & Dissertations

Electrical & Computer Engineering

Summer 1998

The Cluster Multipole Algorithm for Far-Field Computations

Rakesh R. Patel

Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/ece_etds



Part of the [Computer Engineering Commons](#), [Computer Sciences Commons](#), and the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Patel, Rakesh R.. "The Cluster Multipole Algorithm for Far-Field Computations" (1998). Doctor of Philosophy (PhD), Dissertation, Electrical & Computer Engineering, Old Dominion University, DOI: 10.25777/kb79-yz35
https://digitalcommons.odu.edu/ece_etds/104

This Dissertation is brought to you for free and open access by the Electrical & Computer Engineering at ODU Digital Commons. It has been accepted for inclusion in Electrical & Computer Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

**THE CLUSTER MULTIPOLE ALGORITHM FOR FAR-FIELD
COMPUTATIONS**

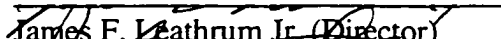
by


Rakesh R. Patel
M.S.E.E., May 1993, Old Dominion University, Norfolk, VA.


A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY
ELECTRICAL ENGINEERING
OLD DOMINION UNIVERSITY
July 1998

Approved By:


James F. Leathrum Jr. (Director)


John Stoughton (Member)


Martin Meyer (Member)


Chester Grosch (Member)

ABSTRACT

THE CLUSTER MULTIPOLE ALGORITHM FOR FAR-FIELD COMPUTATIONS.

Rakesh R. Patel

Old Dominion University, Norfolk, VA. July, 1998.

Director: Dr. James F. Leathrum Jr.

Computer simulations of N -body systems are beneficial to study the overall behavior of a number of physical systems in fields such as astrophysics, molecular dynamics, and computational fluid dynamics. A new approach for computer simulations of N -body systems is proposed in this research. The new algorithm is called the Cluster Multipole Algorithm (CMA). The goals of the new algorithm are to improve the applicability to non-point sources and to provide more control on the accuracy over current algorithms. The algorithm is targeted to applications that do not require rebuilding the data structure about the system every time step due to current limitations in the construction of the data structure. Examples of slowly changing systems can be found in molecular dynamics, capacitance, and computational fluid dynamics simulations. As the data structure development is improved, the new algorithm will be applicable to a wider range of applications.

The CMA exhibits the flexibility of both Appel's algorithm and the Fast Multipole Method (FMM) without sacrificing the order of computation ($O(N)$) for "well structured" clusters. The CMA provides more control on the accuracy of computations as compared to both the FMM and Appel's algorithm resulting in enhanced performance.

A set of requirements are imposed on the data structures which are applicable, to maintain $O(N)$ computation. However, the algorithm is capable of handling a wide range of data structures beyond the FMM.

To My Parents, Sister, and Wife.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. James Leathrum Jr., for his support, direction, and encouragement to achieve this milestone. I would also like to thank my committee for their support and assistance: Dr. John Stoughton, Dr. Chester Grosch, and Dr. Martin Meyer. Dr. Andrew Appel was also helpful in providing insight into his work.

Special thanks goes to Dr. Vishnu Lakdawala for his insightful discussions throughout my graduate career at ODU and to Dr. Roland Mielke for believing in me and giving me an opportunity to be an instructor for the very first time.

I would also like to thank all EE and CS faculty for their excellent instructions during my graduate courses. Also, thanks to all EE graduate students for making my life enjoyable.

During my entire graduate studies, my family's support and patience was unmatched. Special thanks to my parents for getting me this far and to my wife for believing in me. I would like to dedicate this work to my parents, sister, and wife, whose sacrifice and support during this endeavor is unimaginable.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	vi
LIST OF FIGURES	ix
LIST OF TABLES	x
CHAPTER 1	1
INTRODUCTION	1
1.1. BACKGROUND.....	1
1.2. OBJECTIVE OF RESEARCH	5
1.3. OUTLINE OF THESIS.....	8
CHAPTER 2	9
N-BODY ALGORITHMS	9
2.1. BACKGROUND.....	9
2.2. COMPUTATIONAL STRUCTURE	9
2.2.1. <i>Discrete Time Step Error</i>	12
2.3. DIRECT COMPUTATION	13
2.4. APPEL'S ALGORITHM.....	15
2.5. FAST MULTIPOLE METHOD (FMM)	19
2.5.1. <i>Algorithm Description</i>	21
2.5.2. <i>Interaction List Improvements</i>	29
2.5.3. <i>Grid Structure Requirements</i>	30
2.5.4. <i>The Adaptive FMM</i>	30
2.6. ESSELINK'S ALGORITHM.....	33
2.7. BARNES-HUT METHOD.....	34
CHAPTER 3	37
THE CLUSTER MULTIPOLE ALGORITHM	37
3.1. INTRODUCTION	37
3.2. LIMITATIONS OF APPEL'S ALGORITHM AND THE FMM	40
3.2.1. <i>Non-Point Source Applications</i>	40
3.2.2. <i>Accuracy</i>	43
3.3. THE CLUSTER-MULTIPOLE ALGORITHM.....	45
3.3.1. <i>Algorithm</i>	45
3.4. DATA STRUCTURE	52
3.4.1. <i>Requirements</i>	53
3.4.2. <i>The CMA Clustering Process</i>	55
3.5. TIME COMPLEXITY.....	56
3.5.1. <i>Data Structure</i>	56
3.5.2. <i>Algorithm</i>	57
3.6. APPLICABILITY TO NON-POINT SOURCE APPLICATIONS.....	58
CHAPTER 4	59

SIMULATION RESULTS.....	59
4.1. INTRODUCTION	59
4.2. BACKGROUND.....	59
4.3. COMPARISON OF THE CMA WITH THE FMM.....	60
4.3.1. <i>FMM Results</i>	61
4.3.2. <i>Appel Algorithm's Results</i>	62
4.3.3. <i>CMA Results</i>	63
4.4. INTERACTION LIST ANALYSIS	66
CHAPTER 5	69
CONCLUSION.....	69
5.1. CONCLUSIONS.....	69
5.2. FUTURE RESEARCH.....	70
REFERENCES.....	72
APPENDIX A	76
ALTERNATIVE GRID STRUCTURES FOR THE FAST MULTIPOLE METHOD.....	76
A.1. INTRODUCTION.....	76
A.2. CIRCLE OF CONVERGENCE	77
A.3. GRID STRUCTURE REQUIREMENTS.....	78
A.3.1. <i>Bounded Number of Particles Per Box</i>	79
A.3.2. <i>Bounded Size of the Interaction List</i>	80
A.3.3. <i>Separation of Convergence Circles</i>	81
A.3.4. <i>Bounded Parent-Child Relationship</i>	82
A.3.5. <i>Bounded Number of Near Neighbors</i>	82
A.4. TWO-DIMENSIONAL GRID STRUCTURES.....	83
A.4.1. <i>Square Grid</i>	84
A.4.2. <i>Triangular Grid</i>	85
A.4.3. <i>Hexagonal Grid</i>	88
A.4.4. <i>Comparison of Grid Structures</i>	93
A.4.5. <i>Formation of Other Grid Regions</i>	94
A.5. THREE-DIMENSIONAL GRID STRUCTURES.....	95
A.5.1. <i>Cube</i>	96
A.5.2. <i>Tetrahedron</i>	96
A.5.3. <i>Other Structures</i>	97
BIOGRAPHY	98

LIST OF FIGURES

FIGURE 1.1. AN APPROXIMATION OF A GROUP OF PARTICLES BY A SINGLE PARTICLE	2
FIGURE 1.2. RELATIVE SIZE AND DISTANCE BETWEEN GROUPS DICTATE APPROXIMATION RULES.	4
FIGURE 2.1. A CLASSICAL N-BODY PROBLEM.....	11
FIGURE 2.2. PSEUDO-CODE FOR THE DIRECT ALGORITHM.....	14
FIGURE 2.3. TWO CLUSTERS OF PARTICLES IN APPEL'S ALGORITHM.....	16
FIGURE 2.4. PSEUDO-CODE FOR THE APPEL'S ALGORITHM.....	18
FIGURE 2.5. WELL-SEPARATEDNESS IN THE FMM.....	20
FIGURE 2.6. HIERARCHICAL GRID FOR THE FMM.....	22
FIGURE 2.7. CIRCLES OF CONVERGENCE.....	23
FIGURE 2.8. INTERACTION LIST FOR THE FMM.....	24
FIGURE 2.9. COMPOUNDING OF THE INTERACTION LISTS DURING THE TREE WALK.	25
FIGURE 2.10. PSEUDO-CODE FOR THE FMM.....	27
FIGURE 2.11. PARENTAL INTERACTION LIST.....	28
FIGURE 2.12. A DIVISION OF CELLS FOR THE ADAPTIVE FMM.	32
FIGURE 3.1. AN EXAMPLE OF A NON-POINT SOURCE.....	41
FIGURE 3.2. A GAUSSIAN GRID STRUCTURE.....	54
FIGURE 4.1. ERROR VERSES TIME PLANE FOR THE ORIGINAL SQUARE GRID FMM.	61
FIGURE 4.2. ERROR VERSES TIME PLANE FOR APPEL'S ALGORITHM FOR VARIOUS VALUES OF δ	62
FIGURE 4.3. ERROR VERSES TIME PLANE FOR THE CMA: UNIFORM DISTRIBUTION.....	64
FIGURE 4.4. THE UNIFORM CMA RESULTS FOR HIGHER VALUES OF δ	66
FIGURE 4.5. INTERACTION LIST FOR $\delta = 0.44$	68
FIGURE 4.6. INTERACTION LIST FOR $\delta = 0.50$	68
FIGURE A.1. THE SEPARATION BETWEEN THE CIRCLES OF CONVERGENCE AND THE INTERPRETATION OF C....	78
FIGURE A.2. A TWO-DIMENSIONAL TRIANGULAR GRID. DIFFERENT LINE PATTERNS ARE USED TO REPRESENT THE PARENT-CHILD RELATIONSHIP.....	85
FIGURE A.3. THE INTERACTION LIST FOR BOX B IS DENOTED BY THE DARK BOXES.....	87
FIGURE A.4. A TRIANGULAR GRID USING ISOSCELES TRIANGLES.	88
FIGURE A.5. A TWO-DIMENSIONAL HEXAGONAL GRID.....	89
FIGURE A.6. ALTERNATIVE PARENT-CHILD RELATIONSHIPS FOR THE HEXAGONAL GRID.....	90
FIGURE A.7. A MODIFIED PARENT-CHILD RELATIONSHIP FOR THE HEXAGONAL GRID.	91
FIGURE A.8. THE CIRCLE OF CONVERGENCE FOR THE PROPOSED PARENT-CHILD RELATIONSHIP.	91
FIGURE A.9. THE BOXES DRAWN USING THICK LINES ARE IN THE INTERACTION LIST OF THE BOX B.	93

LIST OF TABLES

TABLE 4.1. RELATIONSHIP BETWEEN δ AND SEPARATION IN BOXES.....	65
TABLE A.1. A COMPARISON BETWEEN THREE TWO-DIMENSIONAL GRIDS	94

CHAPTER 1

INTRODUCTION

1.1. Background

A new algorithm is devised for the solution of the N-body problem which is better suited to non-point source applications. The goals of the new algorithm are to improve the applicability to non-point sources and to provide more control on the accuracy over current algorithms. The algorithm is $O(N)$ for “well structured” clusters.

The study of physical systems by particle simulation is called the “*many-body*” or “*N-body*” problem. The studies involve computing the interaction of many bodies with each other. Such studies are conducted in celestial mechanics, plasma physics, electrostatics, molecular dynamics, and fluid mechanics, as well as semiconductor device simulation [20]. As an example, the simulation for celestial mechanics or molecular dynamics finds the trajectories of each particle over some time interval, given the initial position, the initial velocity, the external force, and the nature of the forces that the particles exert on each other.

The number of particles used for such studies is quite large. It is estimated that to get insight into three-dimensional turbulent flow, about one million particles are needed [20]. Thus, such simulations require intensive and prolonged computations involving on the order of 10^{12} body to body interactions for one time step of the computation. Thus, the question of efficiency is of great interest.

There are two strategies that can be applied in the quest for more knowledge from bigger and better particle simulations. One can use the brute force approach: simple algorithms on bigger and faster machines. This classical approach to the N-body problem calculates the interaction between all possible pairs of particles through a direct summation of all pairs. This approach is termed the *direct* method or the *Particle-Particle* method as it computes all particle-particle interactions explicitly. However, the computationally intensive nature of the N-body simulations makes this approach not a viable option for large N-body simulations.

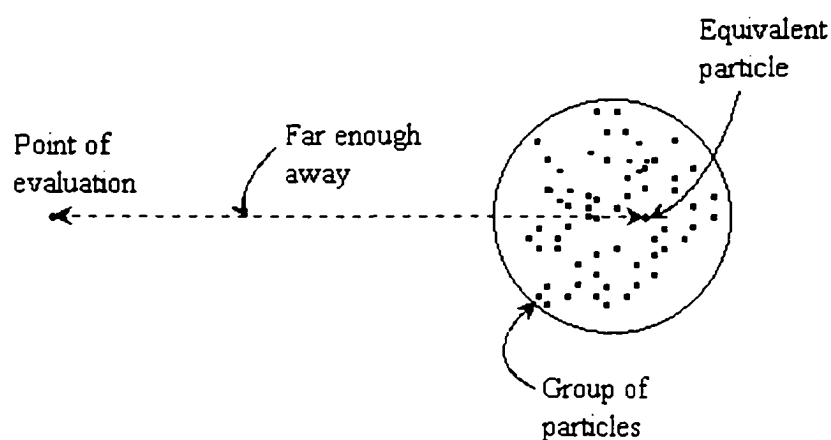


Figure 1.1. An Approximation of a Group of Particles by a Single Particle.

The second approach is to try to develop better algorithms that can solve problems to a desired accuracy using much less computational power. In 1687, Isaac Newton gave a powerful insight into the nature of physical systems: If the magnitude of interaction between particles falls off rapidly with distance, then the effect of a large group of particles may be approximated by a single equivalent particle, if the group of particles is

far enough away from the point at which the effect is being evaluated, as shown in Figure 1.1. The hierarchical application of this insight implies that the farther away the particles, the larger the group that can be approximated by a single particle (see Figure 1.2). Although Newton arrived at his powerful insight in the context of gravitation, hierarchical N-body methods based on it have found increasing applicability in various problem domains. Two situations arise again and again in a variety of particle algorithms:

1. Local computation of short-range interactions.
2. Computing global sums for long-range interactions.

Ultimately, the most powerful method will be a combination of these two approaches - a sophisticated algorithm running on a parallel machine. However, a problem with this approach is that the implementation of complicated numerical and computational methods on parallel computers is difficult.

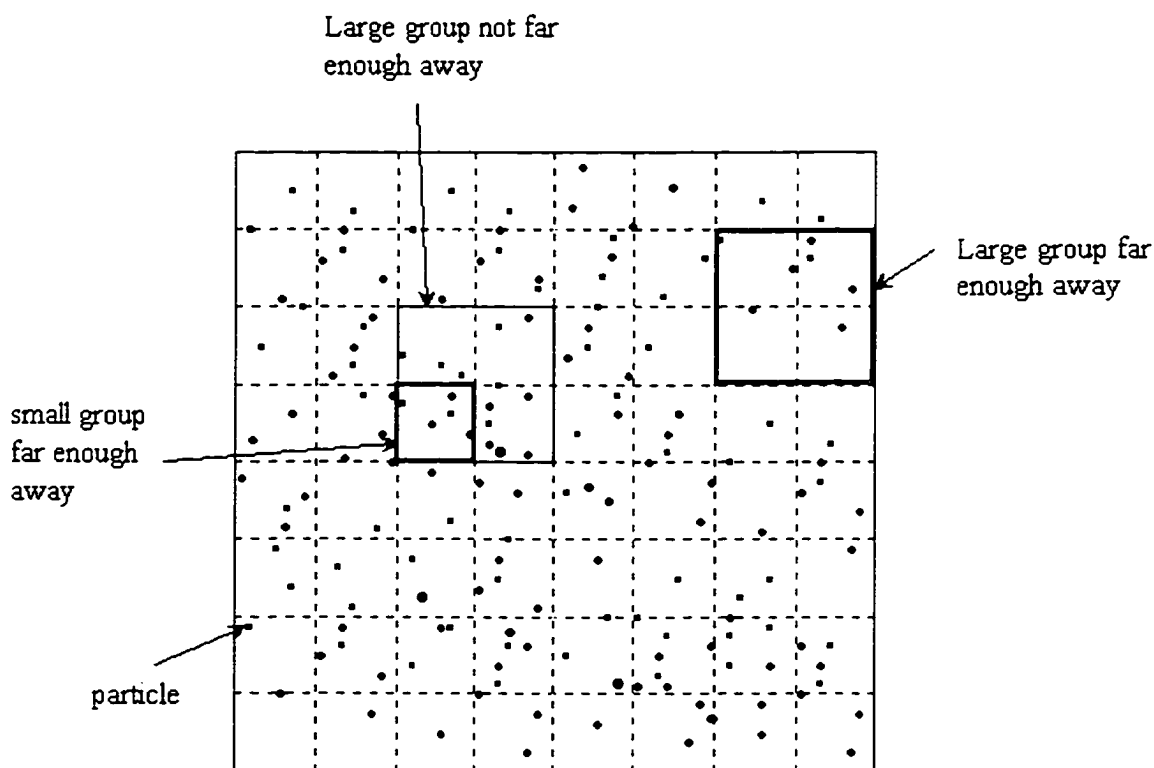


Figure 1.2. Relative Size and Distance Between Groups Dictate Approximation Rules.

A number of hierarchical algorithms have been developed in the last two decades to improve the complexity of the computation. These algorithms are based on the fact that many physical systems exhibit a large range of scales in their information requirements, in both space and time. A point in the physical domain requires progressively less information from parts of the domain further away from it. Hierarchical algorithms exploit the range of spatial scales to propagate information through the domain. Hierarchical multipole methods have become prevalent in molecular dynamics [6, 10, 37] and gravitational physics [18, 28] and have been introduced into the fields of capacitance calculation [29, 30, 31, 32], computational fluid dynamics [9, 35, 40], and electromagnetics [12]. The methods use multipole expansions which define the

effect of bodies within a region on points sufficiently well separated from the region. The expansions are exact with infinite series for computations, accuracy is selected based on the number of terms retained in the expansions. The direct algorithm is exact for a moment in time, but is actually an approximation over time as discrete time steps are used. The hierarchical algorithms are approximations of the direct algorithm which in itself is exact for given particle positions, but inexact when used with the discrete time steps.

Several algorithms have been developed based on multipole methods: the Fast Multipole Method (FMM) [14, 15], the Adaptive Fast Multipole Method (AFMM) [14], the Barnes and Hut algorithm (BHA) [3, 4], the Cell Multipole Method (CMM) [10], the Preconditioned GMRES algorithm with Adaptive Multipole Acceleration (PAMA) [30], and the Parallel Multipole Tree Algorithm (PMTA) [5, 7]. All of the methods utilize multipole expansions to describe the effect of bodies (i.e. particles, astrophysical bodies, etc.) within a sphere on points distant from the sphere, where the influence diminishes as a function of distance. A hierarchical structure groups bodies together based on proximity to allow definition of multipole expansions for each group. The multipole expansions are then used to compute the effect of the bodies in a group on distant bodies.

1.2. Objective of Research

The FMM reduces the time complexity for the simulation of N-body systems to $O(N)$. However, the FMM is not well suited for applications with non-point sources. Also, the

accuracy of the FMM is dependent on the shape, size, and separation of the boxes in its data structure. The FMM provides no flexibility in the above parameters.

A new approach for computer simulations of N-body systems is proposed in this research. The new algorithm is called the Cluster Multipole Algorithm (CMA). The goals of the new algorithm are to improve the applicability to non-point sources and to provide more control on the accuracy over current algorithms. The algorithm is targeted to applications which do not require rebuilding the data structure about the system every time step due to current limitations in the construction of the data structure. Therefore, the systems under study must have minimal changes in position over time. Examples of slowly changing systems can be found in capacitance and computational fluid dynamics simulations, as well as some molecular dynamics simulations. As the data structure development is improved, the new algorithm will be applicable to a wider range of applications. The 2-dimensional version of the algorithm is considered for easier illustration and simpler implementation to study its characteristics. However, the algorithm is fully applicable to 3-dimensional applications.

The CMA combines the clustering process of Appel [1] and the multipole method of Greengard and Rokhlin [14]. The CMA exhibits the flexibility of both the Appel and Greengard methods without sacrificing the order of computation ($O(N)$) for “well structured” clusters (found in the Greengard method and the Appel method for well formed particle distributions [1, 15]). The CMA provides more control on the accuracy of computations as compared to both the FMM and Appel’s algorithm. The accuracy of the

CMA can be controlled by two independent parameters, the accuracy measure δ , defining the degree of separation of clusters, and the number of multipole terms, p .

The benefits of the CMA over the other two algorithms are presented. The CMA also provides insight into Greengard and Rokhlin's algorithm, actually providing a performance improvement to the existing algorithm. To prove the usefulness of the CMA, the algorithm has been implemented and the results are compared with the FMM results.

The FMM is well suited to uniform distributions in square regions, but is not well suited to non-uniform distributions or irregular shapes. The Adaptive FMM, developed by Greengard and Rokhlin, targets non-uniform distributions [14, 15]. In case of non-uniform systems, failure to maintain the locality of data creates the potential of increased communication overheads in parallel implementations and degrades the running time of the problem. An approach is presented to find alternative grid structures for the FMM which maintain the locality of data within the structure to reduce communication. Once such a grid is found, the standard FMM can be applied to solve a class of non-uniform N-body problems. Also, a number of well-defined geometrical shapes, such as triangles, squares, and hexagons, are analyzed for their usefulness to the FMM or a similar hierarchical algorithm.

1.3. Outline of Thesis

In Chapter 2, several widely known N-body algorithms are described as a basis of this research. A complete description of the Cluster Multipole Algorithm (CMA) is given in Chapter 3. The algorithm and data structure time complexity is also derived. Chapter 4 provides a detailed comparison between the CMA and the FMM using simulation results, followed by the concluding remarks in Chapter 5. A detailed analysis of alternative grid structures for the FMM is given in the Appendix A.

CHAPTER 2

N-BODY ALGORITHMS

2.1. Background

The study of physical systems by particle simulation is called the “*many-body*” or the “*N-body*” problem. Several algorithms for N-body problems are discussed in this Chapter. The first is the direct algorithm which computes the interaction between all particle pairs directly. The other (hierarchical) algorithms are approximations of the direct algorithm: Appel’s algorithm [1], the Fast Multipole Method [14], Esselink’s algorithm [13], and Barnes-Hut algorithm [4]. The direct algorithm is exact for a moment in time, but is actually an approximation over time as discrete time steps are used. The hierarchical algorithms are approximation of the direct algorithm which in itself is exact for given particle positions, but inexact when used with the discrete time steps (Section 2.2.1).

2.2. Computational Structure

Increasingly popular hierarchical algorithms are based on the following fundamental insight into the physics of many natural phenomena: Many physical systems exhibit a large range of scales in their information requirements, both in space and time. A point in the physical domain requires progressively less information from parts of the domain that are further away from it. Hierarchical algorithms exploit the range of spatial scales to efficiently propagate global information through the domain. Prominent among these

algorithms are N-body methods, multigrid methods, domain decomposition methods, multi-level preconditioners, and adaptive mesh-refinement algorithms [8].

The classical N-body method models a physical domain as a system of N discrete bodies and studies the evolution of this system under the influences exerted on each body by all other bodies. Hierarchical methods use spatial approximation where a group of particles may be approximated by a single equivalent particle if the group is far enough away from the point at which its effect is being evaluated. The farther away the particles, the larger the group that can be approximated.

A classical N-body problem models a particle as a point mass. The time period for which the physical system's evolution is studied is discretized into *time-steps*. Every time-step involves several phases of computation, such as computing forces and potentials and updating particle properties such as position. The computation phase is by far the most time-consuming in typical applications, and hierarchical methods are used to speed up this phase. Figure 2.1 depicts a pictorial view of a classical N-body problem.

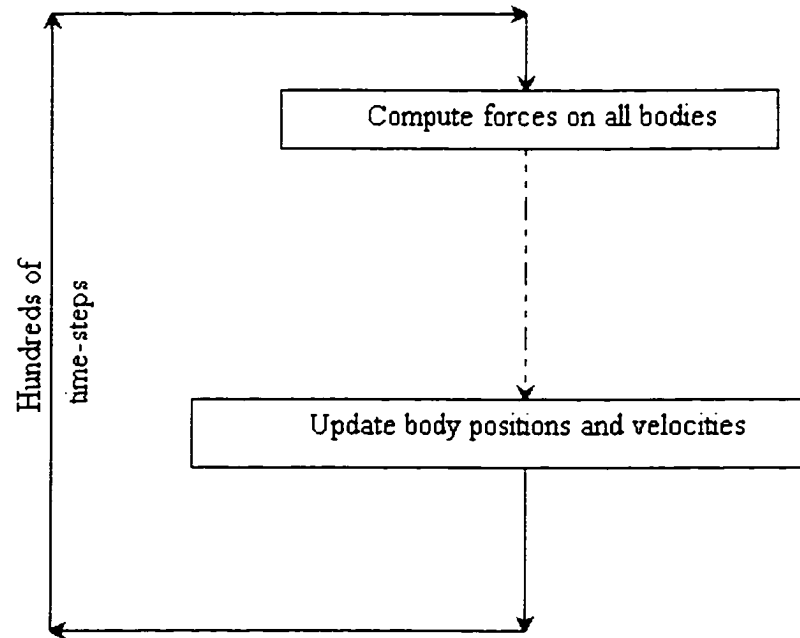


Figure 2.1. A Classical N-Body Problem.

The hierarchical methods build a tree-structured, hierarchical representation of physical space and compute interactions by traversing the tree. The tree representing physical space is the main data structure. The structure of the tree changes across time-steps, since the particle distribution and its bounding box change with time. The root of the tree represents this bounding box, which contains all the particles in the system. The tree is built by recursively subdividing space cells until some termination condition is met. This condition is usually specified as the maximum number of particles allowed per cell. Refer to [14] for a discussion of boundary conditions.

2.2.1. Discrete Time Step Error

A classical N-body problem discretizes the evolution of a physical system into hundreds or thousands of time steps. This discretization introduces approximations into any numerical calculation of the N-body problem.

Because the N-body problem cannot be done in closed form, the calculation must be done numerically. That is, at each time t , the gravitational forces of each mass on each of the others may be computed by Newton's laws. Using the inverse square force law, an approximation to the true acceleration and velocity of each particle over a time dt can be computed. By many iterations of this method, the position of each particle after an arbitrary length of time may be found.

Using a "naive" algorithm, the acceleration acting upon each particle is computed at each iteration. Using acceleration, a modified velocity over the next time increment is computed, and then the position of each particle at the end of the time increment is computed by using

$$\mathbf{r}_{\text{new}} = \mathbf{r}_{\text{old}} + \mathbf{v} dt. \quad (2.1)$$

where \mathbf{r} , \mathbf{v} , and dt are position vector, velocity vector, and time increment, respectively. The time increment dt must be made small enough that the accelerations do not greatly change between t and $t + dt$. As the time step dt is decreased, the error from the discretization is reduced. Also, the decrease in time step results in an increased computation time to simulate a fixed length of time.

2.3. Direct Computation

Direct N-body computations involve computing the force component for all body-body pairs in the system, and summing them for each body to get the total force on the body. Using electrostatics as an example and starting with Coulomb's law, the electrostatic force between two particles is

$$\vec{F} = \hat{r} \frac{q_1 q_2}{4\pi\epsilon r^2}, \quad (2.2)$$

where q_1 and q_2 are the charges on the particles and r is the distance between the particles.

For N particles, the force on a single particle j resulting from the other N-1 particles is

$$\vec{F}_j = \frac{1}{4\pi\epsilon} \sum_{i \neq j} \hat{r}_{ij} \frac{q_j q_i}{r_{ij}^2}, \quad j = 1, 2, \dots, N, \quad (2.3)$$

where r_{ij} is the distance between the particle j and i^{th} charge. The potential at the j^{th} particle can be obtained from

$$\phi_j = \frac{1}{4\pi\epsilon} \sum_{i \neq j} \frac{q_i}{r_{ij}}, \quad j = 1, 2, \dots, N. \quad (2.4)$$

The total number of interactions computed is $N(N-1)$. When computing the force, the symmetry of the force between two particles ($\vec{F}_{ij} = -\vec{F}_{ji}$) can be used to reduce the amount of computation. Thus, in order to calculate the forces acting on all N particles, $N(N-1)/2$ interactions must be computed. Figure 2.2 shows pseudo-code for the direct algorithm.

Comment: The vector $force[i]$ is the sum of all forces acting on particle i .

```

for  $i$  from 1 to  $N$  do
   $force[i] = 0$ 
   $force[j] = 0$ 
  for  $j$  from 1 to  $i-1$  do
    temp = force computed between particles  $i$  and  $j$ 
     $force[i] += temp$ 
     $force[j] -= temp$ 

```

Figure 2.2. Pseudo-Code for the Direct Algorithm.

This $O(N^2)$ process is unacceptable when looking at systems of millions of bodies, and is even difficult for thousands of bodies. Applications such as molecular dynamics require repeated evaluation of these forces for thousands of time steps. This makes the direct computation with a million particles impractical on current computer systems in a reasonable amount of time. However, this is the base case the ensuing algorithms are compared to. This method makes no assumptions about the geometry of the system. Consequently, it is easy to adapt to a variety of applications and can follow gross changes in shape and radial profile. It should be noted though that for dynamic systems where the bodies have a large degree of motion, the direct computation error over time could be large due to the discretization error. It is an exact computation given knowledge of the current positions of the bodies, but as bodies move over some time period Δt , the movement produces an error component which is a function of Δt .

One approach to improving the speed of the direct algorithm is to develop special purpose architectures. One example is the GRAPE (GRAVity PipE) processor [21]. The

GRAPE processor recognizes that the x -, y -, and z -directions can be computed in parallel. It has also been recognized, though not implemented, that several GRAPEs may be placed in parallel for greater performance. For collisionless systems, where less accuracy is required, the data length was optimized. The GRAPE designers were able to reduce most arithmetic operations to 8 bit floating point formats, though position data requires a 16 bit fixed point format and as a result, the accumulated force maintains a 48 bit fixed point format to avoid overflow [21]. The 8 bit operations are performed by using a lookup table stored in a logarithmic format. Computations are performed by passing the particle positions through a pipeline, accumulating the results for the force on a particle at the end.

2.4. Appel's Algorithm

Appel's algorithm is an attempt to improve the computational complexity of the direct algorithm by using a monopole approximation of a cluster of particles. Appel's algorithm is briefly described in this Section. A detailed version can be found in [1]. The algorithm is based on the approximation that the calculation of force between two clusters of particles can be accomplished by considering them as a point mass rather than computing all the individual forces between all their constituent particles, provided that the two clusters are sufficiently far apart.

The algorithm employs the process of forming groups (or subsets) of particles (or any other object), based on a set of predefined rules. These groups are called *clusters*.

The algorithm may adapt itself to the distribution of particles, thus providing a “natural” clustering if the clustering algorithm is optimal. When two clusters of bodies are sufficiently distant from each other, each cluster is approximated by a point mass at the center of gravity with a mass equal to the sum of masses of all particles within the cluster. This approximation is then used to compute the forces between the clusters. A predefined user controllable parameter, called δ , is an input to the algorithm. This parameter determines two clusters for which an approximation will be made. Consequently, δ controls the accuracy of the algorithm and is defined later.

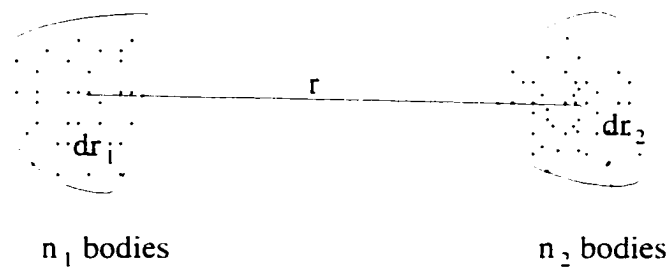


Figure 2.3. Two Clusters of Particles in Appel's Algorithm.

Consider the arrangement of masses shown in Figure 2.3, which we will assume to be a subset of the particles in a many-body simulation. To compute the interaction of each particle on every other particle, we may break the computation into three parts: those interactions between two particles in the left-hand cluster (intra-cluster interaction), those interactions between two particles in the right-hand cluster (intra-cluster interaction), and the interaction between two particles from different clusters (inter-cluster interaction).

The inter-cluster interaction may be approximated by considering each cluster a point mass, thus computing only one interaction. The number of computations required to calculate the inter-cluster interaction has thus been reduced from $(n_1 \cdot n_2)$ to $(n_1 + n_2)$, the intra-cluster calculation remains unchanged.

The selection of an appropriate data structure is important to provide easier clustering process. A binary tree is used in [1]. The binary tree associates "near" particles or clusters to form new clusters hierarchically. The leaves of a binary tree are particles and internal nodes are clusters of particles. The root of the tree contains all bodies or particles in the system. Every node has an associated mass, position, and the approximate radius of the cluster. A mass for the internal node is equal to the sum of the masses of its two child nodes and a position equal to the center of mass of its child nodes. The tree is formed such that nearby subclusters are the children of the same cluster. This is done to improve the efficiency of the algorithm.

The approximation can be applied for two clusters whose radii are small relative to their separation. If dr_1 and dr_2 are the radii and r is the distance between two clusters, then the approximation can be applied if $dr_1 / r < \delta$ and $dr_2 / r < \delta$, where δ is some fixed criteria for accuracy. If δ is set to zero, then the algorithm will recur to the individual particles (since $dr_1 = dr_2 = 0$), and no approximations will be made. This is equivalent to the direct computation. For any non-zero value of δ , the computed results are approximations to the exact results. As the value of δ increases, more approximations will be made. Consequently, the accuracy decreases with improved performance. The

parameters dr_1 and dr_2 are stored in the tree. If the accuracy criterion is not satisfied (that is, if the clusters are large and close together), then the calculation of the interaction of each of the two subclusters of one cluster with each of the two subclusters of the other cluster must be made.

```

Comment: Appel's algorithm.
Comment: ComputeAccel computes intra-cluster interactions.
procedure ComputeAccel(B)
  if B is not a leaf
  then ComputeAccel(Bleft-child)
        ComputeAccel(Bright-child)
        TwoCluster(Bleft-child, Bright-child)
Comment: TwoCluster computes inter-cluster interactions.
procedure TwoCluster(A, B)
  d = rB - rA.
  if (drA / d > δ) and (drA > drB)
  then TwoCluster(Aleft-child, B)
        TwoCluster(Aright-child, B)
  else if drB / d > δ
  then TwoCluster(A, Bleft-child)
        TwoCluster(A, Bright-child)
  else compute AccelA
        compute AccelB

```

Figure 2.4. Pseudo-Code for the Appel's Algorithm.

The algorithm is composed of two recursive procedures, namely `ComputeAccel` and `TwoCluster`, which compute intra-cluster (local) interactions and inter-cluster interactions respectively. Figure 2.4 shows pseudo-code for the algorithm. The algorithm begins at the root of the binary tree and traverses the tree recursively to look for the clusters for which the approximation can be applied. If the approximation cannot be applied, then the calculation of the interaction of each of the two subclusters of one

cluster with each of the two subclusters of the other cluster must be made (inter-cluster interactions). However, in some cases the interaction of one cluster with the two subclusters of the other cluster can be made. The leaves of the tree (particles) always meet the accuracy criterion.

The average height of a binary tree is $\log N$. For each of N particles, an $O(\log N)$ operation is required. Therefore, the computation of forces on all particles require $O(N \log N)$ time. However, Esselink showed that Appel's algorithm is $O(N)$ when used with well formed structures, as described in Section 2.6.

It is hard to analyze errors introduced by Appel's algorithm because of the arbitrary structure of the tree. It is unclear how to estimate the errors caused by the process of approximating clusters of particles together as single pseudo-particles, because clusters can take more or less arbitrary shapes and sizes. However, the error is programmable by adjusting δ .

2.5. Fast Multipole Method (FMM)

The Fast Multipole Method (FMM) addresses the $O(N \log N)$ time complexity of Appel's algorithm by using multipole approximations of groups of particles. The FMM also provides an improved error analysis as compared to Appel's algorithm. An overview of the FMM is presented in this Section. The FMM uses a recursive decomposition of the computational domain into a tree structure (a quad tree for 2-dimensions and an octal tree for 3-dimensions). It approximates particles in a cell of the tree by an equivalent series

expansion about a point in the cell for use when computing the effect of the particles on a distant point in space. The approximation is done by a higher-order series expansion about the geometric center of the cell. This series expansion is called the *multipole expansion*. The number of terms used in a multipole expansion determines the accuracy of the algorithm (an infinite expansion is exact). The use of the hierarchical data structure and a series of approximations provide a means to perform the N-body problem in $O(N)$ time.

The FMM determines “well-separatedness” of two cells based on their lengths and the distance between them. A cell is considered far enough away or “well-separated” from another cell b if its separation from b is greater than the length of b . Figure 2.5 shows the concept of well-separatedness in the FMM. Here, cells A and C are well-separated from each other. Cell D is well-separated from cell C.

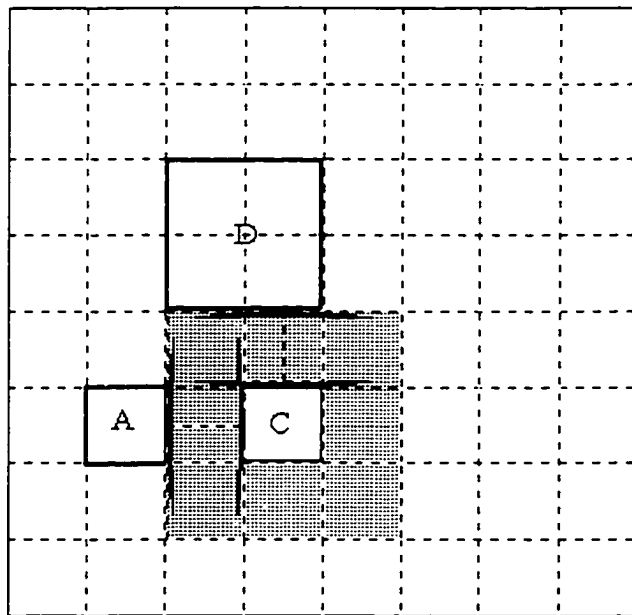


Figure 2.5. Well-Separatedness in the FMM.

2.5.1. Algorithm Description

The reader is referred to [14, 15, 16] for a complete description of the fast multipole algorithm. We begin with a computational box - a unit square in two dimensions, a unit cube in three - in which are contained all interacting particles of interest. We next define a hierarchy of grids (in two dimensions; in three dimensions we have a comparable hierarchy of cubic meshes) as in Figure 2.6. The 0th level is the original computational box itself; in two dimensions each successive grid level is obtained by dividing each cell of the previous level into four subcells (in three dimensions, each cell is divided into eight subcells). The grid is subdivided until each cell on the finest grid level has fewer than a predefined maximum number of particles per cell. The original algorithm calls for the maximum level of refinement L to be such that there is on average one particle per grid cell on the finest grid level, thus $L = \log_4 N$ for a system of N particles.

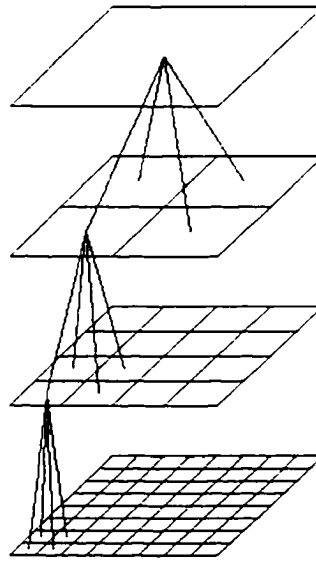


Figure 2.6. Hierarchical Grid for the FMM.

The algorithm uses two passes over the tree. The first is an upward pass which calculates a multipole expansion (ϕ) about each box's center. Each ϕ is a power series which describes the effect of the particles within a particular box on distant particles. The expansion converges for all particles outside the circle (the sphere in three dimensions) containing the particles forming the expansion. This circle is called the *circle of convergence* (see Figure 2.7). The series is exact with an infinite number of terms, and arbitrary accuracy can be obtained by truncating to p terms. The truncation to a finite number of terms in the series expansions introduces error into the calculation [11]. The further the evaluation point is from the center of the circle, the greater is the accuracy. The ϕ 's at the finest grid level are computed from the particles within the box. Each parent's ϕ is then determined by combining its children's ϕ 's walking up the tree as shown in Figure 2.6.

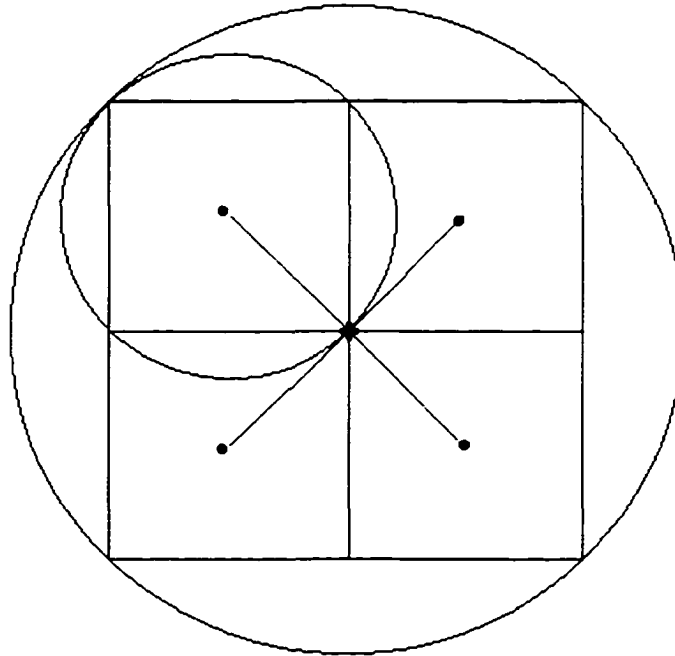


Figure 2.7. Circles of Convergence.

Once the upward pass is complete, a downward pass calculates a local expansion (ψ) about each box's center. Each ψ is a Taylor expansion, also with p terms, which describes the effect of all particles distant from the box on the particles within the box. Nearby boxes must be excluded in the computation of ψ at each level for convergence. The ψ of a box is computed by including the parent's ψ , and also including a conversion of the ϕ 's of boxes which are distant from the box of interest, but which are not included in the parent's ψ , to a local ψ term. The set of boxes whose ϕ 's are used to compute a given box's ψ is called the *interaction list*, of which an example is given in Figure 2.8 for a 2-dimensional structure. Here, the empty boxes are *near neighbors* and the light shaded boxes are the interaction list. Greengard defines the interaction list for a box i to be those boxes on the same grid level as i which are not included in the computation of i 's parent's

ψ and are also more than one box (2 boxes for three dimensions) away from i . Figure 2.9 demonstrates how the interaction lists are compounded during the tree walk to compute the ψ of a box on the finest grid level (level 4 for this example). Note that due to the interaction lists, levels 0 and 1 are irrelevant to the computation since the interaction lists of boxes on these levels are empty. It will be shown that other definitions of the interaction list are possible which improve the performance of the algorithm.

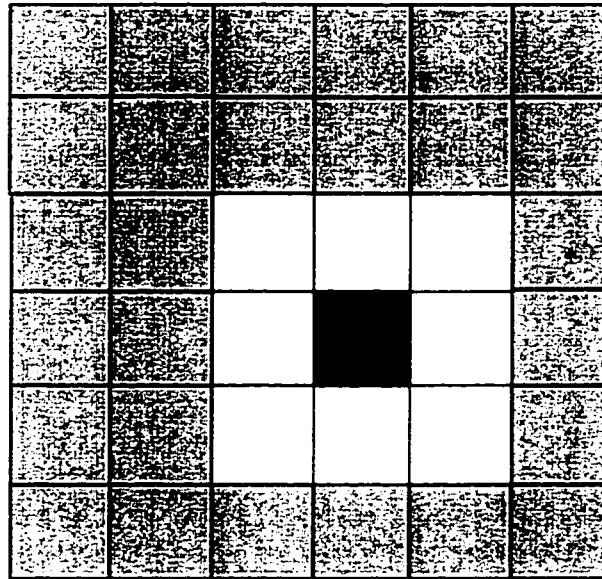


Figure 2.8. Interaction List for the FMM.

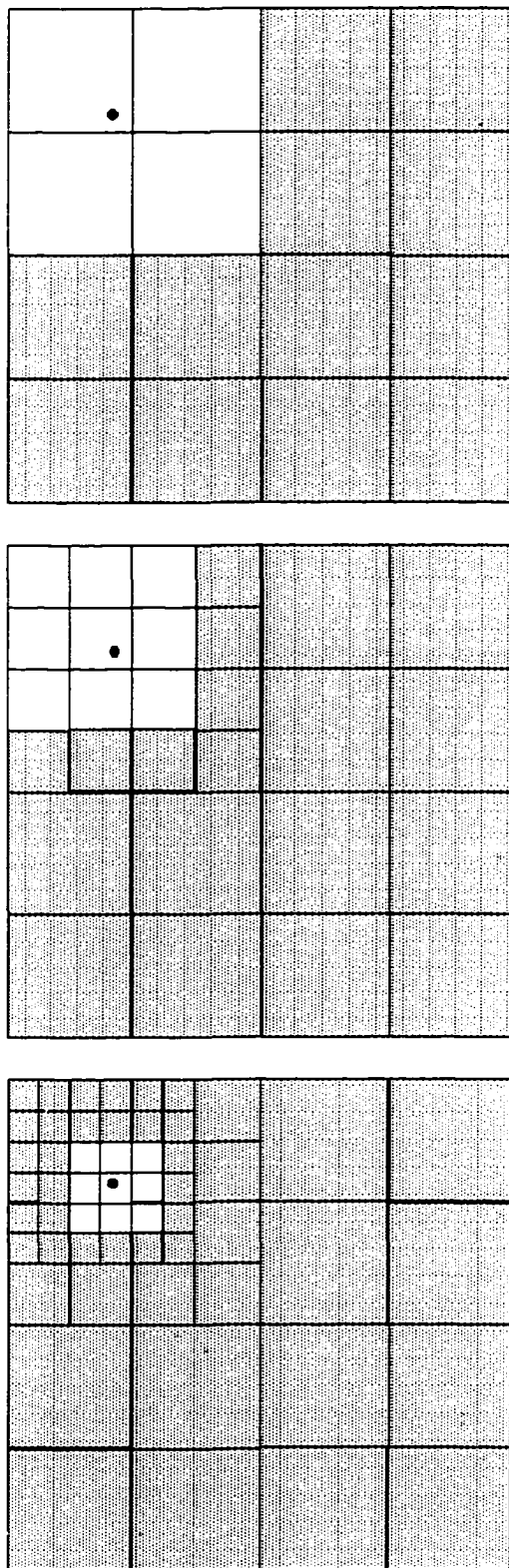


Figure 2.9. Compounding of the Interaction Lists During the Tree Walk.

When both passes over the tree are complete, the ψ 's on the finest grid level are used to calculate the potential or force due to all distant particles on the particles within each box. In turn, the effect of particles in near neighbor boxes on the finest grid level, which are excluded from the interaction list, is computed explicitly using direct computation. The maximum number of near neighbor boxes in two dimensions is 8; in three dimensions, the maximum is 24. The final result from the direct summation is summed with the distant term for the final force exerted on each particle. The efficiency of the FMM is a function of the number of particles per leaf cell in the tree and the number of terms in the expansions. Figure 2.10 shows the algorithm in pseudo-code format.

There are a total of $O(N)$ boxes in the tree, and each box requires an $O(1)$ operation for both the upward and downward passes. Therefore, the total order of computation is $O(N)$ with a constant of proportionality dependent on p , the degree of expansions, where p is chosen to achieve a given accuracy. The speed of the algorithm can be improved by reducing the size of the interaction list as discussed in the next Section.

Comment: Initialization
Choose a level of refinement $n \approx \lceil \log_4 N \rceil$, a precision ϵ , and set $p = \lceil -\log(\epsilon) \rceil$.

Comment: Upward Pass
for i from 1 to 4^n do
 Form a p -term multipole expansion $\phi_{n,i}$
end for
for l from $n-1$ to 0
 for i from 1 to 4^l
 Form a p -term multipole expansion $\phi_{l,i}$ by shifting the center of each child box's expansion to the current box center and adding them together.
 end for
end for

Comment: Downward Pass
Set $\psi'_{1,1} = \psi'_{1,2} = \psi'_{1,3} = \psi'_{1,4} = (0, 0, 0, 0)$
for l from 1 to $n-1$
 for i from 1 to 4^l
 Convert the multipole expansion $\phi_{l,j}$ of each box j in the interaction list of box i to a local expansion about the center of box i (ψ_{temp}).
 $\psi_{l,i} = \psi'_{l,i} + \psi_{temp}$
 end for
 for i from 1 to 4^l
 Expand $\psi_{l,i}$ about the children's box centers to form the expansion $\psi'_{l+1,j}$ for i 's children.
 end for
end for
for i from 1 to 4^n
 Convert the multipole expansion $\phi_{n,j}$ of each box j in the interaction list of box i to a local expansion about the center of box i (ψ_{temp}).
 $\psi_{n,i} = \psi'_{n,i} + \psi_{temp}$
end for
for i from 1 to 4^n
 For each particle p_j located at the point z_j in box i , calculate $\psi_{n,i}(z_j)$.
end for
for i from 1 to 4^n
 For each particle p_j in box i , directly compute interactions with all other particles within the box and its near neighbors ($\psi_{direct}(z_j)$).
end for
for i from 1 to 4^n
 For each particle p_j in box i , $\psi_{total}(z_j) = \psi_{direct}(z_j) + \psi_{n,i}(z_j)$.
end for

Figure 2.10. Pseudo-Code for the FMM.

Figure 2.11. Parental Interaction List.

2.5.2. Interaction List Improvements

The interaction list has been studied as to methods of improvement [23, 41]. The complexity of the algorithm remains $O(N)$, but the constant of proportionality is improved. Each possible interaction list has a different degree of accuracy and speed. Consider the interaction list shown in Figure 2.8. Note that there are 20 boxes in the interaction list whose parents have all children also included in the interaction list. Instead of converting each child separately, an approximation is to convert the parent's multipole expansion to the local expansion of the box of interest (since the parent box's multipole expansion includes the multipoles of all 4 children). This creates a new interaction list, called the *parental* interaction list, as shown in Figure 2.11. The size of the interaction list is reduced from 27 to 12, and an even greater improvement is found in three dimensions where the interaction list size reduces from 875 to 189. Some accuracy is sacrificed since the circles of convergence are not as well separated as the children's circles. However, by reducing the size of the interaction lists, speed is significantly improved, allowing p to be adjusted to compensate for the introduced error. Note that the performance is definitely improved when considering computations for a given accuracy, i.e. selecting p to provide the desired accuracy [23]. The parental interaction list can be computed by using two loops, one over the boxes on the same grid level as the box of interest and one on the parent's grid level.

2.5.3. Grid Structure Requirements

For an N -body solution using the FMM which converges with $O(N)$ time complexity, the grid structure must meet certain criteria. The FMM requirements are given as follows:

1. The grid regions should be constructed so as to have an upper limit on the number of particles in each leaf box. This is necessary to maintain the $O(N)$ time complexity for the direct portion of the FMM.
2. A grid structure must exhibit an upper bound on the size of the interaction list. This is necessary to maintain $O(N)$ time complexity. This also ensures well bounded communication overhead when parallelized because the data required for computation is well bounded both in amount and proximity in the data structure.
3. Each grid region in the interaction list must be sufficiently separated from the region under consideration to allow for convergence to a solution. This requires that circles of convergence which encompass the complete area of each individual box should not overlap. The better the separation, the more accurate the approximations.
4. There must be an upper limit on the number of near neighbors for each box in the tree. This is necessary to ensure $O(N)$ time complexity for the direct portion of the FMM.

2.5.4. The Adaptive FMM

The FMM is well suited to uniform distributions in square regions, but is not well suited to non-uniform distributions or irregular shapes. To address this problem, the adaptive

FMM (AFMM) was created. A detailed description of the AFMM is given in [15]. The AFMM subdivides the grids on each level depending on the density of particles in each box (Figure 2.12). In this way, the work is concentrated in regions which require it due to a higher density of particles. In other words, we do not use the same number of levels for all parts of the computational box. Generally, this would result in a large number of empty boxes at finer levels of the procedure. To eliminate these empty boxes, some integer $s > 0$ is fixed, and at every level of refinement we subdivide only those boxes that contain more than s charges. At every level of refinement, a table of non-empty boxes is maintained, so that once an empty box is encountered, it is completely ignored by the subsequent process.

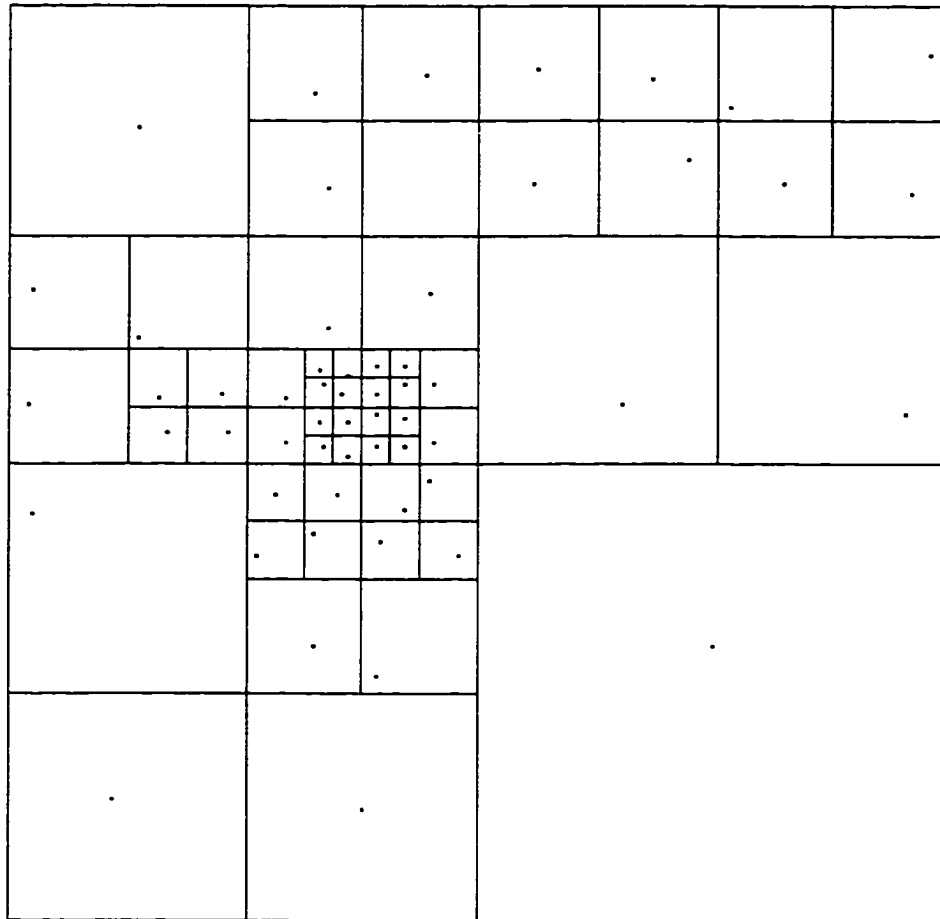


Figure 2.12. A Division of Cells for the Adaptive FMM.

The interaction list is not as simple as in the case of the uniform FMM. The interaction list now may traverse many levels in the hierarchy. We now have four lists based on where the box lies in the hierarchical grid. The management of these four lists is more complicated than the interaction list in the FMM, but at the benefit of a more flexible algorithm. The computational complexity of the AFMM remains $O(N)$ independent of the statistics of distribution.

The FMM structure parallelizes well as the load is easily balanced and all boxes in the interaction list are found a finite distance away in the structure thus reducing communication. The AFMM structure is not as well defined. It is not as easy to manage the trade-off between a balanced load and communication [25, 38].

2.6. Esselink's Algorithm

Esselink proved that Appel's algorithm is $O(N)$ when used with a *well formed* hierarchical grid tree structure such as one used in the FMM. In fact, Esselink showed that with the FMM structure, Appel's algorithm is just a subset of the FMM. Esselink's algorithm [13] uses the same procedures used by Appel's algorithm; however, it uses a hierarchical cubic grid structure instead of a binary tree. The root of the structure is a unit cube which contains the problem domain. In three dimensions, the cubic root cluster is divided into eight cubic subclusters to maintain the shape of all clusters. Each of these subclusters is further divided into eight subclusters and the process is repeated until all the clusters at the lowest level contain at most one particle. A homogeneous distribution of particles is assumed and the number of particles is a power of eight. A similar structure is used for two dimensions except a square is used, and the square is subdivided into four children instead of eight.

The algorithm begins at the root and traverses the structure recursively to look for the clusters for which the approximation can be applied. If the radius of two clusters divided by the distance does not exceed a given accuracy criterion δ , the approximation is

applied. Otherwise the two clusters are spliced into their subclusters and the algorithm is called recursively. This process is repeated until the computations for all clusters at the lowest level of the tree are completed.

Two procedures, called *ComputeAccel* and *TwoNode*, are used to perform the computation. The *TwoNode* procedure calculates the force exerted by the particles in one cluster on the particles in the other cluster and vice versa. If possible, an approximation is applied as described previously. *ComputeAccel* recursively splits a given cluster into eight clusters and calls *TwoNode* for each pair of child clusters. Esselink showed that the total number of calls to *ComputeAccel* and *TwoNode* is $O(N)$ when δ is non-zero. Therefore, the algorithm is linear in the number of particles for a homogeneous distribution. In case of a non-homogeneous distribution, it can take advantage of natural clustering. In summary, Esselink proved that certain clustering techniques, namely the hierarchical grid tree structure used in the FMM, produce $O(N)$ time when applied to Appel's algorithm.

2.7. Barnes-Hut Method

In the Barnes-Hut method [4], the force-computation phase within a time-step is expanded into three phases:

1. Building the tree: The current positions of the particles are used to determine the dimensions of the root cell of the tree. The tree is then built by adding particles one by one into initially empty root cell and subdividing a cell into its four children (eight

in three dimensions) as soon as it contains more than one particle. The result is a tree whose internal nodes are space cells and whose leaves are individual particles. Empty cells resulting from a cell subdivision are ignored. The tree extends to more levels in regions that have high particle densities, which results in an adaptive algorithm.

2. Computing cell centers of mass: An upward pass is made through the tree, starting at the leaves, to compute the centers of mass of internal cells from the centers of mass of their children. The expected computational complexity of this phase is $O(N)$ since the expected number of cells in the tree is $O(N)$.
3. Computing forces: The tree is then used to compute the forces acting on all the particles. This phase consumes well over 90% of the sequential execution time in typical problems.

Since N particles are being loaded into an initially empty tree and since the expected height of the tree when the i^{th} particle is being inserted is $\log i$, the expected computational complexity is $\sum_{i=1}^N (\log i) = N \log N$. The tree is traversed once per particle to compute the net force acting on that particle. The force-computation algorithm for a particle P starts at the root of the tree and conducts the following test recursively for every cell it visits: If the cell's center of mass is far enough away from P , the entire subtree under that cell is approximated by a single "particle" at the cell's center of mass, and the force that this center of mass exerts on the particle is computed. If the center of mass is not far enough away from the particle, the cell must be "opened" and each of its subcells visited. A cell is determined to be far enough away if the condition $l/d < \theta$ is satisfied,

where l is the length of a side of the cell, d is the distance of particle P from center of mass of the cell, and θ is a user-defined parameter used to control the accuracy of the computed forces (similar to δ for Appel's algorithm). In this way, a particle P traverses more levels of those parts of the tree which represent space that is physically close to it and groups other particles at a hierarchy of length scales. The complexity of this phase typically scales as $(1/\theta^2) \log N$ for realistic values of θ [17].

The Barnes-Hut method directly computes only particle-particle or particle-cell interactions. The accuracy and speed of the algorithm are dependent on a user-adjustable parameter θ , which corresponds to δ of Appel's algorithm. As the value of θ increases, more approximations are made, thus causing a decrease in accuracy with improved performance. The Barnes-Hut algorithm is inherently adaptive: It does not make any assumptions about the distribution of particles. The mathematics of the Barnes-Hut algorithm is the same both in two and three dimensions.

CHAPTER 3

THE CLUSTER MULTIPOLE ALGORITHM

3.1. Introduction

A new approach for computer simulations of N-body (multi-body) systems is presented. Current approaches to simulating N-body systems are based on approximating far field effects, generally by use of a hierarchical data structure and expansion series to approximate the effect of a set of bodies on a distant point in space. Consequently, these approaches are called hierarchical methods. The accuracy can be controlled by varying the number of terms in the expansion series. Generally, these methods are suitable for point sources. The goals of the new algorithm are to improve the applicability to non-point sources and to provide more control on the accuracy over current algorithms. The algorithm is targeted to applications which do not require rebuilding the data structure about the system every time step due to current limitations in the construction of the data structure (this is a topic for future research). Examples of slowly changing systems can be found in molecular dynamics, capacitance, and computational fluid dynamics simulations. As the data structure development is improved, the new algorithm will be applicable to a wider range of applications. The new algorithm, called the Cluster Multipole Algorithm (CMA), combines the clustering process of Appel and the multipole method of Greengard and Rokhlin. The CMA exhibits the flexibility of both the Appel and fast multipole methods without sacrificing the order of computation ($O(N)$) for “well

structured” clusters as found in the FMM. The accuracy of the CMA can be controlled by two independent parameters, the accuracy measure δ and the number of multipole terms, p . The CMA also provides insight into Greengard and Rokhlin's algorithm, actually providing a performance improvement to the existing algorithm.

Appel's algorithm, while not a multipole algorithm, is the hierarchical algorithm which initiated the ensuing research in multipole algorithms. It creates clusters of objects based on their size and position. Here objects may be particles, galaxies, or molecules. A monopole approximation, based on total mass and the center of gravity of a set of bodies, is applied if two clusters are small as compared to their separation. Two problems with Appel's algorithm are:

1. Its complexity has not been proven (actually, it is believed to be $O(M \log N)$ and $O(N)$ for specific cases).
2. The use of the monopole approximation limits its accuracy while maintaining performance.

Alternatively, the FMM uses a fixed predefined hierarchical grid structure over the region under consideration. The fixed structure restricts the applicability of the algorithm to non-point sources as the bodies may not be fully contained within a grid region. Also the result of the Greengard method is dependent on the shape, size, and relative separation of the underlying grid structure. This research defines the characteristics of the FMM for the definition of new structures.

The Cluster-Multipole Algorithm (CMA) eliminates the disadvantages of Appel's algorithm and the FMM by combining the benefits of both. The interaction list of the FMM is improved by incorporating a more general controlling parameter for the interaction list. This results in the improved performance for a given accuracy. The research is presented in two dimensions for simplicity and clarity, both in computation and in visual presentation. However, the CMA is easily extended to three dimensions with the same benefits. The work also uses point sources as examples to allow direct comparisons between algorithms, though it is currently best suited to non-point source applications.

The CMA attempts to provide a more general structure for non-point sources and more control on the accuracy of the computation. Examples of non-point sources are the panels used to describe the topologies of circuits in capacitance calculations [31, 29, 30, 32] or airframes in panel methods for computational fluid dynamics (CFD) [24].

First, a brief description of the problems associated with Appel's and Greengard's methods are presented to provide the reasoning for the development of the CMA. The CMA is then described as a solution to those problems. For any algorithm, the availability of an appropriate data structure is important. Consequently, the process for building a necessary data structure is described. Then, the order of the algorithm along with the amount of computation time required to create the data structure is presented.

Finally, a section describing how the CMA addresses the limitations of Appel's and Greengard's methods is presented.

3.2. Limitations of Appel's algorithm and the FMM

The algorithms described in Chapter 2 have proven valuable for point source applications. However, there is always the desire to map algorithms into alternative application areas as well as to improve the performance of the algorithms. This section demonstrates the problems with the Appel's and Greengard's algorithms which the proposed algorithm addresses.

3.2.1. Non-Point Source Applications

Appel's and Greengard's algorithms have been widely used in simulations involving point sources, such as molecular dynamics and gravitational physics. However, work has been done to use non-point sources in FMM simulations in the form of panel methods for CFD [2, 24]. There is a fundamental flaw in utilizing the FMM with non-point source applications in that a body may not be totally enclosed within the sphere of convergence about a box in the hierarchical grid.

The general approach to including panels in the FMM is to locate panels in grid boxes based on the center of mass of the panel as demonstrated in Figure 3.1. The center of mass constitutes a single point to allow inclusion in the grid structure, but this does not

ensure that the panel will be enclosed within the sphere of convergence. The effect of a panel extending beyond the sphere of convergence of the box is to actually extend the radius of the sphere of convergence. This obviously reduces the accuracy of the algorithm when applied to panel methods.

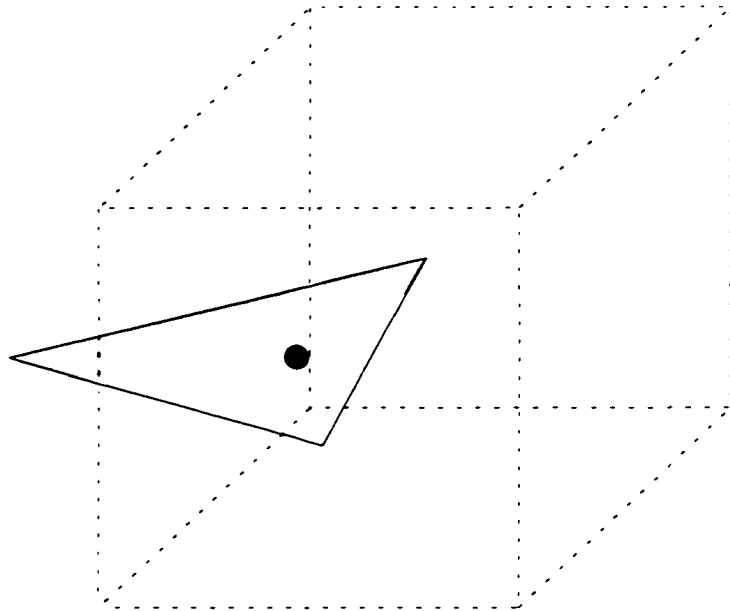


Figure 3.1. An Example of a Non-Point Source.

One solution to the problem is to keep the size of the grid boxes large enough compared to the size of the panels to reduce the probability that a panel will extend beyond the sphere of convergence. This works on the sides of the boxes, but it is still possible to extend beyond the sphere if the panel is located at the corner of the box where the sphere of convergence actually comes in contact with the box and the effectiveness of the FMM is still reduced. Therefore, it is desirable to develop an algorithm with high

accuracy, but no dependence on the rigid grid structure of the FMM. Appel's algorithm obviously has the latter characteristic, but accuracy is limited by the approximation used. A goal of the proposed algorithm is to allow greater flexibility in the formation of the sphere of convergence, while still allowing a sufficient separation between groups of bodies for approximations to ensure a high level of accuracy. This requires greater flexibility in the definition of the interaction list for the FMM and more accuracy than the monopole definition of Appel's algorithm.

3.2.1.1. 3-D Capacitance Extraction

An example of non-point sources can be found in 3-D capacitance extraction problems [29, 30, 31, 32]. In the design of high-performance integrated circuits and integrated circuit packaging, there are many cases where accurate estimates of the capacitance of complicated three-dimensional structures are important for determining final circuit speeds or functionality. Two examples of complicated three-dimensional structures for which capacitance strongly affects performance are dynamic memory cells and the chip carriers commonly used in high density packaging. In these problems, capacitance extraction is made tractable by assuming that the conductors are ideal and are embedded in a piecewise-constant dielectric medium. Then to compute the capacitances, Laplace's equation is solved numerically over the charge-free region, with the conductors providing boundary conditions.

Although there are a variety of numerical methods that can be used to solve Laplace's equation, for three-dimensional capacitance calculations the usual approach is to apply a boundary-element technique to the integral form of Laplace's equation [33, 36]. In these approaches the surfaces or edges of all the conductors are broken into small panels or tiles and it is assumed that on each panel i , a charge, q_i , is uniformly or piecewise linearly distributed. The potential on each panel is then computed by summing the contributions to the potential from all the panels using Laplace's equation Green's functions. In this way, a matrix of potential coefficients, P , relating the set of n panel potentials and the set of n panel charges is constructed. The resulting $n \times n$ system of equations must be solved to compute capacitances. Typically, Gaussian elimination or Cholesky factorization is used to solve the system of equations, in which case the number of operations is order n^3 . Clearly, this approach becomes computationally intractable if the number of panels exceeds several hundred, and this limits the size of the problem that can be analyzed to one with a few conductors.

Panel methods used for CFD computation are comparable to capacitance extraction problem. Both problems share similar issues.

3.2.2. Accuracy

The algorithms discussed in Chapter 2 are all approximations of the direct algorithm. As such, the relative accuracy of the algorithms is important. Any algorithm which can

improve the run time of a simulation for a given accuracy is a benefit, given that the accuracy is acceptable. The problems with the accuracy of Appel's algorithm and the FMM are discussed in this section.

Appel's algorithm suffers from two primary problems: the difficulty of constructing an appropriate hierarchical data structure and the limited approximation made for a group of bodies. However, Appel's algorithm is very flexible in that once a hierarchical data structure is determined, the measure to determine which groups to consider as approximations is very flexible. Therefore, the proposed algorithm attempts to maintain that flexibility.

Conversely, the FMM provides an arbitrarily accurate approximation of a group of bodies based on the number of expansion terms, but a little freedom in selecting which groups are considered for approximation. In fact, little work has been done to study the true effect of the interaction list on accuracy [23, 41]. It has just been assumed that the regularly formed interaction lists are the best available. The proposed algorithm allows the study of interaction lists in terms of the distance criterion utilized in Appel's algorithm.

The CMA attempts to take the accuracy benefits from both algorithms and combine them into a single algorithm. The algorithm should be flexible in the formation of interaction lists, and yet highly accurate in the approximation of a group of bodies. In

addition, the improvement of the algorithm's ability to handle non-point source applications makes it attractive for N-body simulation.

3.3. The Cluster-Multipole Algorithm

The inability of the FMM in handling bodies which were not points in space, such as panels, was the driving force for the development of the Cluster Multipole Algorithm (CMA). The CMA uses a more general approach to determine the interaction list. It is not mandatory to have a fixed grid structure with the constraints imposed by the FMM. In effect, a grid structure of any shape and size can be used to form "virtual" clusters. It employs the hierarchical data structure of the FMM and the criteria of Appel's algorithm to determine a set of interaction lists which are unique for each hierarchical data structure and set of bodies.

3.3.1. Algorithm

For a given distribution of particles or objects, a suitable grid (data) structure is formed with a view point of load balancing and data locality to minimize communication in a parallel implementation. Currently for use in panel methods, the data structure is created by hand with clustering based on components or functionality such as wing, cockpit, etc. for an airframe. Eventually the process will be automated, a subject for future research requiring integration with the current strategies for generating the panels.

(One possible automated process would be to still insert into the grid based on the center of mass, but to recompute the radius of the sphere of convergence based on the actual panels in a given grid, i.e. each box could have a different radius.) The radius of the circle of convergence of each box (cluster) is then computed and stored in the data structure. The formation of the data structure is crucial to maintain $O(N)$ time complexity. The data structure must be formed so as to have an upper limit on

- the size of the interaction list.
- the number of particles in each cluster.
- the number of near neighbor clusters, and
- the number of child clusters.

These bounds are necessary to allow $O(N)$ time complexity for given distributions of bodies as discussed in Chapter 2.

The next step is to compute the interaction list for each cluster in the data structure. The interaction list for cluster i is defined as a list of clusters which are contained within i 's parent, parent's near neighbors, and near neighbors themselves, and which satisfy the condition

$$\text{radius of cluster } i \text{ and } j / \text{distance between clusters } i \text{ and } j < \delta, \quad (3.1)$$

for cluster j in the interaction list, where δ is the accuracy criteria from Appel's algorithm.

The value of δ is inversely proportional to the accuracy. When $\delta = 0$, the solution is

equivalent to the direct computation. As the value of δ increases, more approximations are made. When the FMM data structure is used, $\delta = 0.3535$ is equivalent to a minimum of one box separation, and $\delta = 0.235$ corresponds to a two box separation. δ provides a high resolution in terms of the separation of boxes in the FMM. In other words, the CMA can have any amount of separation between the clusters to include them in the interaction list. This is in contrast to the FMM in which the separation can only be an integer multiple of boxes.

After the interaction lists are computed and stored, the rest of the algorithm remains the same as the FMM (the complete description is given below). The multipole expansions at the center of all clusters are computed during upward pass and the local expansions are computed in the downward pass. In the final step, the interactions of particles within near neighbors are computed using the direct method. The upper bound on the size of the interaction list is dependent on the maximum number of children (C) and maximum number of near neighbors (n). Specifically, the upper bound is given by

$$nC - (n+1). \quad (3.2)$$

The CMA applied to particle potentials can then be described by the following:

Initialization:

Choose a level of refinement $n = \lceil \log_C N \rceil$ (where C is the maximum number of children),

an accuracy criteria δ , a precision ϵ , and set $p = \lceil -\log(\epsilon) \rceil$.

Upward Pass:

Step 1: Multipole expansions at the finest level

Form multipole expansions ϕ of potential field due to particles in each cluster about the cluster center z at the finest grid level using

$$\phi(z) = Q \log(z) + \sum_{k=1}^p \frac{a_k}{z^k}, \quad (3.3)$$

where

$$Q = \sum_{i=1}^m q_i \quad \text{and} \quad a_k = \sum_{i=1}^m \frac{-q_i z_i^k}{k}. \quad (3.4)$$

Here, m is the number of charges with potentials q_i located at points z_i , $|z_i| < r$, and $|z| > r$.

The multipole expansion describes the potential field due to all particles within a particular cluster on all distant particles.

Step 2: Multipole expansions at all coarser levels

Form a multipole expansion about the center of each cluster at all coarser mesh levels, each expansion representing the potential field due to all particles contained in one cluster. This is done by shifting the center of each child cluster's expansion to the current cluster center using

$$\phi(z) = a_0 \log(z) + \sum_{l=1}^{l=p} \frac{b_l}{z^l}, \quad (3.5)$$

where

$$b_l = -\frac{a_0 z_0^l}{l} + \sum_{k=1}^{k=l} a_k z_0^{l-k} \binom{l-1}{k-1}. \quad (3.6)$$

and adding them together. Here $\binom{l}{k}$ are the binomial coefficients. z_0 is the center of a child cluster and R is the radius of a child cluster. z is any point outside the current (parent) cluster, i.e. $|z| > R + |z_0|$.

Downward Pass:

Step 3: Local expansions at all coarser levels

Form a local expansion ψ about the center of each cluster at each grid level $l \leq n-1$. The local expansion describes the field due to all particles in the system that are not contained in the current cluster or its nearest neighbors. This is done by converting the multipole expansion $\phi_{l,j}$ of each cluster j in the interaction list of cluster *ibox* to a local expansion ψ about the center of the cluster *ibox* using

$$\varphi(z) = \sum_{l=0}^p b_l z^l. \quad (3.7)$$

where

$$b_0 = a_0 \log(-z_0) + \sum_{k=1}^{k=p} \frac{a_k}{z_0^k} (-1)^k, \quad (3.8)$$

and

$$b_l = -\frac{a_0}{l z_0^l} + \frac{1}{z_0^l} \sum_{k=1}^{k=p} \frac{a_k}{z_0^k} \binom{l+k-1}{k-1} (-1)^k, \text{ for } l \geq 1. \quad (3.9)$$

Here, z_0 is the center of a cluster j in the interaction list of *ibox* and R is the radius of the cluster j . z is any point within *ibox* and $|z| < R$. The local expansions are added together, and the result is added to the initial local expansion. Once the local expansion is obtained

for a given cluster, it is shifted to the centers of the cluster's children, thus forming the initial local expansion for the clusters at the next level. The shifting is done by using

$$\sum_{k=0}^n a_k (z - z_0)^k = \sum_{l=0}^n \left(\sum_{k=l}^n a_k \binom{k}{l} (-z_0)^{k-l} \right) z^l. \quad (3.10)$$

Here, z_0 is the center of the parent cluster and z is any point in a child cluster.

Step 4: Local expansions at the finest level

Compute interactions at the finest grid level by converting the multipole expansion of each cluster j in the interaction list of cluster $ibox$ to a local expansion about the center of cluster $ibox$, adding these local expansions together, and adding the result to the initial local expansion. Local expansions at the finest grid level are now available. They can be used to generate the potential or force due to all particles outside the near neighbor clusters at the finest level.

Step 5: Local expansions at each particle

Evaluate local expansions at particle positions to obtain the potential or force due to distant particles using Equation 3.10. Here, z_0 is the center of the cluster in which a particle is located and z is the location of a particle. At the end of this step, the effect of all distant particles on each particle in the system is known. These are called *far-field* interactions.

Step 6: Potentials due to near neighbor particles

For every particle in *ibox*, compute interactions with all other particles within the cluster and its near neighbors directly. These are called *near-field* interactions.

Step 7: Total potentials

For every particle in each cluster, add direct (near-field) and far-field terms together.

Summarizing,

- Create the hierarchical data structure for the clusters.
- Create the interaction list based on the input parameter δ .
- Perform the upward pass over the tree, creating a multipole expansion about the center of each cluster.
- Perform the downward pass over the tree, creating a local expansion about the center of each cluster.
- For each body, compute the far-field force term from the corresponding local expansion.
- For each body, compute the near-field force term from the bodies in near neighbor clusters and add far-field and near-field terms together.

Thus, it is the data structure which provides the algorithmic improvement, not the tree walk. However, the upper bounds discussed earlier allow the time complexity to remain $O(N)$. The upward pass maintains $O(N)$ time complexity as a result of the bound on the number of children, while the downward pass is maintained by the bound on the

number of children and the size of the interaction lists. Finally, the last step (computing the forces from bodies in near neighbors) is maintained by the bound on the number of bodies in each cluster and the number of near neighbors. As a result, the CMA is $O(N)$ for a given distribution of bodies, independent of the number of bodies. However, if increasing the number of bodies changes the distribution, the $O(N)$ time complexity may not hold.

The accuracy of the CMA can be controlled by two independent parameters: δ and p . Each parameter can be changed independent of the other. This provides a better control to manage the trade-off between the accuracy and computation time. An increase in the value of δ causes an increase in the size of the interaction list. In other words, more approximations will be made in the computation which decreases the accuracy with increase in performance. As the value of p is increased the accuracy increases with decrease in performance. The value of δ and p can be selected appropriately to suit the application requirements.

3.4. Data Structure

An appropriate data structure is critical for an N-body algorithm. In general, the order of the algorithm is dependent on the nature of the data structure used. The data structure provides major algorithmic improvements, and is crucial to maintain $O(N)$ time

complexity. The basic requirements of the data structure used by the algorithm are described, followed by the clustering process.

3.4.1. Requirements

For an N -body solution using the CMA, which converges with $O(N)$ time complexity on a serial system, the data structure must meet certain criteria. These requirements, derived from the FMM (though not explicitly stated), are given as follows:

1. The data structure should be constructed so as to have an upper limit on the number of particles in each cluster. This is necessary to maintain the $O(N)$ time complexity for the direct portion of the CMA.
2. A structure must exhibit an upper bound on the size of the interaction list. This ensures well bounded communication overhead when the algorithm is parallelized because the data required for computation is well bounded both in amount and proximity in the data structure. This is also necessary to maintain the $O(N)$ time complexity.
3. Each region (cluster) in the interaction list must be sufficiently separated from the region under consideration to allow for convergence to a solution. This requires that circles of convergence which encompass the complete area of each individual cluster should not overlap.

4. There must be an upper limit on the number of near neighbors of each cluster in the tree. This is necessary to ensure $O(N)$ time complexity for the direct portion of the CMA.
5. There must be an upper bound on the number of children for each cluster. This ensures that the size of the interaction list is bounded.

An example of a grid which does not satisfy these requirements is shown in Figure 3.2. A detailed description on the construction of this grid structure can be found in [34].

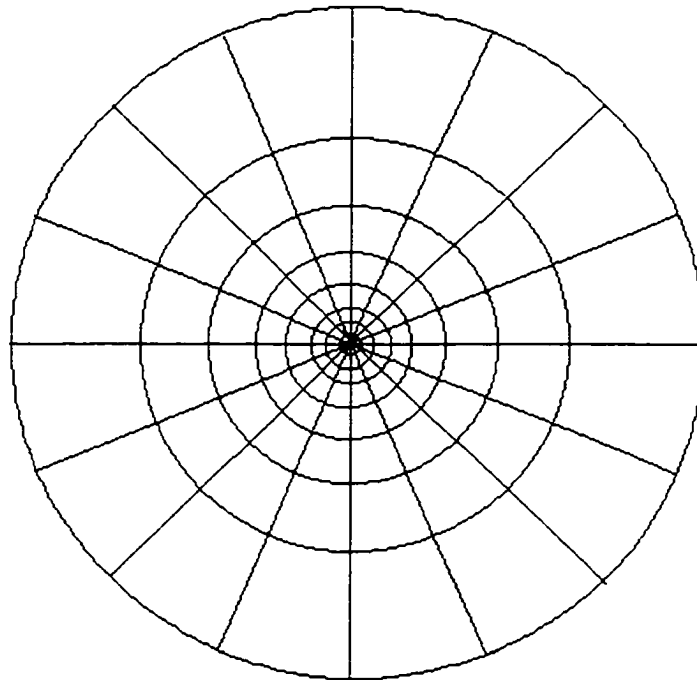


Figure 3.2. A Gaussian Grid Structure.

3.4.2. The CMA Clustering Process

The tree structure of the FMM is formed by recursively dividing a system into smaller groups of particles. Each group may contain a number of physical entities which may be in motion. Therefore, the FMM structure is rebuilt every time-step to ensure its correctness. Appel's algorithm uses a predefined accuracy criteria to form a binary tree of clusters of particles. In general, an n -ary tree structure can be formed. The CMA uses a hierarchical grid tree structure of the FMM and the criteria of Appel's algorithm to form the interaction list.

The tree structure of the CMA is formed using a process similar to that of Appel's algorithm, whereas the computation process is similar to the FMM. In the CMA, the clusters are formed as a function of the system structure and they may reflect the system's characteristics. This is important for some applications such as the molecular dynamics (MD) of solids and aircraft simulations. In the MD simulation of solids, molecules describe physical entities of the system. In the aerodynamics noise and force simulation, panels are used to describe surface parameters of an aircraft. The spatial relationship between the molecules and panels do not change with time because of the tight bonds among them. Consequently, the rebuilding of the tree on every time step is not required. This is possible because of the nature of the application domain.

The creation of clusters is currently done by hand. This process is not claimed to be $O(N)$. However, since the structure building process is not repeated every time step,

the impact on the overall order of the algorithm, if any, is minimal. To find the order of the structure building process is a candidate for future research.

3.5. Time Complexity

An analysis of the CMA time complexity is given in this section. The time complexity of the CMA can be divided into two parts: the time required to create the data structure and the time spent on actual computation. The creation of the data structure consists of creating the complete tree and computing the interaction list for each cluster. The computation of the algorithm time complexity does not take into account the computer system, language used, and implementation.

3.5.1. Data Structure

The time complexity for the creation of the complete data structure is determined in this section. It is shown that if the construction of the tree is $O(N)$, then the interaction list can be computed in $O(N)$. This makes the CMA applicable to a FMM data structure in $O(N)$ time.

Assume that we have an algorithm for constructing the tree structure which requires $O(N)$ time. The interaction lists can be calculated in $O(N)$ time using this process provided that there is an upper bound on their size. For the data structure used by the CMA, the interaction list is determined by examining parent and parent's near neighbors and their direct children. The upper bound on the size of the interaction list is

dependent on the maximum number of children and maximum number of near neighbors. If there is an upper bound on the near neighbor list (n) and the number of children (C), then the upper bound on the size of the interaction list is given by $K = nC - (n+1)$. Therefore, $O(K)$ comparisons are made for each cell, resulting in a total of $O(KN)$ or $O(N)$ comparisons for the whole structure (Here, it is assumed that there are a total of $O(N)$ boxes in the tree, a necessity to construct the tree in $O(N)$ time. This is true when structures similar to the one used in the FMM are used. If different structures are used, the total number of boxes will be approximately $O(N)$). Thus, if the structure can be built in $O(N)$ time, then the whole process is $O(N)$. At this time, the construction of the structure is assumed to be a function of the problem domain.

3.5.2. Algorithm

A brief analysis of the algorithmic time complexity for the CMA is given below:

<i>Step Number</i>	<i>Operation Count</i>	<i>Explanation</i>
Step 1	order Np	Each particle contributes to one expansion at the finest level.
Step 2	order Np^2	At the l^{th} level, Cl shifts involving p^2 work per shift must be performed, where C is the maximum number of children.
Step 3	order $\leq (K+1).Np^2$	K is the upper bound in the size of the interaction list for each cluster.
Step 4	order $\leq K Np^2$	There are at most K clusters in the interaction list and N boxes total.
Step 5	order Np	One p -term expansion is calculated for each particle.
Step 6	order $(M+1) N k_n/2$	k_n is a bound on the number of particles per cluster at the finest level and M is an upper bound on the number of near neighbors.
Step 7	order N	Adding near neighbor terms to far field terms.

Therefore, the total running time is

$$T(p, K, M, k_n, N) = N (2 a \cdot p + b p^2 + c (K+1) p^2 + d K p^2 + 0.5 e (M+1) k_n + f), \quad (3.11)$$

where a through f are constants. p is predefined for accuracy, and thus can be considered a constant.

3.6. Applicability to Non-Point Source Applications

The CMA was originally developed with the use of panel methods in mind. It was previously mentioned how the accuracy of the FMM was limited with use of panel methods. The CMA addresses those issues. The CMA provides a level of flexibility to the definition of an interaction list and near neighbors by ensuring that groups of bodies are sufficiently separated to allow accurate approximations. The removal of the reliance on a rigid grid structure allows the groupings to be made on the relationship of panels to each other instead of their relationship with a grid structure. This provides sufficient flexibility to make the CMA advantageous for use in panel methods. Current work is incorporating the CMA into a panel code. The work is based on a hand definition of the hierarchical grid structure and the grouping of the panels. Once this definition can be automated, the CMA will be a useful tool in CFD codes. Even without the mapping into panel methods, the CMA provides a benefit over the FMM, even when applied to the FMM data structure.

CHAPTER 4

SIMULATION RESULTS

4.1. Introduction

The results of the Cluster Multipole Algorithm (CMA) are compared to the Fast Multipole Method (FMM) and Appel's algorithm using the FMM data structure. The simulation results show the benefits of the CMA when applied to problems well suited to the other algorithms and even suggest an improvement to the FMM.

4.2. Background

To properly compare algorithms, the relationship of execution time versus error is considered. For a given acceptable error (selected within the algorithms by the appropriate parameters δ and p), the best execution time will be that which executes the fastest. The error (rms) equation used is

$$E = \left(\frac{\sum |R_{id} - R_{if}|^2}{\sum |R_{id}|^2} \right)^{1/2} \quad (4.1)$$

where R_{id} is the potential at the i^{th} particle obtained by the direct method and R_{if} is the result obtained by fast methods (FMM or CMA). This measure of error was originally used by Greengard and Rokhlin [14]. It is used to provide a better comparison between the CMA and the FMM results. Note that it ignores error introduced by time discretization. In all plots, the log of the error E is used for clarity.

The direct method is exact given that all particle positions are known accurately, i.e. it is exact for a single time step. The CMA and FMM are approximations to the direct method. The accuracy of the CMA can be controlled by two independent parameters: δ and p . Each parameter can be changed independent of the other. This provides a better control to manage the trade-off between the accuracy and computation time. An increase in the value of δ causes a decrease in the accuracy with increase in performance. As the value of p is increased the accuracy increases with decrease in performance.

4.3. Comparison of the CMA with the FMM

The results of the implementation of the FMM, Appel's algorithm, and the CMA using a square grid with a uniform distribution of particles are presented in this section. The purpose is to demonstrate the benefit of the CMA over the FMM for a problem domain where the FMM is well suited. As a side effect, it will be shown using the CMA that a new interaction list should be considered for use in the FMM.

The FMM is implemented in its original form (using Greengard and Rokhlin's interaction list definition). The CMA is implemented on the same hierarchical data structure for a range of values of δ and p to identify an appropriate interaction list. Appel's algorithm is implemented using the FMM data structure for a range of values of δ for $p = 0$. The algorithms are applied to a system of 10,000 uniformly distributed particles and a hierarchical data structure with six grid levels. The number of levels does affect performance and must be appropriately selected.

4.3.1. FMM Results

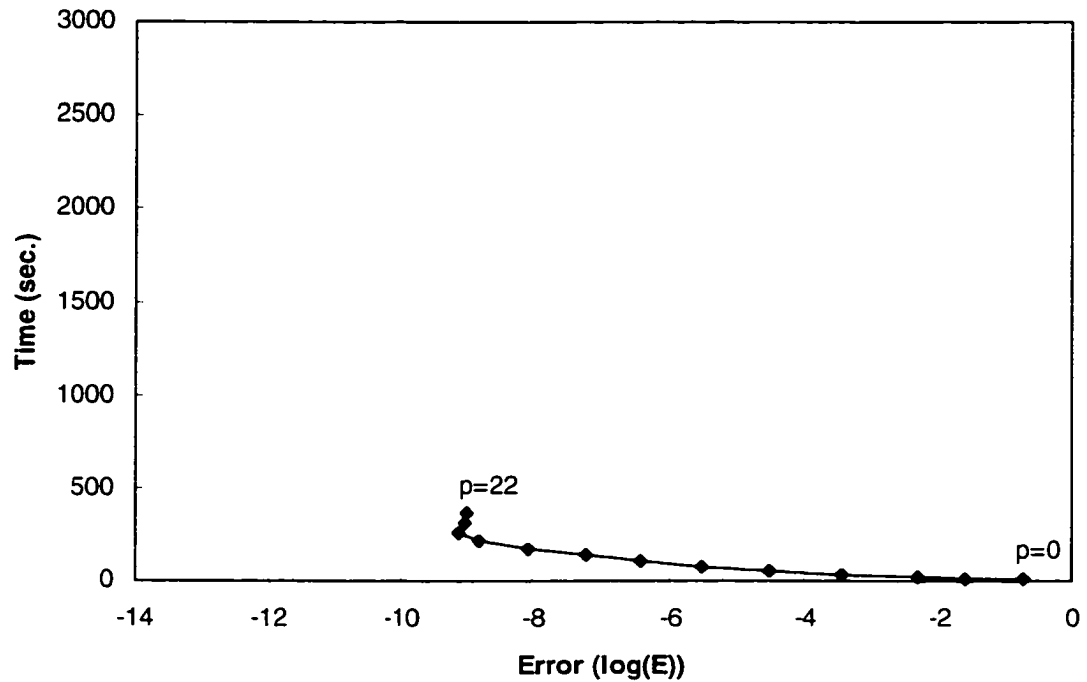


Figure 4.1. Error versus Time Plane for the Original Square Grid FMM.

The accuracy and execution time of the FMM are dependent on the number of terms (p) in the multipole and local expansions. To study the effect of p on accuracy and total time, the FMM is implemented using Greengard's interaction list (Figure 2.5). The value of p is varied from 0 to 22 in increments of 2. $p = 0$ is also considered as a special case, which is equivalent to a monopole approximation (the approximation used by Appel). A plot of the absolute error versus time for the range of values of p is shown in Figure 4.1. Obviously, the error is maximum and the execution time minimum at $p = 0$ (right end of the graph) and the execution time is maximum and the error minimum at $p = 22$ (left end

of the graph). The accuracy improves until some value of p after which it remains almost constant due to the limitations of the floating point representation, however the execution time continues to increase. For a given range of accuracy and time, an appropriate value of p can be determined using this graph.

4.3.2. Appel Algorithm's Results

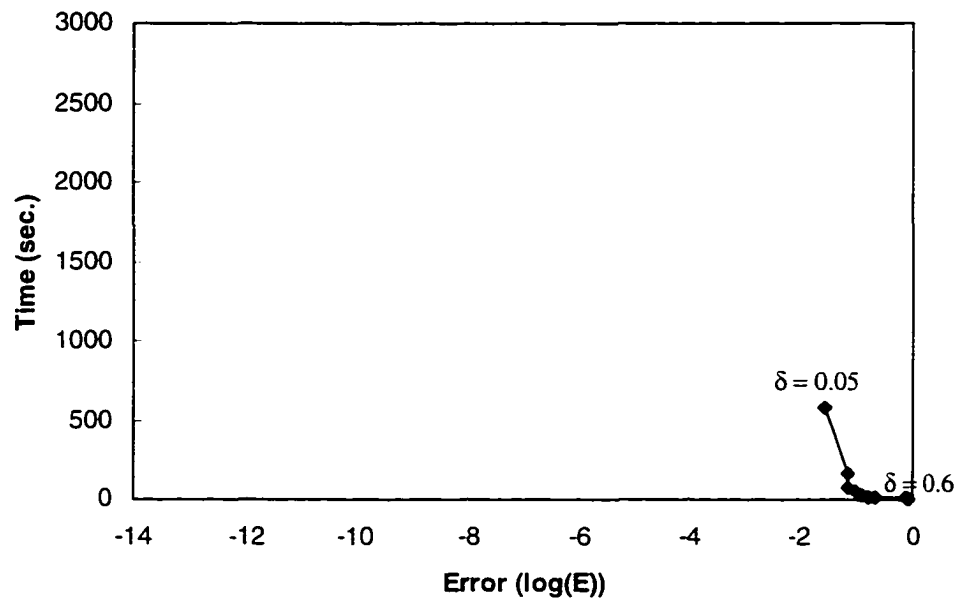


Figure 4.2. Error versus Time Plane for Appel's Algorithm for Various Values of δ .

Appel's algorithm uses monopole approximation, which is equivalent to $p = 0$ for the FMM. To study the effect of δ on accuracy and execution time, Appel's algorithm is implemented using the FMM data structure. The value of δ is varied from 0.05 to 0.60 keeping $p = 0$. A plane of the absolute error versus time for the range of values of δ is

shown in Figure 4.2. As the value of δ increases, the error increases and the execution time decreases. This is because the clusters get closer together as δ increases and more approximations are made.

4.3.3. CMA Results

The CMA is implemented using the square hierarchical grid to generate the clusters used in the algorithm. To observe the effect of δ and p on accuracy and computation time, a number of simulations are run for various sets of values of δ and p . The values of the absolute error and total execution time are computed and plotted for each set of δ and p . Figure 4.3 shows a graph for several values of δ (each line) with varying p for a uniform distribution. The value of δ increases from top to bottom of the graph and the value of p increases from right to left. δ is varied from 0.05 to 0.25 in increments of 0.05 and p is varied from 0 to 8 ($p = 0, 1, 2, 4, 6, \text{ and } 8$). As δ increases, the distance between two clusters for which an approximation is made decreases. Consequently, more approximations are made with the increase in δ , improving the performance and degrading the accuracy for a constant p .

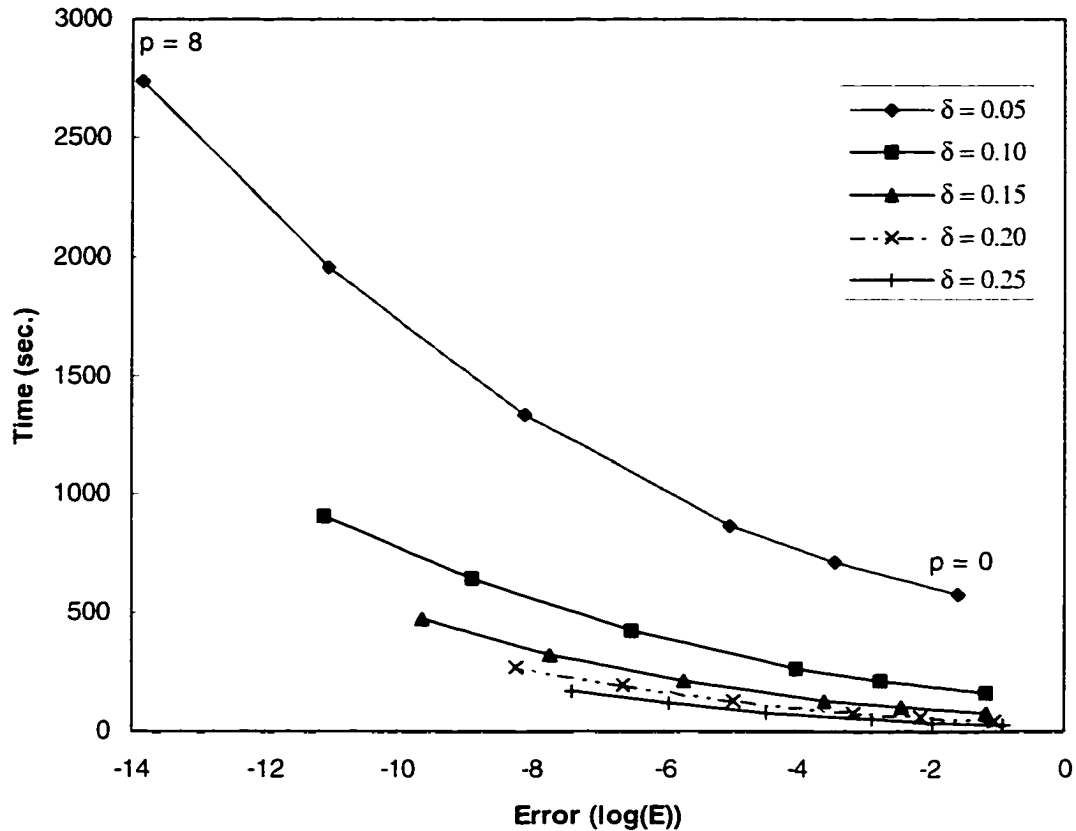


Figure 4.3. Error versus Time Plane for the CMA: Uniform Distribution.

The value of δ corresponding to a one box separation in the interaction list is equal to $\sqrt{2}/4 = 0.3535$. To observe the effect of higher values of δ , four values of δ in the range between 0.36 and 0.55 are selected and plotted as shown in Figure 4.4. The values of p are ranged from zero to ten by twos. For $\delta = 0.55$ and above, the error in computation is large, and there is essentially no improvement in the accuracy with the increase in p . This is because clusters are too close to each other to allow p to have any effect on the accuracy. $\delta = 0.44$ appears to be optimal for accuracy better than four digits

since it provides a minimum simulation time for log errors less than -4 . However, for lesser accuracy, $\delta = 0.50$ is better. It should be noted that the CMA timing results for $\delta = 0.36$ matches with the FMM (one box separation) results. This value of δ matches with the theoretical value of $\delta = 0.3535$ for the original FMM. Table 1 shows the relationship between δ and box separation.

Table 4.1. Relationship Between δ and Separation in Boxes.

Box Separation	δ	Equivalent Method
0	0	Direct
0.25	0.707	
0.5	0.471	
1	0.3535	Original FMM
1.5	0.283	
2	0.235	
3	0.177	
4	0.141	

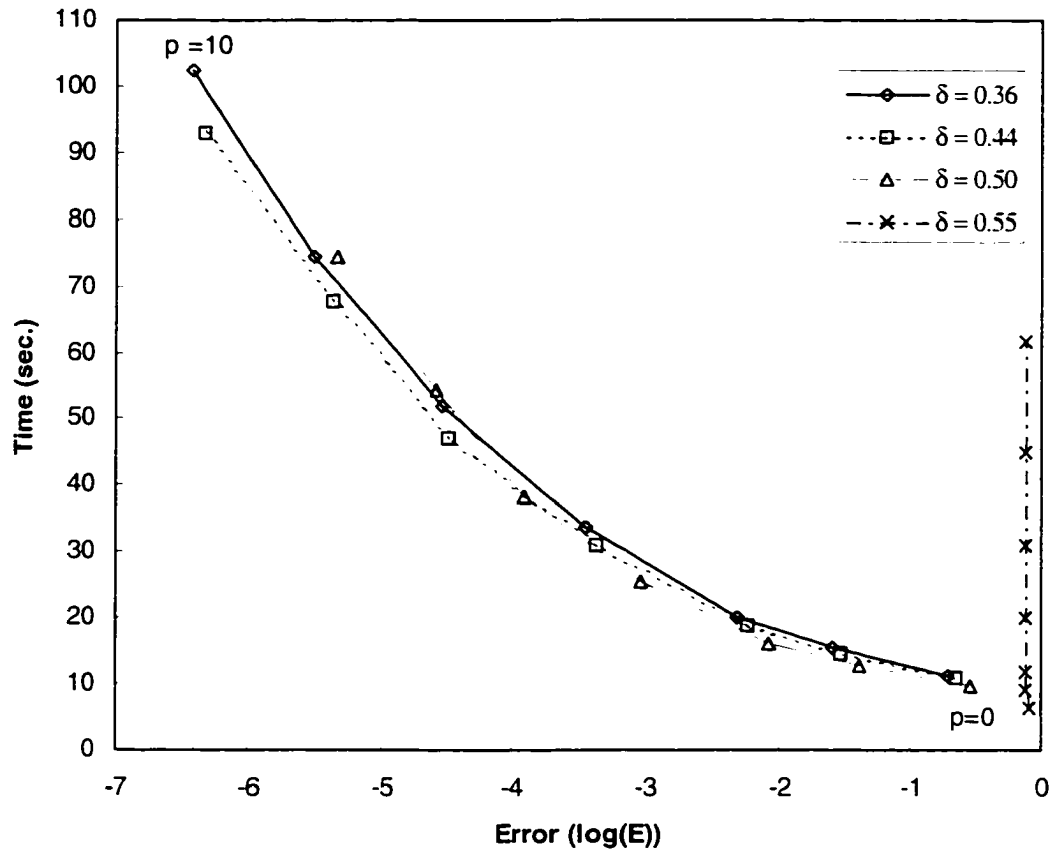


Figure 4.4. The Uniform CMA Results for Higher Values of δ .

4.4. Interaction List Analysis

The results from the CMA are used to propose a more appropriate interaction list for the FMM. The interaction list resulted from different values of δ are presented which can be incorporated into the FMM. The interaction lists for $\delta = 0.44$ and $\delta = 0.50$ are shown in Figures 4.5 and 4.6 respectively. A box with dark shading is a box of interest, boxes with light shading are in the interaction list, and boxes with no shading are near neighbors.

Large boxes in the interaction list correspond to parent boxes (as used in the parental interaction list). The size of the interaction lists are 24 and 18 respectively. The interaction list for $\delta = 0.36$ is same as the FMM interaction list. The CMA provides insight into the interaction list, demonstrating that while the parental interaction list of Figure 2.8 is better than Greengard and Rokhlin's original interaction list in Figure 2.5, a compromise is actually superior. For log errors less than -4, the interaction list in Figure 4.5 is appropriate, and for errors greater than -4, the interaction list in Figure 4.6 is appropriate. Thus the CMA proves beneficial simply in its interaction list analysis providing improved interaction lists for the FMM.

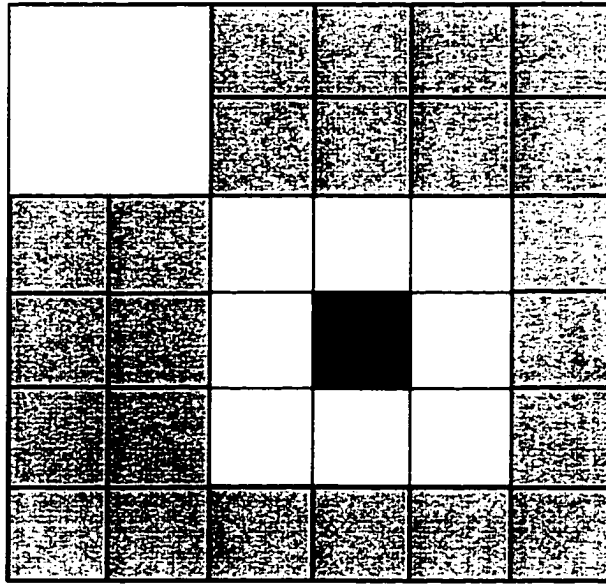


Figure 4.5. Interaction List for $\delta = 0.44$.

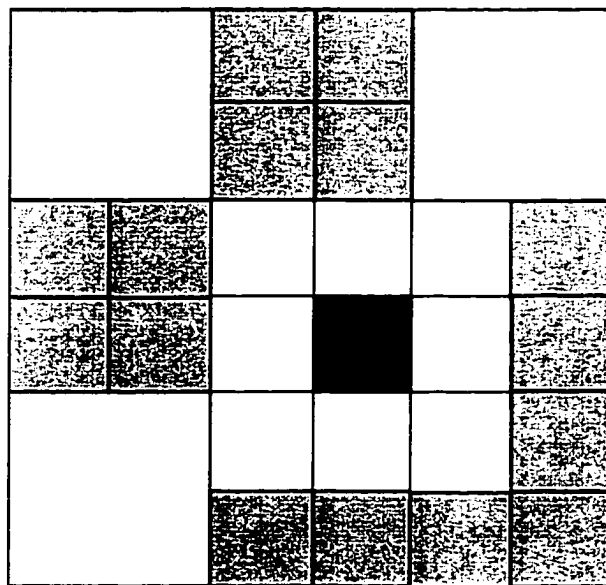


Figure 4.6. Interaction List for $\delta = 0.50$.

CHAPTER 5

CONCLUSION

5.1. Conclusions

The Cluster Multipole Algorithm (CMA) provides generalization to the Fast Multipole Method (FMM). It provides greater flexibility than the FMM and it is easily adaptable to different hierarchical structures. The CMA quantifies the restrictions on the data structures. Also, the abstract definition of the interaction list results in the improved FMM results.

The Cluster Multipole Algorithm utilizes the best features of Appel's algorithm and the Fast Multipole Method to provide an algorithm which has higher performance for a given accuracy. The CMA is well suited to non-point sources and provides more control on the accuracy over current algorithms. The CMA exhibits the flexibility of both the Appel and fast multipole methods without sacrificing the order of computation ($O(N)$) for "well structured" clusters found in the Greengard method and the Appel method for well formed particle distributions. The accuracy of the CMA can be controlled by two independent parameters, the accuracy measure δ and the number of multipole terms, p . The algorithm has been demonstrated to have higher performance than both Appel's algorithm and the FMM when applied to a problem with the hierarchical data structure used in the FMM. In fact, the CMA has demonstrated that the interaction list utilized in

the FMM is not the optimal interaction list. The CMA provides two alternative interaction lists for use depending on the desired accuracy.

The CMA also improves the ability to handle non-point source applications by removing the requirement on the rigid grid structure of the FMM and providing a better approximation of a group of bodies than Appel's algorithm. Therefore, while currently limited in the definition of the hierarchical data structure, the CMA is better suited to non-point source applications since the hierarchical data structure remains unchanged during the computation. Examples of non-point sources are the panels used to describe the topologies of circuits in capacitance calculations and airframes in panel methods for computational fluid dynamics (CFD).

5.2. Future Research

The CMA targets applications which do not require rebuilding the data structure about the system every time step due to current limitations in the construction of the data structure. This is a topic for future research. Examples of slowly changing systems can be found in molecular dynamics, capacitance, and computational fluid dynamics simulations. As the data structure development is improved, the new algorithm will be applicable to a wider range of applications.

Future research may be directed to an approach of developing a grid structure which closely approximates the distribution under study. The idea is to maintain the locality of data within the structure to reduce communication in parallel implementations. The grid is

then adjusted to satisfy the criteria placed on the grid by the FMM. These criteria include a given separation of grid regions for convergence and maintaining the $O(N)$ time complexity in serial computation. Once such a grid is found, the standard FMM can be applied to solve a class of non-uniform N -body problems.

For panel methods, δ tends to iterate over the system several times without the structure of the system changing radically. Therefore, δ could be dynamically varied at each iteration so that performance increases as δ iterates. A value of δ could be iteratively found by using sensitivity derivatives of the algorithm parameters.

Traditionally, the error analysis of N -body force computations involves only the magnitude of the force. The direction of the force along with its magnitude should be taken into account during future error analysis.

REFERENCES

- [1] A. Appel. 1985. "An Efficient Program for Many-Body Simulation." *SIAM Journal of Statistical Computing*, vol. 6, n. 1: p. 85.
- [2] D. Ashby, et. al. 1991. "Potential Flow Theory and Operation Guide for the Panel Code PMARC." *NASA Technical Memorandum 102851*, January 1991.
- [3] J. Barnes. 1990. "A Modified Tree Code: Don't Laugh: It Runs." *Journal of Computational Physics*, 87(1): 161-170.
- [4] J. Barnes and P. Hut. 1986. "A Hierarchical $O(N \log N)$ Force-Calculation Algorithm." *Nature*, 324(4): 446-449.
- [5] J. Board, W. Blanke, D. Gray, Z. Hakura, W. Elliot, and J. Leathrum. 1994. "Scalable Implementations of Multipole-Accelerated Algorithms for Molecular Dynamics." In *1994 Scalable High-Performance Computing Conference (SHPCC94)*, p. 87-94, April 1994.
- [6] J. Board, J. Causey, J. Leathrum, A. Windemuth, and K. Schultzen. 1992. "Accelerated Molecular Dynamics Simulations with the Parallel Fast Multipole Algorithm." *Chem. Phys. Let.*, vol. 198: 89-94.
- [7] J. Board, Z. Hakura, W. Elliot, and W. Rankin. 1995. "Scalable Variants of Multipole-Accelerated Algorithms for Molecular Dynamics Applications." In *Proceedings, Seventh SIAM Conference on Parallel Processing for Scientific Computing*, Feb. 1995.
- [8] T. Chan. 1990. "Hierarchical Algorithms and Architectures for Parallel Scientific Computing." In *Proceedings of ACM International Conference on Supercomputing*, May 1990.
- [9] K. Chua and T. Quackenbush. 1993. "Fast Three Dimensional Vortex Method for Unsteady Wake Calculations." *AIAA Journal*, 31(10): 1957-1958, Oct. 1993.
- [10] H. Q. Ding, N. Karasawa, and W. Goddard III. 1992. "Atomic Level Simulations on a Million Particles: The Cell Particle Method for Coulomb and London Nonbond Interactions." *The Journal of Chemical Physics*, 97(6): 4309-4315, Sept. 1992.
- [11] W. D. Elliott. 1994. "Revisiting the Fast Multipole Algorithm Error Bounds." *Technical Report 94-008*, Duke University, 1994.

- [12] N. Engheta, W. Murphy, and V. Rokhlin. 1992. "The Fast Multipole Method (FMM) for Electromagnetic Scattering Problems." *IEEE Transactions on Antennas and Propagation*, 40(6): 634-641, June 1992.
- [13] K. Esselink. 1992. "The Order of Appel's Algorithm." *Information Processing Letters*, vol. 41, p. 141.
- [14] L. Greengard. 1988. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA.
- [15] L. Greengard and V. Rokhlin. 1987. "A Fast Algorithm for Particle Simulations." *Journal of Computational Physics*, v. 73: 325-348.
- [16] L. Greengard and W. Gropp. 1990. "A parallel version of the fast multipole method." *Computers and Mathematics with Applications*, v. 20, n. 7: p. 63.
- [17] L. Hernquist. "Hierarchical N-body methods." *Comput. Phys. Comm.*, 48, 1988, p. 107-115.
- [18] L. Hernquist and N. Katz. 1989. "TREESPH: A Unification of SPH with the Hierarchical Tree Method." *The Astrophysical Journal Supplement Series*, 70: 419-446.
- [19] J. Hess and A. Smith. 1967. "Calculation of Potential Flow About Arbitrary Bodies." *Progress in Aeronautical Sciences*, vol. 8, pp. 1-138, 1967.
- [20] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. McGraw Hill, New York, 1981.
- [21] T. Ito, J. Makino, T. Ebisuzaki, and D. Sugimoto. "A special-purpose N-body machine GRAPE-1." *Computational Physics Communications*, 60:187-194, 1990.
- [22] J. Katzenelson. 1989. "Computational Structure of the N-Body Problem." *SIAM Journal of Scientific and Statistical Computing*, v. 10, n. 4.
- [23] J. Leathrum. 1992. *Parallelization of the Fast Multipole Algorithm: Algorithm and Architecture Design*. Ph.D. Thesis, Duke University, Raleigh-Durham, NC.
- [24] J. Leathrum. 1996. "Sensitivity Analysis in Multipole-Accelerated Panel Methods for Potential Flow." Submitted to *the Journal of Engineering Mathematics*, January 1996.
- [25] J. Leathrum and J. Board. 1992. "Mapping the Adaptive Fast Multipole Algorithm onto MIMD Systems." *Unstructured Scientific Computation on Scalable*

Multiprocessors. P. Mehrotra, J. Saltz, and R. Voigt, eds. Scientific and Engineering Computation Series. MIT Press, Cambridge, MA., 161-178.

[26] J. Leathrum and R. Patel, "Development of a Grid Structure for Non-Uniform N-Body Problems." *Simulation Multi-Conference: High-Performance Computing*, pp. 297-302, April 1994.

[27] A. Loeb, *Space Structures: Their Harmony and Counterpoint*, Addison-Wesley Publishing Company, Advanced Book Program, Reading, MA, pp. 125-127, 1976.

[28] J. Makino and P. Hut. 1989. "Gravitational N-body Algorithms: A Comparison Between Supercomputers and a Highly Parallel Computer." *Computer Physics Reports*, 9: 201-246.

[29] K. Nabors, S. Kim, and J. White. 1992. "Fast Capacitance Extraction of General Three-Dimensional Structures." *IEEE Transactions on Microwave Theory and Techniques*, 40(7): 1496-1506, July 1992.

[30] K. Nabors, F. Korsmeyer, F. Leighton, and J. White. 1994. "Preconditioned, Adaptive, Multipole-Accelerated Iterative Methods for Three-Dimensional First-Kind Integral Equations of Potential Theory." *SIAM Journal of Scientific Computing*, 15(3): 713-735.

[31] K. Nabors and J. White. 1991. "Fastcap: A Multipole Accelerated 3-D Capacitance Extraction Program." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11): 1447-59.

[32] K. Nabors and J. White. 1992. "Multipole-Accelerated Capacitance Extraction Algorithms for 3-D Structures with Multipole Dielectrics." *IEEE Transactions on Circuits and Systems, Part 1*, 39(11): 946-954.

[33] Z. -Q. Ning and P. M. Dewilde, "SPIDER: Capacitance Modeling for VLSI Interactions." *IEEE Transactions on Computer-Aided Design*, vol. 7, pp. 1221-1228, Dec. 1988.

[34] R. Patel and J. Leathrum. 1994. "A Gaussian Grid Structure for the Fast Multipole Algorithm." *Proceedings of the High Performance Computing Conference*, pp. 237-245, September 1994.

[35] G. J. Pringle. 1994. *Numerical Study of Three-Dimensional Flow Using Fast Parallel Particle Algorithms*. Ph.D. Thesis, Napier University, Edinburgh.

- [36] S. Rao, T. Sarkar, and R. Harrington, "The electrostatic field of conducting bodies in multiple dielectric media." *IEEE Transactions on Microwave Theory Tech.*, vol. MTT-32, pp. 1441-1448, Nov. 1984.
- [37] J. Shimada, H. Kaneko, and T. Takada. 1994. "Performance of Fast Multipole Methods for Calculating Electrostatic Interactions in Biomacromolecular Simulations." *Journal of Computational Chemistry*, 15(1): 28-43, Jan. 1994.
- [38] J. P. Singh, C. Holt, T. Totsuka, A. Gupta, and J. Hennessy, "Load Balancing and Data Locality in Adaptive Hierarchical N-Body Methods: Barnes-Hut, Fast Multipole, and Radosity." *Journal of Parallel and Distributed Computing*, 27: 118-141, 1995.
- [39] S. Song, et. al. 1992. "Computer Simulation of 3 Dimensional Stellar Systems in Cylindrical Coordinates." *Astrophysics and Space Science*, vol. 192, no. 2.
- [40] G. Winckelmans, J. Salmon, M. Warren, and A. Leonard. 1995. "The Fast Solution of Three-Dimensional Fluid Dynamical N-Body Problems Using Parallel Tree Codes: Vortex Element Method and Boundary Element Method." In *Proceedings, Seventh SIAM Conference on Parallel Processing for Scientific Computing*, Feb. 1995.
- [41] F. Zhao. 1987. "An $O(N)$ Algorithm for Three-Dimensional N-Body Simulations." *Technical Report 995*, MIT, 1987.
- [42] F. Zhao and S. Johnsson, "The Parallel Multipole Method on the Connection Machine." *SIAM Journal on Scientific and Statistical Computing*, v. 12, n. 6 (Nov.): 1420-1437, 1991.

APPENDIX A

ALTERNATIVE GRID STRUCTURES FOR THE FAST MULTIPOLE METHOD

A.1. Introduction

The original Fast Multipole Method (FMM) was developed using a square grid in two-dimensions and a cube in three-dimensions. All the particles of interest are contained within the grid. The grid structures with other regular shapes have not been considered. This Appendix focuses on the development of alternative grid structures, suitable to the FMM, with different shapes of a region under consideration. Consequently, for a given uniform N-body system with a well-defined region, the corresponding grid structure may be used. In [26] we presented an approach of developing a grid structure which closely approximates the distribution under study. Specifically, a grid structure was developed for a circular region with the gaussian distribution of particles. A similar approach is taken to form grid structures for other shapes with the uniform distribution. The objective is to develop grid structures which maintain the criteria imposed by the FMM. Triangular and hexagonal regions are considered for the grid development. Other regular shapes, such as pentagon and octagon, could not be arranged adjacent to one another to form a continuous two-dimensional structure.

An outline of the presentation is given as follows. First, a circle of convergence is defined and its effect on the truncation error is described. Second, the grid structure requirements set by the FMM are described. Third, two-dimensional square, triangular, and hexagonal structures are introduced, which is followed by a short section for the corresponding three-dimensional structures. Finally, a comparison between various two-dimensional structures is made, along with the extension to other shapes for the grid.

A.2. Circle of Convergence

The smallest circle which encloses a given box is called *the circle of convergence*. For a given number of terms (p) in the expansions, the truncation error is dependent on the distance between the center of the circle of convergence of a given box and the closest point in the circle in the interaction list. In turn, the accuracy of the algorithm is dependent on the minimum separation between convergence circles [14]. Specifically, the accuracy improves as c increases, where c is the ratio of the distance from the center of one circle to the closest point in the other circle and the radius of the circle. Refer to Figure A.1 for a pictorial interpretation of c . For a given number of terms in the multipole expansion, a higher value of c indicates a lower truncation error. It is shown in [14] that for any $p \geq \max[2, 2c/c-1]$, the error in the downward pass due to a p -term truncated series is bounded by

$$\frac{A(4e(p+c)(c+1)+c^2)}{c(c-1)}\left(\frac{1}{c}\right)^{p+1}, \quad (\text{A.1})$$

where A is defined by

$$A = \sum_{i=1}^m |q_i|, \quad (\text{A.2})$$

and e is the base of natural logarithms. Thus, the truncation error for the local expansion is also of the order of c^{-p} .

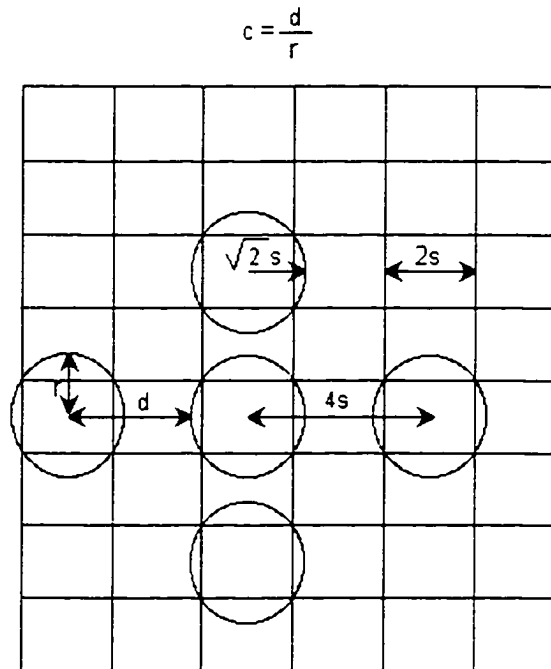


Figure A.1. The Separation Between the Circles of Convergence and the Interpretation of c .

A.3. Grid Structure Requirements

For an N -body solution using the FMM, which converges with $O(N)$ time complexity on a serial system, the grid structure must meet certain criteria. The FMM requirements are listed as follows:

1. Bounded number of particles in each box.
2. Bounded size of the interaction list.
3. Sufficient separation of the grid regions to allow for convergence.
4. Bounded parent-child relationship.
5. Bounded number of near neighbors.

These criteria are explained in regard to the $O(N)$ time complexity.

A.3.1. Bounded Number of Particles Per Box

The last step of the FMM uses the direct method in which the interactions with all particles within the box and its near neighbors are directly computed. Obviously, the computation requirement of this step is dependent on the number of particles in each box at the finest level and the number of near neighbors. Consequently, to maintain $O(N)$ time complexity for the direct portion of the FMM, the maximum number of particles in the finest grid level must not increase indefinitely with the problem size N . In particular, the maximum should be well-bounded for all N . Let k_n be an upper bound on the number of particles per box at the finest grid level. Since the interactions must be computed within the box and its eight nearest neighbors, the computation is of the order of $9Nk_n$ for all particles. However, using Newton's third law the computation can be halved. Therefore, it is clear that to maintain $O(N)$ complexity of the overall algorithm, the grid

structure must be constructed so as to have an upper bound on the number of particles per box.

In this Appendix it is assumed that only uniform particle distributions are considered. Consequently, provided all grid regions have the same area, an upper bound on the number of particles per box in the finest grid level is possible. This is the case for all grids considered in this work, thus all proposed grids will meet this criterion.

A.3.2. Bounded Size of the Interaction List

In the downward pass of the algorithm, a local expansion about the center of each box is obtained by including interactions with boxes in the interaction list. This is done by converting the multipole expansion of each box in the interaction list to a local expansion at the center of the box (Equation 3.7), and accumulating these local expansions. Clearly, the amount of work required for this step is dependent on the size of the interaction list. Specifically, the computation is bounded by $27 N_p^2$ since the maximum interaction list size is 27 boxes.

When the algorithm is parallelized, the boxes in the interaction list may be assigned to several different processors, and the multipole expansion data has to be sent to these processors in order to compute the local expansion. To reduce this communication overhead, the interaction list must be kept to a minimum. Also, the boxes in the interaction list tend to reside away from the given box as interaction list increases. Consequently, data required for computation will increase both in amount and

proximity with the increase in interaction list size, which will, in turn, cause higher communication overhead in a parallel implementation.

Therefore, to ensure a well-bounded communication overhead (in parallel version) and $O(N)$ computation time (in serial version), the interaction list size must be well-bounded and localized [16, 41].

A.3.3. Separation of Convergence Circles

For a given box, a convergence circle is the smallest circle containing the box (Figure A.1). The rate of convergence of the algorithm is dependent on the minimum separation between convergence circles [14]. As defined earlier, c is a measure of the separation between two circles of convergence. For a given acceptable expansion truncation error, a higher value of c indicates that a fewer number of terms (p) in the expansion need to be computed, reducing the p^2 term in performance. Thus, higher values of c are beneficial to the total run time of the algorithm. Geometrically speaking, c is a measure of how well a circle can enclose a box of a given shape; the better the fit, the higher the value of c . Grid structures with shapes of boxes which fit nicely into the circle should possess better convergence properties.

There exists a trade-off between the separation of convergence circles and the interaction list size. The separation can be increased by considering that all interaction list boxes be separated by at least two boxes from the box of interest. The more separated the boxes, the more accurate the expansion, which results in better accuracy for a fixed p .

However, the two box separation also has a very large number of boxes in the interaction list. Specifically, the interaction list size increases to a maximum of 75 boxes (as compared to 27 boxes for one box separation) in two dimensions. In three dimensions, the interaction list increases potentially to a maximum of 875 boxes (as compared to 189 boxes for one box separation). Any increase in the size of the interaction list slows the algorithm.

A.3.4. Bounded Parent-Child Relationship

The parent-child relationship for the grid structure must be bounded in amount and proximity. There must be an upper bound for the number of children of any parent to maintain $O(N)$ time complexity. Bounded parent-child relationship ensures that the interaction list is also bounded.

A.3.5. Bounded Number of Near Neighbors

The number of near neighbors must be bounded to ensure an upper bound on the computation in the final step of the FMM. Since the number of particles in each box and the number of near neighbors are bounded, the time spent in the direct portion (final step) of the FMM is also bounded.

A.4. Two-Dimensional Grid Structures

In this section, various two-dimensional grid structures are introduced. These structures can be used as alternatives to the square grid for the corresponding shapes of the region under consideration. For the corresponding shapes of the region, these structures eliminate empty boxes in the grid hierarchy, which would be present if the square grid were used. Also, a proper combination of these structures enables us to consider regions with other shapes.

In two dimensions, let us define the *space occupier* to be an object of any shape, the replicas of which, if arranged together, can fill (occupy) a given surface without leaving a gap (object with a different shape) in the surface. A space occupier which can fill itself is called a *self-occupier* or *regular space occupier*. The self-occupiers are best suited for the two-dimensional FMM grids, although a space occupier may be used with added complexity. Sample space occupiers are squares, triangles, and hexagons. The square and triangular grids are more suitable for the FMM than the hexagon because each shape is a self-occupier. Consequently, a hexagonal grid requires a special handling, as will be demonstrated.

First, the original square grid is described to aid in the comparison with other structures. The triangular and hexagonal grids are presented as alternatives to the square grid for the corresponding shapes of the region under consideration. A comparison

between these grid structures is made, and other variations of these grid structures are derived.

A.4.1. Square Grid

As described in the previous section, the FMM was developed using a uniform square grid in two dimensions. A portion of the square grid is shown in Figure A.1. The square is a self-occupier. If each box has sides of length $2s$, the radius of the convergence circle for each box is $\sqrt{2}s$. The distance from the center of one circle to the closest point in the circle for the closest box in the interaction list is at least $(4 - \sqrt{2})s = 2.5857s$. However, note that the closest a particle can be is at a distance $3s$, which is much greater than the distance to the circle. The circles are separated by $(4 - 2\sqrt{2})s = 1.1715s$ while the boxes are separated by $2s$. The value of $c = \frac{(4 - \sqrt{2})}{\sqrt{2}} = 1.828$, and the truncation error using p -term expansions is of the order of c^{-p} (Equations 3.4 and A.1). If the accuracy ϵ is fixed, we choose $p = \lceil -\log_c(\epsilon) \rceil$, and the interactions need to be computed only by means of expansions for clusters of particles which are contained in well-separated boxes. The square can fill its own region with squares. The notion of the square grid can be extended to the rectangular grid by combining a number of adjacent squares and forming rectangles.

Summarizing, the interaction list size is bounded by 27 (can be reduced to 12 [23, 42]), the convergence circles are separated by a minimum of $1.1715s$ for all boxes in the interaction list, the number of children of each box is bounded by four, and the number of near neighbors is bounded by eight. Therefore, the square grid satisfies all the grid requirements.

A.4.2. Triangular Grid

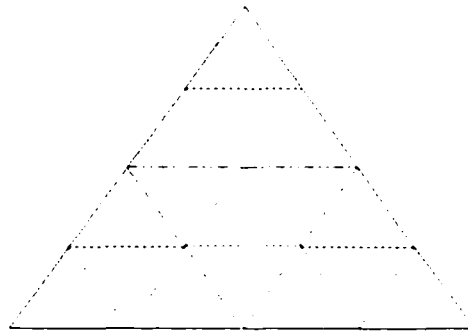


Figure A.2. A two-dimensional triangular grid. Different line patterns are used to represent the parent-child relationship.

A two-dimensional grid for a triangular region is proposed in this section. The two-dimension triangular grid is compared to the square grid with respect to the interaction list size, the number of near neighbors, the parent-child relationship, and the value of c .

If a region of interest for a given problem is triangular (assumed to be equilateral), we begin with a computational region which is an equilateral triangle centered at the origin. It contains all N particles of the system under consideration. By dividing a triangle into four equal sized triangles (by connecting mid-points of all sides) at each

level, a hierarchy of meshes (grids) similar to the one for the square grid can be formed. (Note that the triangle is a self-occupier.) Grid level $i+1$ is obtained from level i by subdividing each region (triangle) into four equal parts, as shown in Figure A.2. Here, level 0 corresponds to the triangle with solid lines, level 1 after the triangle with chain lines are added, and level 2 after the triangles with dotted lines are added. It is evident that the triangular grid has a proper parent-child relationship required by the FMM, and is bounded by four children.

For two boxes A and B to be well separated in the triangular grid it is sufficient that their boundaries do not touch with each other. Figure A.3 shows the interaction list for box b. The interaction list size is 39, though it can be reduced to 18 using techniques described in [23, 42]. The number of near neighbors is 12, which increases computation at the final step. Each box has four children as in case of the square grid. If each side of a triangle is of length s , then the radius of convergence circle is $\frac{s}{\sqrt{3}}$. The distance from the center of one circle to the closest point in the circle of the closest triangle in the interaction list is $\frac{\sqrt{7}-1}{\sqrt{3}}s = 0.95s$. Therefore, the value of $c = \sqrt{7} - 1 = 1.6457$. Note that the circles are separated by $\frac{\sqrt{7}-2}{\sqrt{3}}s = 0.3728s$. The value of c is smaller as compared to that of the square grid, which indicates that a square fits better than a triangle in the circle, and more terms in the expansions are needed to achieve a desired precision. The

triangular grid satisfies all grid requirements outlined before, and consequently, can be used as an alternative to the square grid for uniform distribution of particles within a triangular region. Also, there is an added benefit of the triangular grid. The center of one child box is the same as its parent's center. Therefore, one parent-child relationship requires no translations, and consequently, only three translations are required instead of four. This reduces the amount of computation by some extent.

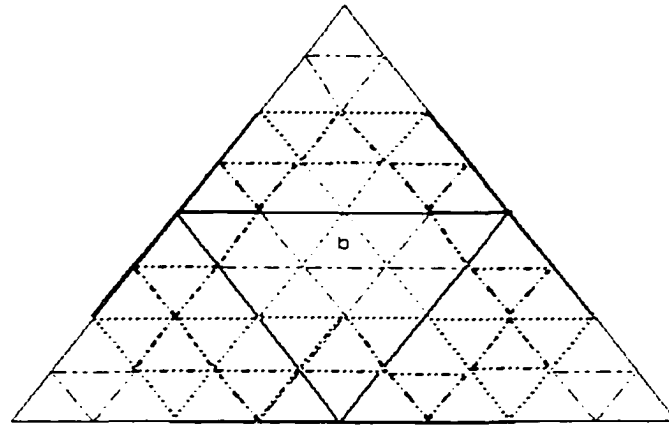


Figure A.3. The interaction list for box b is denoted by the dark boxes.

Since children are formed by connecting midpoints of parent's sides, the children will have same proportions as their parent, allowing them to be subdivided. In general, a triangle of any shape can be divided into proportionate children. Since the circle of convergence must enclose the triangle, the distance from center of mass to the farthest point gives its radius. However, as we deviate from equilateral triangles, the convergence circles become closer (circles may overlap).

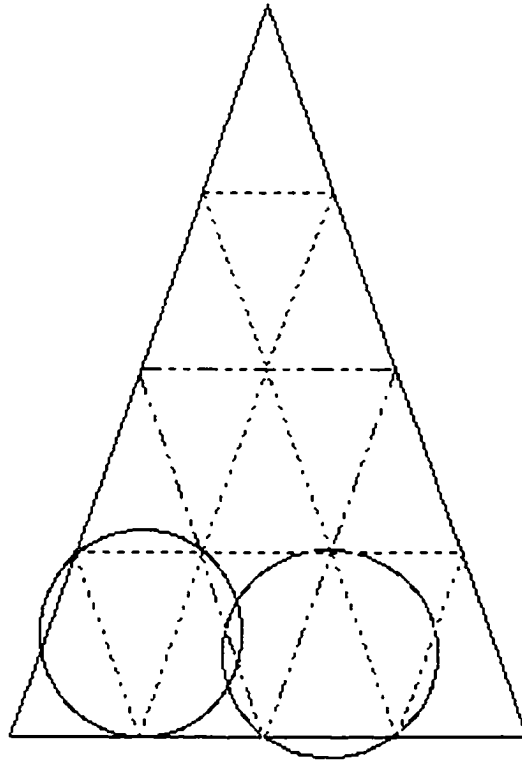


Figure A.4. A triangular grid using isosceles triangles.

As a special case, if the grid shown in Figure A.2 is horizontally contracted, it forms a grid with isosceles triangles as shown in Figure A.4. The convergence circles now overlap as shown. This grid still has a regular structure and a parent-child relationship, and may be used for the FMM. For triangular grids, equilateral triangles are most suited because the separation is better than any other triangular shapes.

A.4.3. Hexagonal Grid

Another alternative to the two-dimensional square grid for uniform distributions is the hexagonal grid. The grid structure is made up of regular hexagons, which are arranged as

shown in Figure A.5. Here, the computational region is a unit regular hexagon (thick dotted lines), which is subdivided into seven hexagons (thick solid lines) which are completely contained within their parent box. It is evident that the hexagon is not a self-occupier. The diamond-shaped spaces at six corners of the hexagon create gaps, as shown in Figure A.5. Each diamond box is really a one-third portion of the hexagon which is formed at the common point of three adjacent parent hexagons. Subsequently, six hexagons at the corners of the parent box (thick lines) are also shared by its two neighbors.

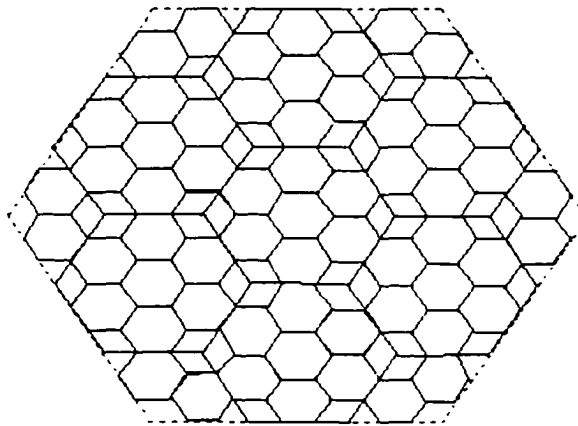


Figure A.5. A two-dimensional hexagonal grid.

Alternatively, the parent hexagon may be selected as shown by thick dotted lines in Figure A.6. However, each box now has 13 children, and the interaction list increases because the area of the parent box is much larger. Therefore, the first parent-child relationship is more suitable. Another alternative for the parent is shown by thick solid

lines in Figure A.6. This relation is better than the other, because the interaction list is smaller since each box has only three children as compared to 13.

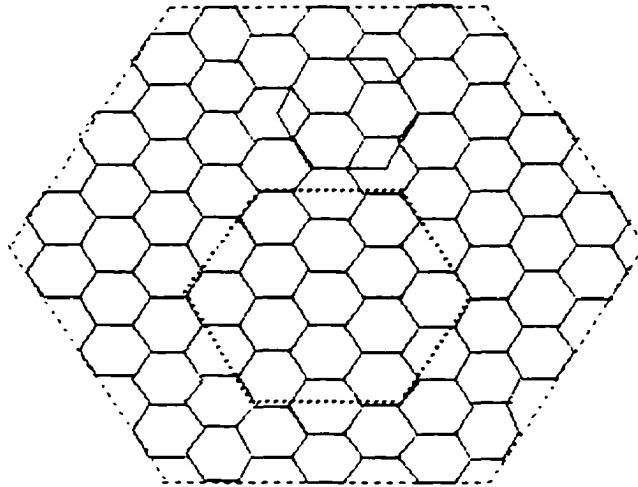


Figure A.6. Alternative parent-child relationships for the hexagonal grid.

For the parent-child relationship proposed in Figure A.5, the consideration of the diamond boxes by themselves as special cases increases the interaction list size, reduces the separation between convergence circles, and adds an extra complexity in the grid division process. One alternative to this problem is to redefine the parent-child relation as shown in Figure A.7. Here, the hexagon drawn with thick dotted lines constitutes a parent by forming a group of nine children. In effect, the two partial hexagons at top corners of the parent are also included in the parent. In the next grid level each child hexagon is divided into nine smaller hexagons with the same pattern. This creates a smaller extended hexagon which extends beyond its parent and its parent's parent. In the

subsequent levels, a chain of smaller and smaller hexagonal children is formed as shown in Figure A.8.

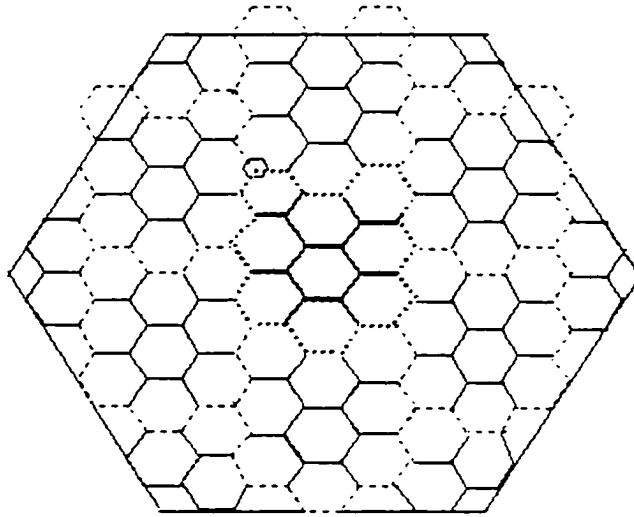


Figure A.7. A modified parent-child relationship for the hexagonal grid.

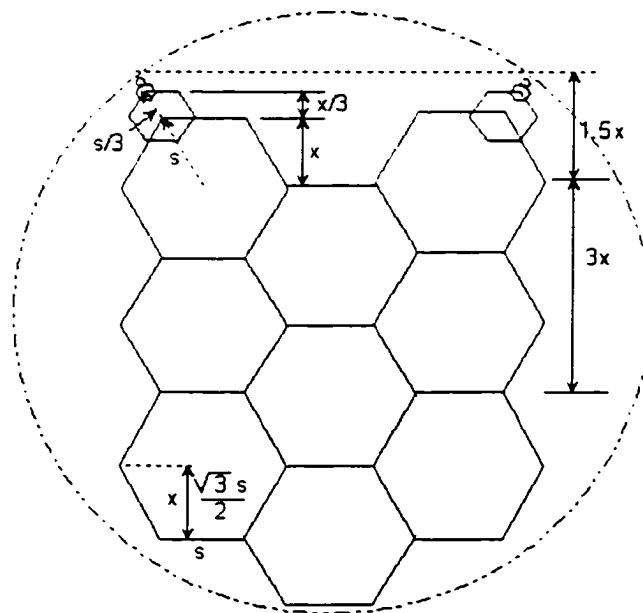


Figure A.8. The circle of convergence for the proposed parent-child relationship.

If the length of each side of a regular hexagon is s , then the upper bound on the radial length of the chain of boxes can be found using the series

$$s + \frac{s}{3} + \frac{s}{9} + \frac{s}{27} + \dots = s \sum_{i=0}^L \left(\frac{1}{3}\right)^i, \quad (\text{A.3})$$

which is bounded by $\frac{1}{1-\frac{1}{3}}s = 1.5s$. Here, L is the maximum number of grid levels.

Consequently, the upper bound on the total radial length of the box is $6s + 1.5s = 7.5s$.

After some geometrical analysis, the radius of the circle of convergence is found to be bounded by $3.6524s$. The distance between the center of one circle and the center of the closest box in the interaction list is $9s$. Therefore, the value of $c = 1.4641$. Although the value of c is less than that for the square grid, the distance between two closest circles of convergence is $1.6952s$ which is greater than the corresponding distance for the square grid. Also, the distance between circles of convergence improves at finer grid levels. At the finest grid level, where each box is exactly a hexagon, the circles of convergence are most accurate. These indicate a better fit of the hexagon into a circle, and consequently, better convergence properties of the hexagonal grid.

The interaction list is defined as a set of boxes which are well separated from the given box and which are the children of its parent and parent's neighbors. Figure A.9 shows the interaction list for a box b in the hexagonal grid. The interaction list size is 56, and the number of near neighbors are six.

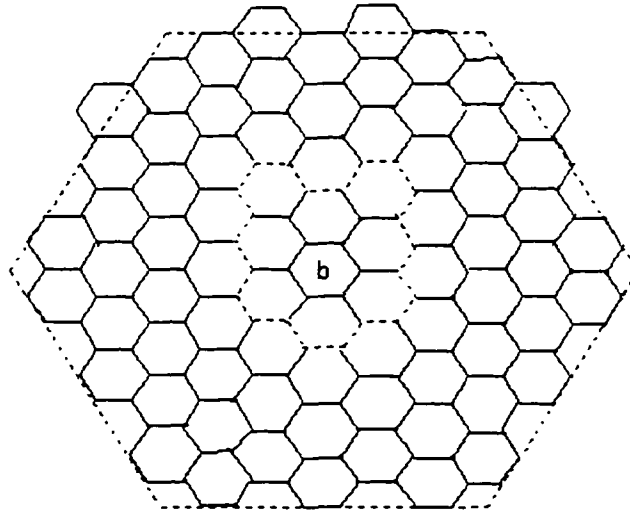


Figure A.9. The boxes drawn using thick lines are in the interaction list of the box b .

In summary, for the hexagonal grid the interaction list size is bounded by 56, the convergence circles are separated by a minimum of $1.6952s$ for all boxes in the interaction list, the number of children of each box is bounded by nine, and the number of near neighbors is bounded by six. Subsequently, the hexagonal grid satisfies all the grid requirements.

A.4.4. Comparison of Grid Structures

A comparison of various two-dimensional grids discussed previously is made in this section. The parameters of interest for comparison are the separation ratio c , the number of terms p in expansions, the order c^p of truncation error, the interaction list size, and the number of near neighbors. The values of these parameters for the triangular, square, and hexagonal grids are shown in Table A.1.

Grid	Distance between circles	c	$p = \frac{2c}{c-1}$	c^{-p}	Interaction List	Near Neighbors
Triangular	0.3728s	1.6457	5.0974	0.0789	39	12
Square	1.1715s	1.8284	4.4142	0.0696	27	8
Hexagonal	1.6952s	1.4641	6.3094	0.0902	56	6

Table A.1. A comparison between three two-dimensional grids.

In the discussion of the FMM, it was pointed out that the increase in the interaction list and near neighbors causes an increase in the amount of computation. The hexagonal grid is more accurate than other grids for the same p because of a higher separation of circles. In other words, for a given accuracy, a smaller value of p can be used with the hexagonal grid. If using small p , being able to use p less by one can overcome other deficits such as bigger size of the interaction list. Also, a smaller number of near neighbors reduces the computation in the direct portion of the FMM.

Summarizing, when selecting a grid structure for a given problem, the trade-off between the interaction list size, the number of near neighbors, and the value of p must be observed.

A.4.5. Formation of Other Grid Regions

We may combine boxes of the same shape to form a box with a different shape. For example, a rectangle can be formed by combining adjacent squares and a hexagon can be

formed using triangles. Consequently, a combination of different shapes may be used in the grid structure. In turn, the regions with other shapes can be handled by dividing them into either the square, triangular, or hexagonal structure. For example, a rectangular region can be handled by forming the square grid and a hexagonal region can be handled by forming the triangular grid within it. For a circular region, the hexagonal grid is best suited since it better approximates the circle than the square or triangle does. Similarly, for the given region a proper shape of the grid boxes may be selected.

A.5. Three-Dimensional Grid Structures

Three-dimensional grid structures are introduced in this section. Conceptually, any three-dimensional *space filler* may be used as a grid structure for the FMM. A space filler is defined as a cell whose replicas together may fill all of three-dimensional space [27]. However, the analysis of a structure becomes more complex if the structure is not a *regular space filler* (a space filler which can fill itself). The regular space fillers are best suited to three-dimensional grids. The cube is the only regular space filler [27]. In this section, the original cube structure is described first. The three-dimensional counterpart of the triangular grid (tetrahedron) is discussed next, although it is not a regular space filler. Structures with other shapes are difficult to analyze for the interaction list, the separation between convergence spheres, and the parent-child relationship because of their irregularity. Therefore, they are not considered in this Appendix. This section will

demonstrate that the cube is by far the most suitable three-dimensional grid structure for the FMM.

A.5.1. Cube

In three dimensions, the FMM computational box is a cube with sides of unit length. Each cube is divided into eight children at each level in the grid hierarchy. If the length of each side of cube is s , then the radius of the smallest sphere which encloses cube is $0.866s$. The distance from the center of one sphere to the closest point in the other sphere in the interaction list is at least $(2-0.866)s = 1.134s$. Therefore, $c = 1.134/0.866 = 1.3094$, which is smaller than its two-dimensional counterpart. In general, three-dimensional structures have less separation of the spheres of convergence, and therefore, poor convergence properties as compared to their corresponding two-dimensional structures. Also, the size of interaction list is much larger in three-dimensions (875 as compared to 27 boxes in two dimensions), the reason being poor convergence properties forcing more separation.

A.5.2. Tetrahedron

A tetrahedron may be considered as a three-dimensional counterpart of the triangular grid. It is important to note that tetrahedra are not regular space fillers. In other words, a given tetrahedron cannot be divided into a number of tetrahedra of the same shape and size. For simplicity, we will assume that all tetrahedra in the grid have equal sides. If the length of

each side of a tetrahedron is s , then the radius of the smallest sphere enclosing it is $0.5442s$. Since a cube fits better than a tetrahedron in the sphere, the separation of the spheres is less in case of the tetrahedron. To increase the separation, we need two box separation for the interaction list as used for the cube. Overlapping spheres of convergence and irregularly sized structure are major pitfalls of the tetrahedron.

A.5.3. Other Structures

Three-dimensional structures such as octahedron, dodecahedron, and icosahedron have regular polygons of a single species as their faces. However, these structures are very complex to analyze and are irregular space fillers. The visual interpretations of such grid structures are difficult to picture and understand. Also, the interaction list, the parent-child relationship, and the separation of convergence spheres are difficult to manage and calculate. The complexity of such three-dimensional structures is evident from that of the two-dimensional hexagonal structure. Therefore, these three-dimensional structures are not considered for a detailed analysis.

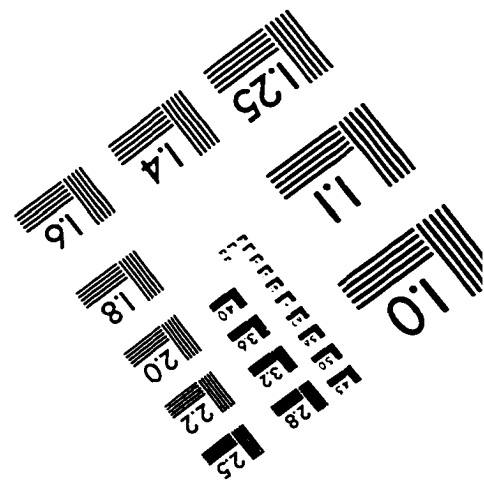
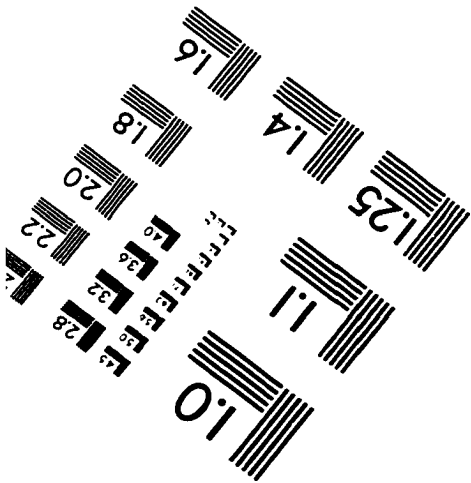
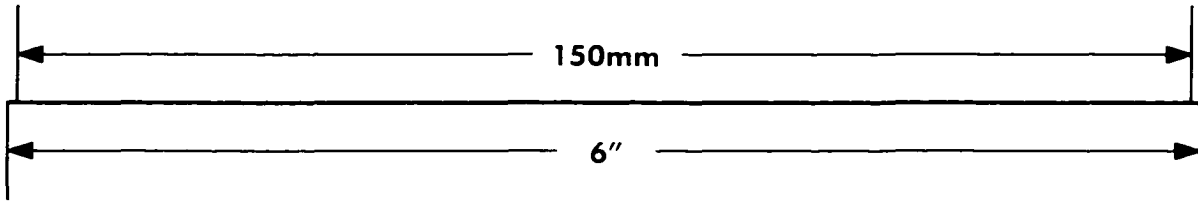
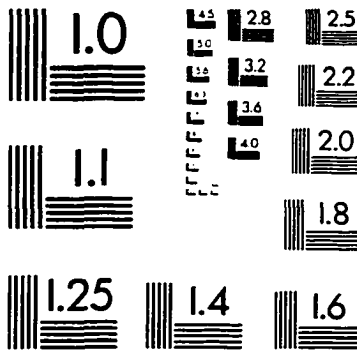
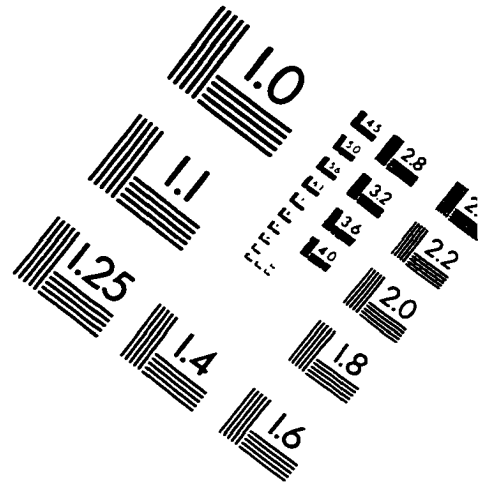
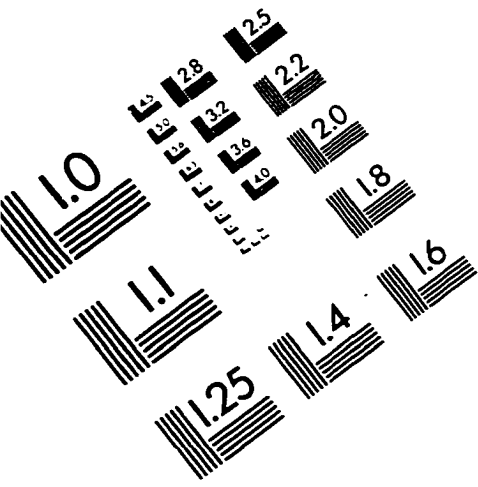
BIOGRAPHY

Rakesh R. Patel was born in Baroda, India on October 2, 1970. He achieved the first rank during the second, third, and final years of undergraduate studies and obtained his B.E. in Electronics in 1991. He was awarded a gold medal for this outstanding academic achievements. Then he came to the Unites States to attend Old Dominion University where he received his MS in Electrical Engineering in 1993. His Master's thesis was entitled "*Resource Utilization Model For The Algorithm To Architecture Mapping Model*". He has been working as a Component Design Engineer at Intel Corp. since 1995.

Mr. Patel has co-authored the following articles with Dr. James Leathrum:

1. R. Patel and J. Leathrum. 1994. "*A Gaussian Grid Structure for the Fast Multipole Algorithm.*" Proceedings of the High Performance Computing Conference, September 1994. pp. 237-245.
2. R. Patel and J. Leathrum. "*A Grid Structure for Non-uniform N-body Simulations on Parallel Computer Systems using the Fast Multipole Algorithm.*" PCAT-94.
3. R. Patel and J. Leathrum. "*A Gaussian Grid for Parallel Implementations of N-body Problems Using the Fast Multipole Algorithm.*" SC-94.
4. J. Leathrum and R. Patel. "*Development of a Grid Structure for Non-Uniform N-Body Problems.*" Simulation Multi-Conference: High-Performance Computing, April-1994. pp. 297-302.

IMAGE EVALUATION TEST TARGET (QA-3)



APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved