UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

An Implementation of the Fast Multipole Method Without Multipoles

Christopher R. Anderson

July 1990

CAM Report 90-14

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

An Implementation of the Fast Multipole Method Without Multipoles

Christopher R. Anderson

July 1990
CAM Report 90-14

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

AN IMPLEMENTATION OF THE FAST MULTIPOLE METHOD WITHOUT MULTIPOLES

CHRISTOPHER R. ANDERSON*

Abstract. We present an implementation of the fast multipole method which uses approximations based on Poisson's formula. Details for the implementation in both two and three dimensions are given. We also discuss how the multigrid aspect of the fast multipole method can be exploited to yield efficient programming procedures. The issue of the selection of appropriate refinement level for the method is addressed. Computational results are given which show the importance of good level selection. We present an efficient technique which can be used to determine an optimal level to choose for the method.

1. Introduction. The purpose of this paper is three-fold. First we will present a method for computing N-body interactions which is similar to the fast multipole method (FMM) as developed by L. Greengard and V. Rokhlin [4] [5] and L. Van Dommelen and E. Rundensteiner [12], but one which does not use complex power series in two dimensions or spherical harmonic expansions in three dimensions. Our procedure will be based on the use of Poisson's formula for representing solutions of Laplace's equation. While the accuracy and operation count of the resulting method is almost identical to the fast multipole method, the method does offer some advantages. One advantage is that the component operations of the multipole method such as shifting and combining multipoles are very easy to formulate for approximations based on Poisson's formula. Another advantage is that the difference between the two and three dimensional methods is very slight, and so programming a three dimensional method is relatively straight forward once a two dimensional method has been programmed. The second aspect of this paper is to discuss how multigrid programming strategy can be used to facilitate the programming of our method and others like it (such as the original fast multipole method). Thirdly, we wish to discuss the issue of parameter selection when using these "fast " methods. Essentially, the computational efficiency of these methods depends critically upon the choice of a level of refinement of physical space. A wrong choice can lead to a very inefficient algorithm. We shall present a procedure for obtaining an optimal choice of the refinement level.

The problem of calculating N-body interactions occurs in a wide variety of computational problems - discrete vortex calculations, galaxy simulations, plasma simulations etc. For each of these computational problems the calculation takes on a slightly different form, but each shares the common feature that the interaction is determined via solutions of Laplace's equation. So, rather than address each different application, we will discuss the following N-body model problem : Given N charged particles at locations x_i with strengths κ_i the goal is to calculate the potential $\phi(x_i)$,

* Department of Mathematics, UCLA, Los Angeles, California, 90024

Research Supported by ONR Contract #N00014-86-K-0691, NSF Grant DM586-57663 and IBM Fellowship D880908

where ϕ is a solution of

$$\Delta\phi = \sum_{i=1}^{N} \delta(x - x_i)\kappa_i. \tag{1}$$

Here $\delta(x)$ is Dirac's delta function and Δ is the Laplacian. Since the key ideas behind the method presented here don't change much when one goes from two to three dimensions, we will primarily discuss the two-dimensional case. Those aspects which do change with dimension will be specifically addressed.

Often in simulations one uses smoothed delta functions (or "blobs") and the model problem in this case is identical to (1) but we solve

$$\Delta\phi = \sum_{i=1}^{N} \Psi_\epsilon(x - x_i)\kappa_i \tag{2}$$

where Ψ_ϵ is a smoothed delta function whose support is contained within a disk of radius ϵ. The choice of Ψ and ϵ is important for a simulations accuracy, but not particularly important for the methods used to accelerate the computation of (2). We do make the assumption that the blobs have support contained within disks or spheres of radius ϵ, and so the blob functions cannot be completely general.

The solution ϕ of (1) is given by

$$\phi(x) = \sum_{i=1}^{N} \frac{\kappa_i}{2\pi} \log(|x - x_i|) \tag{3}$$

while for (2)

$$\phi(x) = \sum_{i=1}^{N} \frac{\kappa_i}{2\pi} \Phi_\epsilon(x - x_i) \tag{4}$$

where Φ_ϵ is the potential induced by a single blob. (This can often be calculated explicitly by solving $\Delta\Phi = \Psi_\epsilon(x)$ using the method of separation of variables.)

From (3) or (4) it is clear that if we evaluate the solution ϕ at each point $x_i, i = 1, \ldots \mathrm{N}$, then this computation requires $\mathrm{O(N^2)}$ operations. For particle simulations, the larger the value of N the better, and so there is great interest in reducing this operation count. The fast multipole method is a technique for reducing the operation count of this problem to O(N) operations.

There are two basic ingredients to the fast multipole technique. One ingredient is the process of combining large numbers of particles into single computational elements. When a cluster of particles is "far away" from a particular point, then the potential of the cluster is approximated by the potential induced by a single computational element located inside the cluster. (How far "far away" is must be determined of course.) In the fast multipole method the computational element is a multipole expansion located at the center of a disk containing the cluster of particles. In two dimensions one can represent the potential induced by a collection of particles as the real part of $\log(z)$ where z is a complex number. Given a cluster of particles located at positions z_i, $i = 1 \ldots \mathrm{N}_c$, the approximation used is the following

$$\phi(z) = \mathrm{Re}\,(\sum_{i=1}^{N_c} \log(z - z_i)\,) \approx \mathrm{Re}\,(\, a_0 \log(z - z_0) + \sum_{k=1}^{p} \frac{a_k}{(z - z_0)^k}\,). \tag{5}$$

Here z_0 is the center of a disk enclosing the particles z_i, p is the order of the multipole, and a_k are coefficients chosen so that the multipole is an accurate approximation of the potential. The efficiency of the method comes about because the evaluation of the potential of this single computational element, $\mathrm{O}(p)$ operations, is typically much less work than that of computing the corresponding potential of the whole collection of particles, $\mathrm{O}(\mathrm{N}_c)$ operations. In the three dimensional implementation of the method, the complex multipole is replaced by an expansion in spherical harmonics [5].

The second ingredient of the fast multipole method has to do with organizing the computations so that the application of the technique of combining particles is efficient and doesn't lead to inaccuracies. For example, when one combines particles into single elements, the more widely distributed in space the particles of a given cluster are, the greater the inaccuracy the approximation (5) becomes (for a fixed value of p and a fixed point of evaluation). However, if the particles are fixed and the evaluation position, z, is moved away from the center of the expansion z_0 then the accuracy of the potential approximation at z improves. The net effect is that if a fixed degree of accuracy is desired, one approximates the potential by using a hierarchy of approximations of the form (5). Close to an evaluation point one combines particles over small regions to form multipole approximations. Particles further away are combined over larger regions to form multipole approximations. (Essentially the size of the region over which particles are combined is inversely proportional to their distance to the evaluation point.) In the fast multipole method, this aspect is organized by decomposing the region containing all of the particles into boxes of different sizes. The particles in each of the boxes are combined and their potential is replaced by an approximation of the form (5). The potential at a particular point is then the sum of these approximations and the potential induced by the direct contribution of very close particles. In Figure 1 we show a test particle and the surrounding regions in which the particles are combined into multipole expansions. The potential induced by particles in the region immediately surrounding the test particle is computed using the exact interaction formula (3) or (4).

In the approximation of the potential due to a cluster of particles by a single computational element, one is not forced to use multipoles. There is a possibility for other types of computational elements to be used and in this paper we shall discuss one alternative computational element. The basic strategy is to use Poisson's formula. In two dimensions we have that for points outside of a disk of radius a containing the particles, the potential can be represented by

$$\phi(r,\theta) = \kappa \log(r) + \frac{1}{2\pi} \int_0^{2\pi} \tilde{\phi}(a,s) \left[\frac{1 - (\frac{a}{r})^2}{1 - 2(\frac{a}{r}) \cos((\theta - s)) + (\frac{a}{r})^2} \right] ds. \tag{6}$$

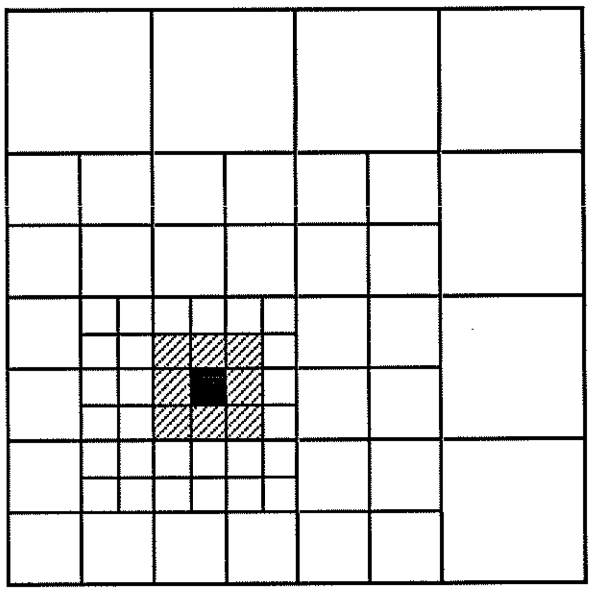


FIG. 1. *The hierarchical clustering of particles used to create a multipole approximation to the potential at a point. The evaluation point is within the darkened box. The potential induced by particles in unshaded boxes are combined into multipole approximations. The potential induced by particles in the lined boxes is computed using the direct interaction formula.*

where the function $\tilde{\phi}(a, s)$ and the value of κ are determined from the values of ϕ on the circumference of the disk. (r, θ) is the position in polar coordinates of the evaluation point from the the center of the disk. In the numerical method, the integral in Poisson's formula is converted into a sum of K quantities (where K is the number of points in an integration rule for (6)). Thus, the potential induced by a cluster of N_c particles can be reduced to the evaluation of a sum of K terms. The idea of using a numerical approximation of Poisson's formula to represent the potential is similar in spirit to an aspect of the method introduced by Rokhlin [9] for solving the equations of scattering theory. While the idea of using Poisson's formula is straight forward, getting Poisson's formula to work numerically proved quite troublesome. The problem is that the numerical evaluation of Poisson's kernel is difficult because of the singularity in the kernel. As the evaluation point approaches the ring where the integration is performed, the kernel becomes more and more singular, and the accuracy deteriorates rapidly. In the first section we discuss how this problem can be eliminated. In the first section we also present the necessary details (for both the two and three dimensions) which facilitate the incorporation of this type of computational element into a fast multipole scheme. In order to distinguish this type of computational element based on Poisson's formula from a multipole element, we refer to them as "outer ring approximations" (two dimensions) or "outer sphere approximations" (three dimensions). (We shall also have use for these approximations inside rings or spheres and these will be "inner ring" and "inner sphere" approximations.) In Figure 2, a schematic of the use of this element is presented. The computational particles (represented by the small circles) induce a potential at locations on the circumference

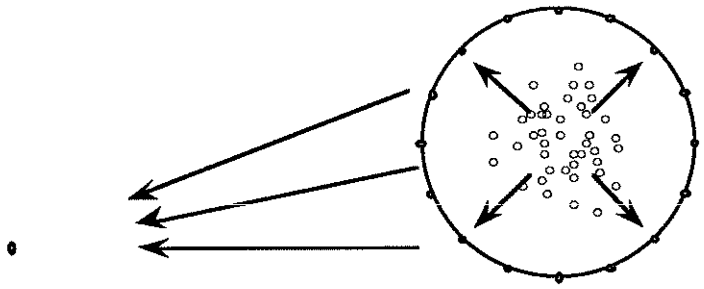


FIG. 2. *Schematic of an outer ring approximation. The computational particles (represented by the small circles) induce a potential at locations on the circumference of a ring which surrounds the particles. The potential outside the ring is evaluated by integrating these values against a (modified) Poisson kernel.*

of a ring which surrounds the particles. The potential outside the ring is evaluated by integrating these values against a Poisson kernel.

There are other possible choices for the computational elements, for example Nowak [8] uses charge distributions over panels. While each different computational element will give rise to a different method, the structure of the algorithm is relatively independent of the type of element used. We shall therefore refer to the class of methods which use the basic computational structure of the fast multipole method (but different elements) as "hierarchical element" methods.

As discussed earlier, in the multipole method one forms an approximation to the potential at a point by summing the potential induced by multipole approximations for different size clusters of particles. The same type of construction will be used with the computational elements based on Poisson's kernel. In both our method and the original multipole method, when one chooses different evaluation points one sums a different collection of these approximations. The organization of the evaluation of the appropriate approximations presents a challenging computational problem. One important component of the fast multipole algorithm as presented by Greengard-Rokhlin [4] is an effective way to construct and evaluate this hierarchy of approximations. An observation which can be made about their method for carrying out this computation is that it is much like the multigrid method. The observation has some merit, and in the second section we show how this inherent multigrid structure can be exploited to yield an efficient programming strategy for the construction and evaluation of the hierarchical set of computational elements. (The identification of the multigrid, or tree structure, in the method is also useful in analyzing the performance of such methods on parallel processors - see [6], for example.) We shall present this discussion in the context using Poisson kernel computational elements, but the results apply directly to the original multipole method as well.

The efficiency of hierarchical element methods comes about from the ability to represent a cluster of many particles by an approximation (multipole or otherwise) which can be evaluated with little numerical work. However, there is the possibility that the work to evaluate such an approximation may be more than that of evaluating the field induced by the particles in the cluster directly. This problem will certainly arise if the clusters contain only a very small number of particles. Thus, one is left with the problem of determining what is the minimal region size which should be used in forming the clusters of particles. This can be determined if the points are uniformly distributed, but if they are not, the problem is much more difficult. Fortunately, an a-priori estimate of the computational time for any level of clustering is readily available and the level which requires the smallest time can be identified and chosen. In the third section we shall discuss this issue and present the details of a method for obtaining timing estimates for each level of clustering. We will also give computational results which illustrate the importance of appropriate level selection.

Lastly we present our conclusions and discuss some of the advantages and disadvantages of the techniques presented in this paper. The programs described in this paper are available from the author.

2. Derivation of Computational Elements. A basic component of the fast multipole method is the ability to represent the potential induced by a large number of particles by a single computational element which is relatively inexpensive to compute (i.e. the multipole expansion). The use of the multipole (or spherical harmonic) expansion is not a necessity, and in this section we present a computational element which is based on Poisson's formula for a circle in two dimensions and a sphere in three dimensions.

Let $\Psi(r,\theta)$ be the potential in two dimensions induced by a collection of N particles at locations $\vec{x}_i = (r_i, \theta_i)$ and strengths κ_i which are contained within in a disk of radius a centered at the origin. This potential is a harmonic function outside of the disk, and so one can use Poisson's formula (with a log term to satisfy the circulation requirements) to represent it. Set $\kappa = \sum_{i=1}^{N} \frac{\kappa_i}{2\pi}$ and $\tilde{\Psi}(r,\theta) = \Psi(r,\theta) - \kappa\log(r)$, then if (r,θ) is a point in the plane outside the disk, we have

$$\Psi(r,\theta) = \kappa\log(r) + \frac{1}{2\pi}\int_0^{2\pi} \tilde{\Psi}(a,s)\left[\frac{1-(\frac{a}{r})^2}{1-2(\frac{a}{r})\cos((\theta-s))+(\frac{a}{r})^2}\right] ds. \tag{7}$$

Our first attempt to get a representation suitable for numerical computation was to use the trapezoidal rule to approximate the integral in (7). Let M be an integer and set K = 2M + 1. If we set $h = 2\pi/\mathrm{K}$ and $s_i = (a\cos(ih), a\sin(ih)), i = 1, \ldots, \mathrm{K}$, then an approximate representation is given by

$$\Psi(r,\theta) \approx \kappa\log(r) + \frac{1}{2\pi}\sum_{i=1}^{\mathrm{K}} \tilde{\Psi}(a,s_i)\left[\frac{1-(\frac{a}{r})^2}{1-2(\frac{a}{r})\cos((\theta-s_i))+(\frac{a}{r})^2}\right] h \tag{8}$$

However, the approximation in (8) is very ill conditioned. As the evaluation point moves towards the ring of radius a, the approximation is exceedingly inaccurate. In Figure 3 the dashed lines are the errors in the potential which are obtained using the approximation (8). The ring is of radius $a = 2$ and the potential is that induced by a single particle of unit strength located at $r = \sqrt{2}/2$ and $\theta = \pi/3$. The upper, middle, and lower dashed line correspond to K = 9, K = 17, and K = 25 points respectively in the integration formula.

The remedy for this problem becomes apparent when one considers the derivation of Poisson's formula. Let ϕ be a solution of Laplace's equation exterior to the disk of radius a which has circulation κ. If $f(\theta)$ is the value of $\phi(r,\theta) - \kappa\log(r)$ at $r = a$ we have, using the method of separation of variables,

$$\phi = \kappa\log(r) + \sum_{k=-\infty}^{k=\infty} c_k \left(\frac{a}{r}\right)^{-|k|} e^{ik\theta} \tag{9}$$

where the coefficients c_k are the Fourier coefficients of the function $f(\theta)$,

$$c_k = \frac{1}{2\pi}\int_0^{2\pi} f(s)e^{-iks}\,ds. \tag{10}$$

If one combines (9) and (10) one obtains

$$\phi = \kappa\log(r) + \sum_{k=-\infty}^{\infty}\left[\frac{1}{2\pi}\int_0^{2\pi} f(s)e^{-iks}\,ds\right]\left(\frac{a}{r}\right)^{-|k|} e^{ik\theta}. \tag{11}$$

Poisson's formula results when one interchanges summation and integration in (11),

$$\begin{aligned}\phi &= \kappa\log(r) + \frac{1}{2\pi}\int_0^{2\pi} f(s)\left[\sum_{k=-\infty}^{\infty} e^{-ik(\theta-s)}\left(\frac{a}{r}\right)^{-|k|}\right] ds\\ &= \kappa\log(r) + \frac{1}{2\pi}\int_0^{2\pi} f(s)\left[\frac{1-(\frac{a}{r})^2}{1-2(\frac{a}{r})\cos((\theta-s))+(\frac{a}{r})^2}\right] ds.\end{aligned}$$

From this derivation one can see that using a trapezoidal rule approximation to the integral in (7) is equivalent to implicitly using the trapezoidal rule to approximate the Fourier coefficients (10). Clearly, if only K = 2M + 1 points are used in the quadrature rule then one should not use any coefficients c_k with $|k| > \mathrm{M}$, i.e. one should only use those Fourier modes in (9) which can be reliably estimated using K equispaced points. Thus, we consider using (9) with only the first M modes,

$$\begin{aligned}\phi &\approx \kappa\log(r) + \frac{1}{2\pi}\int_0^{2\pi}\left[\sum_{k=-\mathrm{M}}^{\mathrm{M}} e^{-ik(\theta-s)}\left(\frac{a}{r}\right)^{-|k|}\right] f(s)ds\\ &= \kappa\log(r) + \frac{1}{2\pi}\int_0^{2\pi} f(s)\left[\frac{1-(\frac{a}{r})^2-2(\frac{a}{r})^{\mathrm{M}+1}\cos((\mathrm{M}+1)(\theta-s))+2(\frac{a}{r})^{\mathrm{M}+2}\cos(\mathrm{M}(\theta-s))}{1-2(\frac{a}{r})\cos((\theta-s))+(\frac{a}{r})^2}\right] ds.\end{aligned}$$

Here, as in the original derivation of the Poisson kernel, the simplification arises from the formula for summing a geometric series. When this integral is approximated by the trapezoidal rule we obtain the numerical approximation,

$$\phi(r,\theta) \approx \tag{12}$$

$$\kappa \log(r) + \frac{1}{2\pi}\sum_{i=1}^{K} f(s_i) \left[\frac{1-(\frac{a}{r})^2-2(\frac{a}{r})^{\mathrm{M}+1}\cos((\mathrm{M}+1)(\theta-s_i))+2(\frac{a}{r})^{\mathrm{M}+2}\cos(\mathrm{M}(\theta-s_i))}{1-2(\frac{a}{r})\cos((\theta-s_i))+(\frac{a}{r})^2}\right] h$$

where the integration points s_i are equispaced on the ring with $h = 2\pi a/\mathrm{K}$. By construction, this discrete approximation is precisely that which would be obtained if one formed an approximate solution of the exterior Laplace equation by combining solutions for the first M modes of the data $f(s)$. The Fourier coefficients used in the approximation are, however, those obtained from the discrete, rather than the continuous, Fourier decomposition of $f(s)$.

We will use (12) with $\kappa = \sum_{i=1}^{\mathrm{N}} \frac{\kappa_i}{2\pi}$ and $f(s_i) = \Psi(a, s_i) - \kappa \log(a)$ to approximate the potential Ψ induced by N particles of strengths κ_i. The steps to create this approximation consist of summing the strengths of the particles in a region of space to obtain the strength of the log term in the approximation. This is then followed by the evaluation of the potential induced by the particles (minus the log term) at equispaced points on the ring of radius a which encompasses these points. To evaluate this approximation at some point outside the disk, one merely adds the contribution of the log term and the sum in (12). As before, if the number of particles N contained within the ring are large compared to K, a substantial saving in work is accomplished by using such an approximation. We shall refer to the approximation (12) as an outer ring approximation.

The improvement in accuracy when (12) is used is demonstrated by the solid lines in Figure 3. In this figure, these lines indicate the error in the potential when (12) is used for the problem of a point charge located at $r = \sqrt{2}/2$ and $\theta = \pi/3$. The upper, middle, and lower solid lines correspond to K = 9,K = 17, and K = 25 integration points respectively. As can be seen from the figure, the error remains sufficiently small as the evaluation point approaches the ring used in the approximation (a ring of radius 2). This behavior is in sharp contrast to the error in the approximation with the unmodified kernel (8) which gives O(1) errors as the evaluation point approaches the ring.

Complete error estimates for the approximation have yet to be worked out, but the accuracy can be partially assessed using the results of Greengard and Rokhlin [4]. In particular, in the derivation of (12) one sees that the approximation is formed by combining the solutions of Laplace's equation corresponding to the first M Fourier modes associated with the data $f(s)$. If in this approximation we used the exact Fourier coefficients (10) instead of the discrete Fourier coefficients, then the approximation which would result would be identical to that of a multipole expansion with M terms. Hence, using the result of [4] we expect that if the particles are centered within a disk of radius α about the origin then the error in the potential at any test point with a radial distance r will be $\mathrm{O}((\frac{\alpha}{r})^{\mathrm{M}})$. It is important to note that this estimate of the error depends on the relative location of the evaluation point to the ring of radius α which surrounds the particles, and not the relative location of the particle from the ring of radius a which is used in the approximation. For example, the solid lines

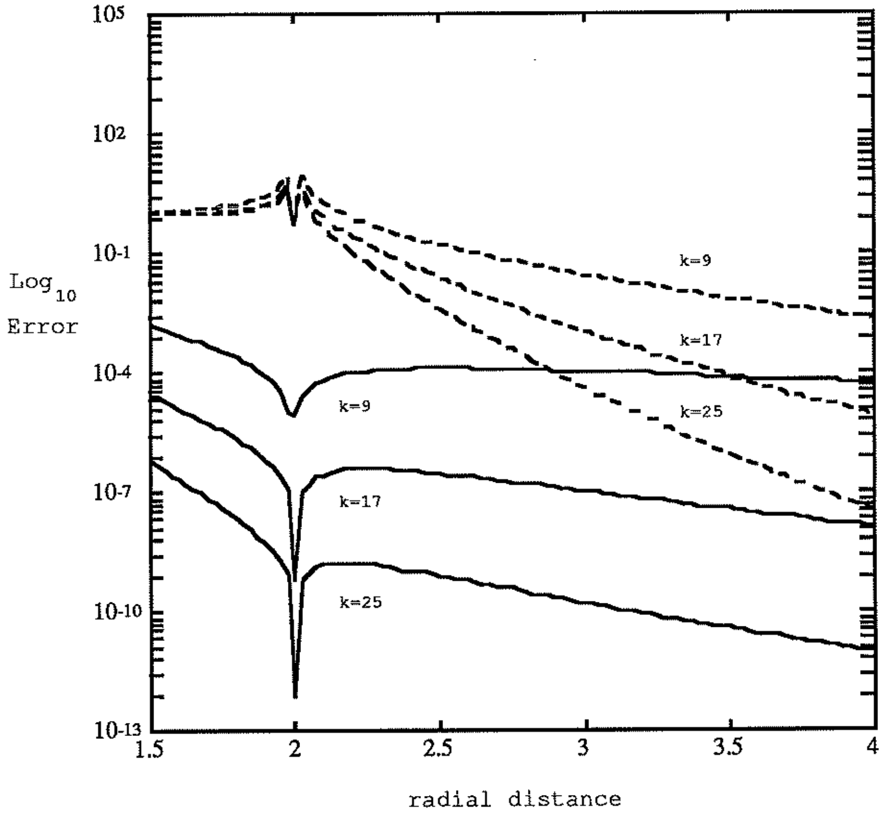


FIG. 3. *Error in the potential when outer ring approximations are used. The dashed lines correspond to an unmodified Poisson kernel, and the solid lines to a modified Poisson kernel. The ring radius was $a = 2$ and the value of K refers to the number of points in the integration formula.*

in Figure 3 indicate that the error in the approximation remains small even as the evaluation point moves inside the ring which is used for the approximation. This is expected, and in fact necessary, since in the complete algorithm we shall have reason to evaluate the approximation for points inside the ring. The ring radius which used in the approximation (12) has an indirect effect on the accuracy. Essentially, the distance of the ring from the center of the disk which contains the particles effects the aliasing which occurs with the implicit use of discrete, rather than continuous Fourier coefficients. We found that setting the ring radius $a = 2\alpha$ gives acceptable results.

In the complete method there is the need to represent the potential inside a given region. Following the same strategy for obtaining a numerically stable outer ring approximations we define an inner ring approximation by

$$\phi(r,\theta) \approx \frac{1}{2\pi}\sum_{i=1}^{K} f(s_i)\left[\frac{1-(\frac{r}{a})^2-2(\frac{r}{a})^{\mathrm{M}+1}\cos((\mathrm{M}+1)(\theta-s_i))+2(\frac{r}{a})^{\mathrm{M}+2}\cos(\mathrm{M}(\theta-s_i))}{1-2(\frac{r}{a})\cos((\theta-s_i))+(\frac{r}{a})^2}\right] h \tag{13}$$

where (r,θ) is the evaluation point and $f(s_i)$ is the value of the potential induced by particles (or outer ring approximations) outside the ring of radius a. The evaluation point can be taken to be outside as well as inside the ring.

To make efficient use of approximations of the form (12), clustering is done on different levels, i.e. approximations are constructed for collections of particles in clusters of increasing size. In this construction it is necessary to combine several outer ring approximations into a single outer ring approximation. (In the multipole method, this is carried out by shifting the origin of the multipole expansions.) This operation of combining outer ring approximations is particularly simple to implement, one just evaluates the potential induced by the component outer ring approximations at the integration points of single outer ring approximation under construction. (See Figure 4). Similarly for the other required combining operations in the method, i.e. forming inner ring approximations to represent the potential of outer ring approximations etc., these are all carried out by just evaluating the component approximations at the set of points needed for the particular approximation. Due to the close relationship between these operations and those that are used in the original multipole method, the errors for ring approximations with K = 2M+1 integration points can be expected to behave in a similar fashion to the errors in the multipole method when M terms in a multipole expansion are used.

In three dimensions the construction is essentially the same. Let $g(\theta,\phi)$ denote values on a sphere of radius a and denote by Ψ the harmonic function external to the sphere with these boundary values. Given a point outside the sphere of radius a, $\vec{x} = (r,\theta,\phi)$, let $\vec{x}_p = (\cos(\theta)\sin(\phi), \sin(\theta)\sin(\phi), \cos(\phi))$ be the point on the unit

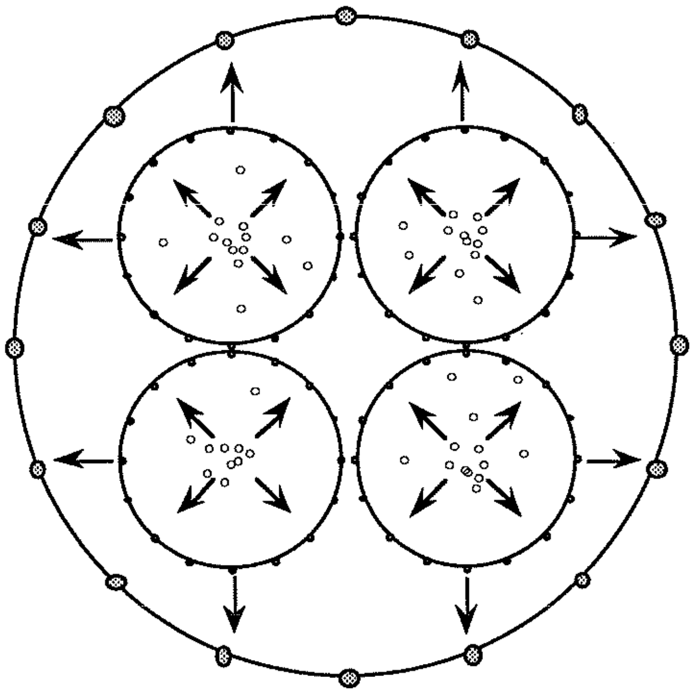


FIG. 4. *Schematic diagram of the operation of combining outer ring approximations. The coarse level outer ring approximation is obtain by evaluating the potential of the outer ring approximations contained within it.*

sphere which points in the direction of $\vec{x}$, then

$$\Psi(\vec{x}) = \frac{1}{4\pi}\int_{S^2}\left[\sum_{n=0}^{\infty}(2n+1)(\frac{r}{a})^{n+1}\mathrm{P}_n(\vec{s}\cdot\vec{x}_p)\right] g(a\vec{s})\,ds \tag{14}$$

where the integration is carried out over S^2, the surface of the unit sphere, and P_n is the nth Legendre function. (See [3], page 513).

As is the case with the ring approximations we shall refrain from using all of the terms in the kernel (14). Given a numerical formula for integrating functions on the surface of the sphere with K integration points $\vec{s_i}$ and weights w_i we use an approximation of the form

$$\Psi(\vec{x}) \approx \sum_{i=1}^{K}\left[\sum_{n=0}^{M}(2n+1)(\frac{r}{a})^{n+1}\mathrm{P}_n(\vec{s_i}\cdot\vec{x}_p)\right] g(a\vec{s_i})\,w_i. \tag{15}$$

We call this approximation an outer sphere approximation.

A critical choice to be made is that of the appropriate number of terms M of the kernel in (15). If one uses an integration formula for the sphere of degree D (i.e. the formula integrates polynomials of up to and including degree D exactly) then an appropriate choice for M is $M \leq \frac{D}{2}$. The reasoning for this is as follows: Implicit in the use of Poisson's formula is the computation of an orthogonal expansion in spherical harmonics. If we keep M terms in the kernel then we are constructing an approximation by combining the potentials corresponding to the first M terms of this expansion. In a numerical approximation of the type (15) we are implicitly using a discrete, and hence accuracy limited, approximation to estimate the orthogonal expansion coefficients. We should therefore choose M on the basis of our ability to computationally estimate the coefficients of a spherical expansion with degree less than or equal to M. Given a function on the sphere comprised of spherical harmonics of degree m, then the spherical expansion coefficients of this function can be determined exactly by an integration formula with accuracy of degree $2m$. (Spherical harmonics of degree $\leq m$ can be expressed as the restriction of polynomials of at most degree m to a sphere, so a formula of degree $2 * m$ can calculate inner products of two of these functions exactly.) Therefore, if we ignore the effects of aliasing, using $\mathrm{M} \leq \frac{\mathrm{D}}{2}$ allows us to determine the first M coefficients in the orthogonal expansion accurately. The error in the resulting approximation can be expected to be on the order of the first neglected term in the expansion - i.e. if we keep M terms, then the error in the potential induced by a collection of particles inside a sphere of radius α about the origin should behave like $\mathrm{O}(\frac{\alpha}{r})^{\mathrm{M}+2}$. (The highest power of r in the sum is $\mathrm{M}+1$.)

Unlike two dimensions in which the trapezoidal rule furnishes us with the optimal integration formula for our integration, in three dimensions the choice of integration formula is more complicated. One obvious choice is the use of product integration formulas. The ones typically employed are those which use trapezoidal integration in the θ direction and then Gaussian quadrature in the ϕ direction. These are somewhat inefficient as the integration points are crowded near the poles so we chose to use integration formulas from a non-product family, namely those described in [10] which are taken from [7]. In Figure 5 and Table 1 we present the results of computations obtained using (15). Each of the curves in Figure 5 are the errors in the potential induced by a single unit strength particle located at $r = \sqrt{3}/2$, $\theta = \pi/3$ and $\phi = 0$. The test points are located along the x-axis ($\theta = 0, \phi = \pi/2$). This selection of points was chosen to be representative of the worst possible (most inaccurate) configuration which would occur when this element is incorporated into the complete method. The individual curves correspond to a 5th, 7th, 9th, 11th and 14th order integration formulas. These formulas required 12, 24, 32, 50, and 72 points respectively. In Table 1 we present the rates of decay of the error as the radial distance to the evaluation point increases. As expected the rate was approximately $\mathrm{M}+2$. It is interesting to note that the 9th order formula has errors which are much better than expected - it appears that the integration formula is of higher order than advertised.

The approximation which is used to represent potentials inside a given region is

$$\Psi(\vec{x}) \approx \sum_{i=1}^{\mathrm{K}} \left[\sum_{n=0}^{\mathrm{M}} (2n+1)(\frac{r}{a})^n \mathrm{P}_n(\vec{s}_i \cdot \vec{x}_p) \right] g(a\vec{s}_i)\, w_i \tag{16}$$

where the notation is that used to define (15). (Note the change in the power of $\frac{r}{a}$ in this formula.) We shall refer to (16) as an inner sphere approximation.

The operations associated with forming and combining the outer and inner sphere approximations are essentially the same as those in two dimensions - one merely eval-

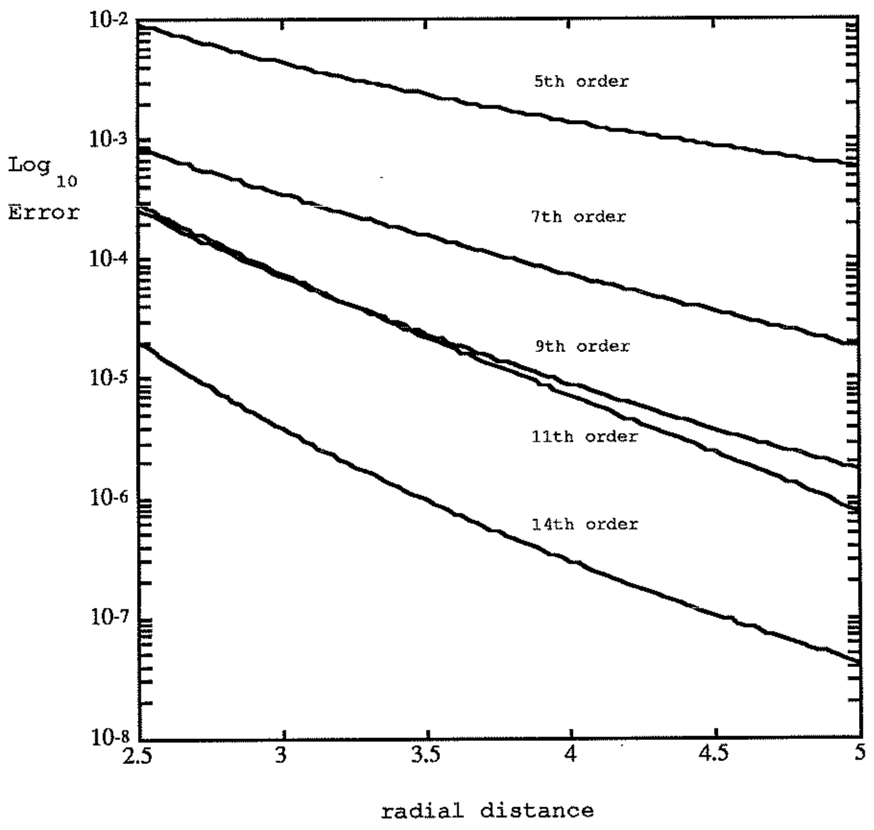


FIG. 5. *Error in the potential for an outer sphere approximation. Each curve represents the error for a given order of integration formula (and fixed number of terms in the Poisson kernel.) The sphere was of radius 3.*

uates the component approximations at the integration points of the new approximation. If the integration points of the new approximation are sufficiently far away from the old approximation, then the accuracy degradation is minimal. One of the big advantages of using outer and inner sphere approximations is the simplicity of the process of combining them. This is in contrast to the complicated formulas which must be used if the approximations are based on spherical harmonics.

3. Hierarchical Element Implementation and Multigrid. In this section we discuss an implementation strategy which uses the correspondence between the algorithmic structure of the fast multipole method [4] and the multigrid method. Although we shall discuss an implementation which uses computational elements based on the Poisson kernel (12) and (15), there is no explicit use of the particular features of this computational element, and so the results apply to programming with multipoles or any other type of element. Our goal here is to just outline the computational procedure which we have used and for more detailed information one should see the programs in [1], [2].

The basic algorithm structure used is that presented in [4]. The computation is broken down into two sweeps, the first sweep consists of constructing outer ring

Order of Integration Method	Number of Terms in Kernel	Average Rate of Error Decay γ	Expected Rate of Error Decay γ
5	2	3.95	4
7	3	5.68	5
9	4	8.85	6
11	5	7.322	7
14	7	8.902	9

TABLE 1
Rate of the spatial decay of the error in the potential for an outer sphere approximation. The potential is assumed to decay like $O(r)^\gamma$

approximations for a hierarchical clustering of the computational particles. The second sweep consists of evaluating (in an efficient manner) the appropriate members of this hierarchy for each of the required evaluation points. There is no need for the evaluation points to be located at the same positions as the computational particles. In both sweeps the computational particles and evaluation points are assumed to lie in a bounded region Ω. A square box is then chosen which encloses Ω. The box is then recursively divided into smaller boxes. At the 0th level we have the original box, at the next level 4 boxes, the next 16 etc. Associated with each level n we have a uniform grid dividing up the original box into 4^n boxes. At the beginning of the calculation one chooses a highest (finest) level of discretization. We shall call this level n_f.

In the first sweep, outer ring approximations are constructed to represent the potential induced by the computational particles in each of the boxes at every level. This construction is performed recursively. Starting at the finest level, outer ring approximations are constructed for the computational particles in each of the boxes at that level. The center of the outer ring approximations are located at the centers of the respective boxes. The radius for the ring approximation is taken to be twice the size of the box. One then proceeds to the next lower (coarser) level and forms outer ring approximations for each of the particles contained within these coarser boxes. Instead of forming an outer ring approximation directly from the particles contained within a given box, one combines the four outer ring approximations associated with the finer boxes contained within the particular coarse box. The process of combining outer ring approximations is accomplished by evaluating the finer level outer ring approximations at the integration points of the coarse level outer ring approximation. Again, the centers of the outer ring approximations at this coarser level are located at the centers of their respective boxes and their radius is twice the coarse box size. This procedure is then repeated for all successively coarser levels. When the sweep is completed, outer ring approximations are available for the particles within each box at every level.

Two issues to be concerned with in implementing this sweep are avoiding unnecessary work and avoiding unnecessary storage requirements. A straight forward implementation would suggest that at each level one constructs an outer ring approximation for each box. If one uses K nodes in the approximation (12), then for n_f levels the storage required will be approximately $\mathrm{K} \times \frac{4^{(n_f+1)}}{3}$ real numbers. Also, the work to construct all of these approximations for N uniformly distributed particles will be on the order of $\mathrm{K} \times \mathrm{N} + \mathrm{K}^2 \frac{4^{(n_f+1)}}{3}$. While this amount of work and storage is necessary if the particles are uniformly distributed, this amount is certainly not needed if they are not. If a given box (at any level) has no computational particles, then one need not compute an outer approximation for that box. Therefore, for non-uniform distributions (which are perhaps more prevalent in applications) one can save on storage and time by accounting for this fact.

The construction carried out in this first sweep is very similar to the computation which is performed in one sweep of a multigrid method. Both types of methods progress through a nested set of grids or boxes, and the operation of forming an outer approximation on a given level from the four outer approximations at the previous finer level is analogous to implementing the restriction operator (the forming of a coarse grid approximation from a fine grid one) in a multigrid method. However, in a hierarchical element method one is dealing with outer approximations associated with the grid boxes rather than solution values, and so the programming strategy is to implement the equivalent of a multigrid restriction operator but with this operator defined to act on approximations *pointed to* by address values in the boxes rather than to act on solution values. To carry out this computation a multigrid pointer structure, F_n, consisting of a nested set of integer arrays is first constructed. At a given level n, F_n is an integer array of size $2^n \times 2^n$. The $(i,j)th$ element of F_n is the starting address in a storage array of the values used in the outer ring approximation associated with the $(i,j)th$ box of the nth level partition of the computational domain. If there is no outer ring approximation associated with that box, then a negative integer is stored in the pointer array.

The computation is now performed as follows. The particles are sorted according to which box at the finest level (level n_f) they reside in. Particles in the same box are link listed together, and the address of the first element of the link list for any given box is stored in a $2^{n_f} \times 2^{n_f}$ integer pointer array I_{n_f} associated with the boxes at the finest level. (By link listing we mean that we associate with each particle an integer, the value of this integer gives the address of the next particle in the same box.) If there are no particles in a given box, then the value stored in I_{n_f} is negative integer. Outer approximations are constructed for each of the boxes at the finest level. This is accomplished by looping over the array I_{n_f}. If there is a non-negative element in I_{n_f} then there are particles in the corresponding box and the potential induced by these particles are evaluated to obtain the values necessary for an outer ring approximation. These outer ring potential values are put into a storage array and the beginning address of these values is stored in the array F_{n_f}. Next, outer ring approximations are constructed for each coarser level. If we are at level k, then

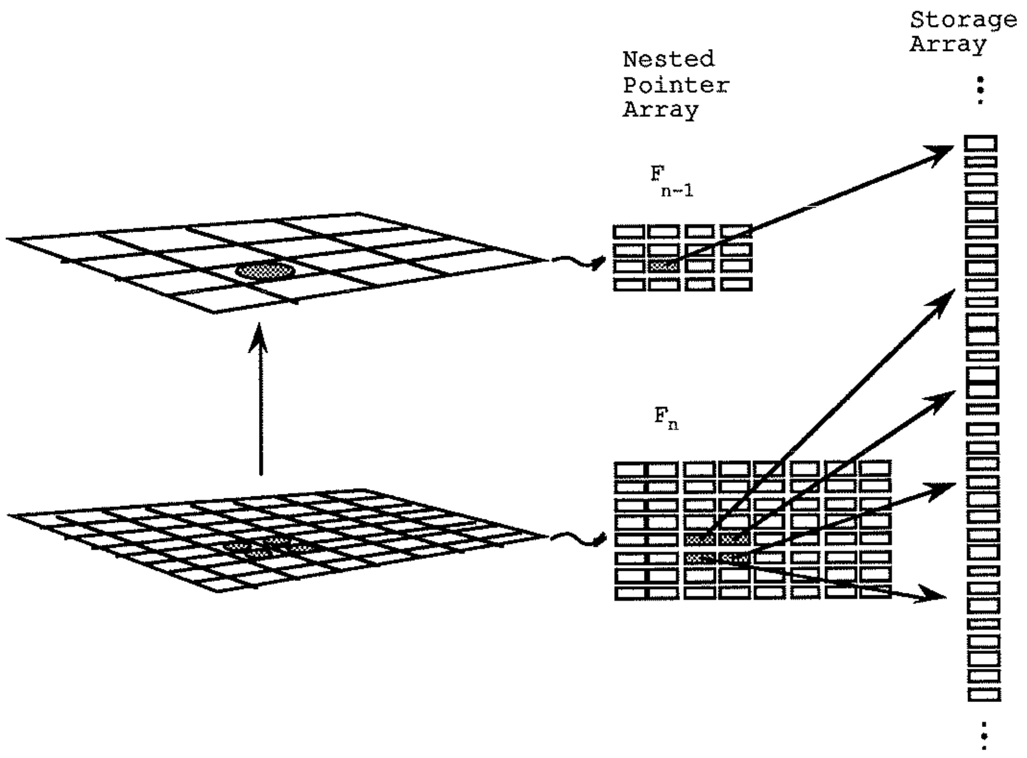


FIG. 6. *Schematic of outer ring construction. Formation of the coarse level outer ring approximation is performed by combing the four finer outer ring approximations beneath it. Addresses which point to the storage array in which the outer ring approximation values are stored in the nested arrays* F_n

the pointer array corresponding to the finer level F_{k+1} is looped over. If there is an outer ring approximation associated with a given box at the finer level (indicated by a non-negative entry in the pointer array), then the box at the coarser level which contains this finer level box (it's parent box) will have an outer ring approximation associated with it. The potential induced by the finer level outer ring approximation is evaluated to construct the coarse box outer ring approximation. These new outer ring approximation values are put into a storage array and the beginning address of the values is stored in the F_k. A pictorial diagram of this process is depicted in Figure 6. This process of constructing outer ring approximations is repeated until the second level is reached. (No outer ring approximations are needed for the first or zeroth levels.)

The second sweep, that in which the evaluation of the outer ring approximations is organized, utilizes the concept of being "well separated" [4]. Assume that the boxes on a given level are ordered in standard lexicographic ordering, then, for a given level of refinement, one box is "well separated with distance L" from another box, if the maximum of the difference between their indices is greater than or equal to L (i.e. the boxes are at least L boxes apart). For a given level, the second sweep consists of constructing inner ring approximations for each box. The inner ring approximations are used to represent the potential from two sources. The first source is the inner

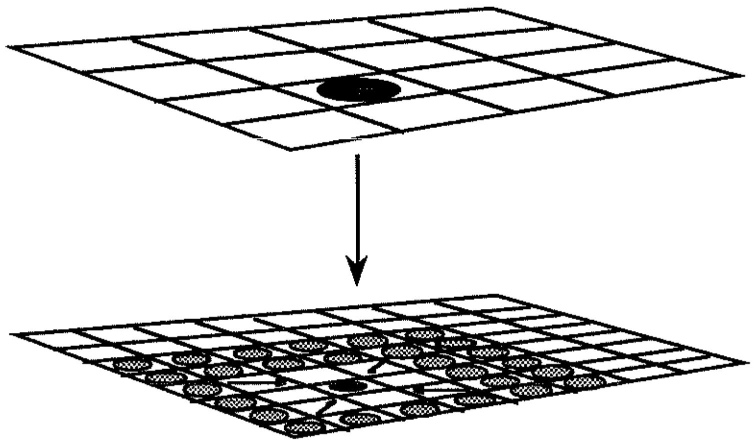


FIG. 7. *Formation of inner ring approximations from parent inner ring approximation and well separated outer ring approximations.*

ring approximation associated with the parent box at the previous coarser level. The second source is from all of the outer ring approximations from boxes which are well separated from the given box (with distance 1 in two dimensions and distance 2 in three dimensions) and are contained within boxes at the previous coarser level which are not well separated from the parent box. A graphical representation of this step is given in Figure 7. This procedure is carried out for all levels except the finest. For this level, the inner ring approximations are only constructed from the inner ring approximations of the parents. At every level, the centers of the inner ring approximations are coincident with the centers of the boxes which which they are associated. The radius of the ring used in the approximation is one half the box size. With the completion of this sweep, the potential at any given evaluation point is obtained by computing the potential of the inner ring approximation associated with the finest level box the point resides in. This potential is then added to the potential induced by particles in the nearby finest level boxes which are not well separated from the given evaluation points box. The reason for choosing a separation distance L = 1 in two dimensions and L = 2 in three dimensions is to ensure that the decay of the error in the approximations behaves like $O(\frac{1}{2})^{\gamma}$ where γ is determined by number of modes kept in the kernel M. (See formula (12)).

As is the case with the first sweep one must avoid using unnecessary storage and doing unnecessary work. Here, at any given level, if there is no evaluation point within a box, then one need not compute an inner ring approximation for that box. Work and storage space can be reduced by accounting for this fact.

The construction carried out in this second sweep is also very similar to the computations which are performed in one sweep of a multigrid method. The process of creating an inner approximation from potential values induced by an inner ring approximation associated with a parent box is much like the prolongation operation of multigrid i.e. the construction of a fine grid solution from a coarse grid one. The process of combining the outer ring approximations from well separated boxes is analogous to performing relaxation. Again, in this sweep we are working with approximations associated with the grid boxes rather than solution values, and so the programming strategy will be to implement multigrid type operators, but with the operators defined to act on the approximations pointed to by address values in a given box rather than solution values. To implement this sweep, we again use a nested pointer array, G_n similar to F_n. At a given level n the size of G_n is $2^n \times 2^n$ and the value at the $(i,j)th$ element of this array is the starting address (in a storage array) of the inner ring approximation values associated with the $(i,j)th$ box of the nth level partition. If there is no inner approximation associated with that box, then a negative integer is stored in the pointer array.

The first task in the actual computation is to identify which boxes will have inner ring approximations associated with them. The general rule is that if a box at any level has an evaluation point in it, then an inner ring approximation will be necessary for that box. The evaluation points are first sorted according to which box at the finest level (level n_f) they reside in. Particles in the same box are link listed together, and the address of the first element of the link list for any given box is stored in an integer pointer array J_{n_f} associated with the finest level boxes. Next the pointer structure G_n is swept through, starting at the finest level. At each level, a zero is stored in the pointer array if the corresponding box in the computational space with which it is associated with contains at least one evaluation point. (This can be checked by considering the presence of zeros in the pointer array locations of the next finer level.) Once this initial sweep is performed then any of the boxes whose pointer is zero indicates that an inner ring approximation will have to be formed for that box.

Now, starting at the coarsest level, each level of the nested pointer array G_n is swept through. If there is a zero in the pointer array location at a given level, then an inner ring approximation is constructed from a parent inner ring approximation and from the appropriate well separated outer ring approximations. These approximations are combined by merely evaluating the potential they induce at the nodes of the inner ring approximation under construction. The addresses of the values needed to evaluate the parent inner ring approximation and the outer ring approximations are obtained from the pointer arrays G_n and F_n at the appropriate level. The values for this inner ring approximation being constructed are put in a storage array and the beginning address of these values is stored in the pointer array G_n . At the last (finest) level, inner ring approximations are formed from the parent inner ring approximations. The potential at the evaluation points is then obtained from these finest level inner ring approximations and direct evaluation of the potential from particles in nearby (not well separated) boxes. The particles in nearby boxes are accessed via the initially constructed link list whose starting addresses are pointed to by the entries in J_{n_f}.

The entire procedure consists of these two sweeps. The first sweep goes from a fine discretization of physical space to a coarse discretization and results in the construction of a hierarchy of outer ring approximations. The second sweep goes from a coarse discretization to a fine one and results in a hierarchy of inner ring expansions. If one considers the operations which are being performed on the pointer arrays, then the method is algorithmically similar to a multigrid V-cycle. In fact, the first version of the code was obtained by modifying an existing multigrid code. The modification consisted of changing the subroutines used in multigrid (like a relaxation routine) so that instead of working with the values at the cell centers, the routine worked with the approximations which were pointed to by the values at cell centers. It proved to be very productive to use the programming techniques which have developed for multigrid, specifically those which are embodied in the sample code given in [11].

It is clear that if the computational particles and the evaluation points are distributed in a region which is not well represented by a square then there will be a great number of elements in the pointer arrays which will be associated with no outer or inner approximations. In order to reduce storage space, and shorten loops over the pointer structure one can allow for regions which are non-square. This is most easily accomplished if the bounding region is constructed to be a rectangular box whose sides have an aspect ratio which is a power of two. To implement this a rectangular box is constructed which contains the computational particles and the evaluation points. The sides of this box are then enlarged to create a rectangle which has an appropriate aspect ratio. The refinement of this rectangle proceeds by successively dividing the larger side by powers of two until the box side is equal to the length of the smaller side. From this point on the refinement is accomplished by refining the boxes in both directions. (See figure 8 which depicts the refinement of a box with aspect ratio four to one.) The corresponding nested set of pointer arrays F_n, G_n, I_{n_f} and J_{n_f} will be non-square and contain a number of elements corresponding to the number of boxes at each level of the decomposition of the original rectangle.

In three dimensions we found that using an outer sphere radius which was three times the box size worked better than using one which was only twice the box size. Other than this difference and the changes in the number of indices in the pointer arrays and the form of the kernel in Poisson's formula, the implementation of the three dimensional method is identical to the two dimensional method described above.

4. Timing and Level Selection.

As discussed in the introduction, the primary component of a hierarchical element method which leads to a computational savings is the representation of the potential induced by a large number of particles by a single computational element which requires less work to evaluate. On the other hand, it is clear that if one uses a computational element to represent the potential induced by only a small number of particles, then this may lead to more, rather than less, computational work. In the implementation of the method presented here, the computational domain is broken up into boxes, and as one increases the level of

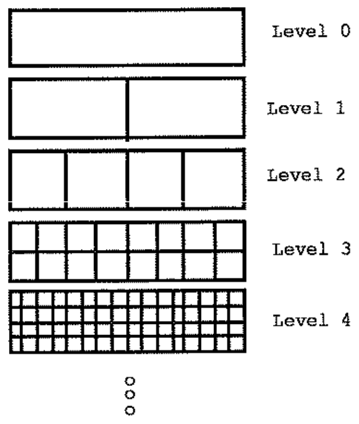
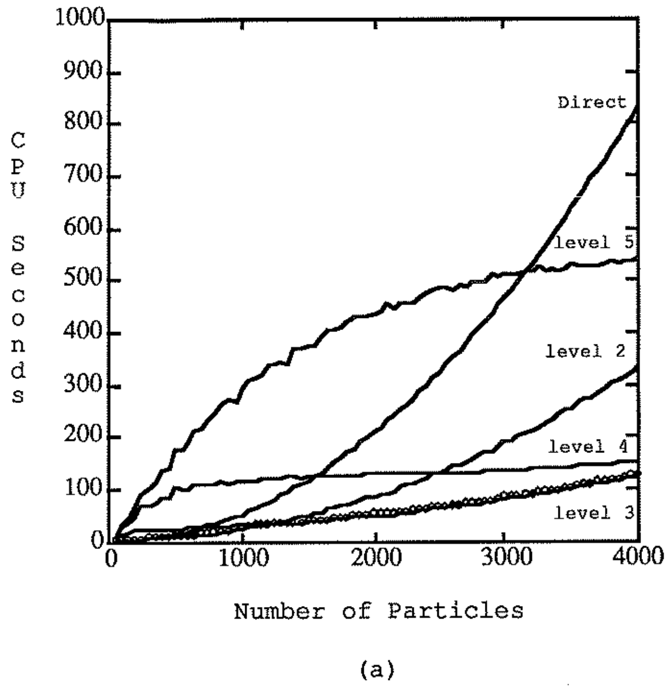


FIG. 8. *Refinement of a rectangle with an aspect ration of four to one.*

refinement, the size of these boxes decreases. Since the number of particles per box will ultimately decrease, and thus the number of particles per computational element will decrease, there is some level for which it is more expensive to use a hierarchical element method than to use the direct method. This is illustrated by the results in Figure 9 and Figure 10. (All of the computations where carried out on a SUN Sparcstation 1. There were 25 points in the ring approximations - a number of points which provides an accuracy of 10^{-4}.) The figures give the time it takes to compute the potential induced by all of the particles at the locations of all the other particles for different numbers of particles. (These times are estimated by the procedure described below.) For Figure 9 the particles were uniformly distributed in a square and for Figure 10 the particles were uniformly distributed in a rectangle with aspect ratio of ten to one. Each of the separate lines in the figures correspond to different finest levels of domain discretization. As is evident from the figures the level one should choose depends on the number of particles. The more the particles - the higher the level of refinement. The choice of level also depends on the distribution of particles. The fact that the timings vary greatly as the level is changed indicate that it is crucial to choose the appropriate level. If the particles are always uniformly distributed in some particular geometric arrangement, then one could work out estimates of the appropriate level of refinement one should choose. If the particles are not distributed in such a way, then the construction of a scheme for identifying the appropriate level seems difficult. However, a computationally efficient and optimal level selection scheme can be constructed.

The key observation is that the amount of computational work at each level in the hierarchical element method is completely determined (and hence one can estimate it) by the number of computational particles and evaluation points in any given box at any given level. In essence the selection scheme is to "dry run" the hierarchical element method, and, instead of performing all the required operations, just increment counters. The counter increments are based on the population density of the particles in each box and models of the computational time necessary to carry



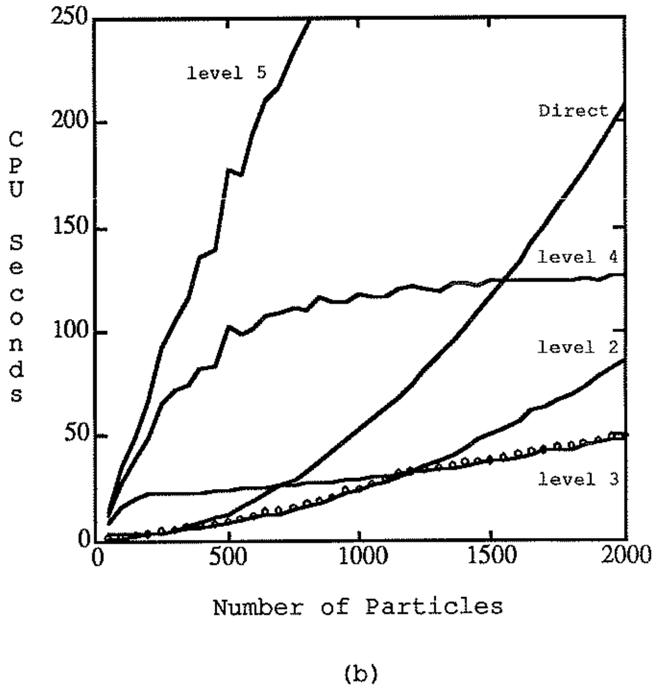


FIG. 9. *CPU time for a hierarchical element method based on ring approximations. The particles are distributed uniformly in a unit square. Each of the solid curves represents the estimated CPU time used by the method for a given level of spatical refinement. The circles represent the actual time taken when the minimal estimate is chosen. (b) is a magnified version of (a).*

out specific computational tasks. Timings are then estimated for the work required for different levels of refinement, and the level selected is the one with the least amount of estimated time. It is not necessary to perform a different timing computation for every level of refinement - the recursive nature of the method allows one to compute the amount of work for each level up to some ultimate level in one computation.

Before presenting the details of the implementation of this selection procedure, we present some results. In Table 2 we present the CPU time estimate (in seconds) from our selection code and the actual CPU time taken for a computation of the potential induced by 2000 points distributed in a unit square. As can be seen, one can get rather good estimates of the time it takes for any given level. The cost of performing this estimation is very small. For 2000 points it takes .6 CPU seconds to estimate the running time for 6 levels of refinement. This time is about 1.25% of the time it takes to actually perform the calculation. Figures 9 and 10 also illustrate the accuracy of the timing estimates. The circles in these figures correspond to the actual CPU time taken to perform the computation when the level with the least amount of estimated time is selected. As desired, these circles coincide with a lower envelope for all of the timing curves in the figure.

In the implementation of the method for approximating the computational time we use a nested set of integer arrays set up like F_n and G_n of section 3. (A nested set of arrays whose individual elements at any level are associated with individual squares in the decomposition of physical space at that same level of refinement.) We

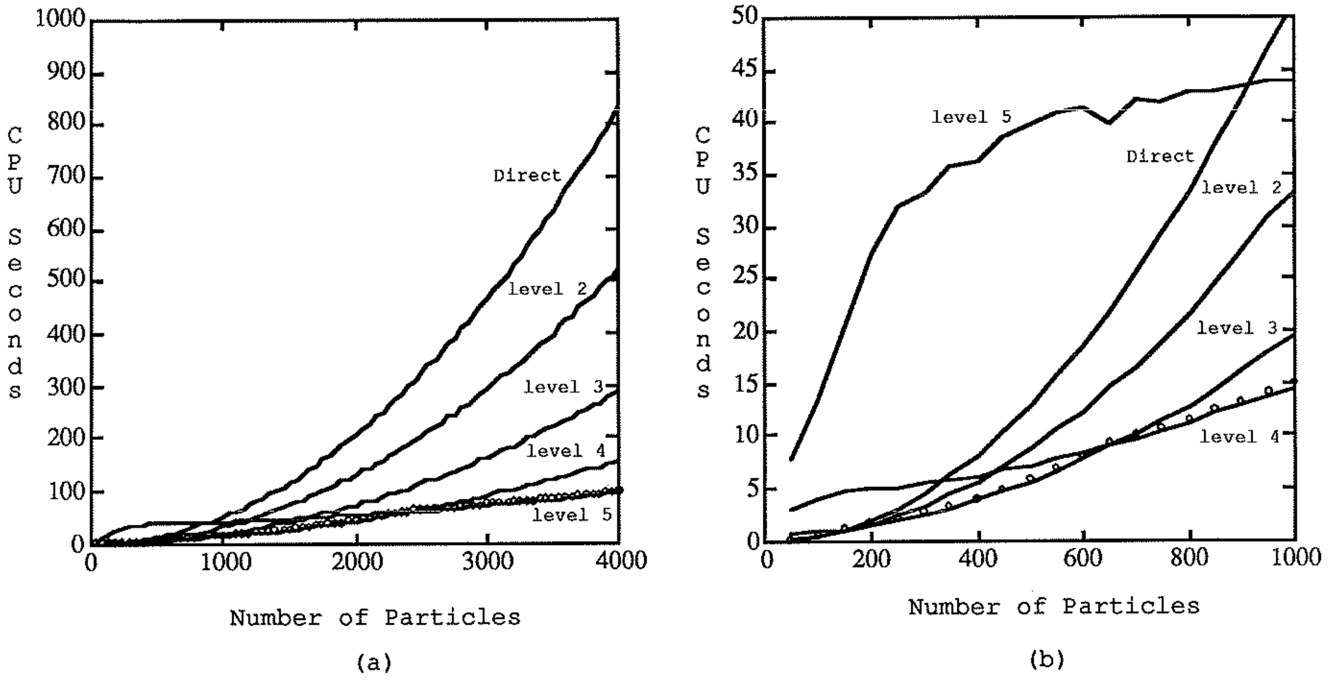


FIG. 10. *CPU time for a hierarchical element method based on ring approximations. The particles are distributed uniformly in a rectangle with an aspect ratio of 10 to 1. Each of the solid curves represents the estimated CPU time used by the method for a given level of spatical refinement. The circles represent the actual time taken when the minimal estimate is chosen. (b) is a magnified version of (a).*

Level	Estimated CPU Time (Secs.)	Actual CPU Time (Secs.)
Direct	207.3	206.8
2	85.5	86.0
3	48.1	48.4
4	126.4	123.5
5	443.2	430.9

TABLE 2
Estimated and Actual CPU time for a hierarchical element method based on ring approximations. The timings are for 2000 particles distributed uniformly in a unit square.

have one nested set of arrays S_n associated with the computational particles and one nested set of arrays T_n associated with the evaluation points. At each level of these nested arrays we store in the $(i,j)th$ element the number of particles (or evaluation points) contained within the $(i,j)th$ box in computational space at that level. To construct these population values a finest level of refinement is chosen and the number of particles in each box is stored in the corresponding array element of S_n associated with that level. Next, the other levels of the array S_n are swept through and the number of particles in each box is determined by summing the number of particles in each of the boxes at the finer level which are contained within the given box. The same procedure is carried out for the evaluation points and the nested array T_n.

In order to use the population density information in the arrays S_n and T_n, we need models of the computational time which is necessary to carry out particular operations of the method. Assume there are n computational particles and m evaluation points. In two dimensions, if we are using an approximations of the type (12) with K points, then the models used are

(0) *direct interaction*	:	$c_0 \times n \times m + b_0$
(1) *formation of outer approximations*	:	$c_1 \times n \times K + b_1$
(2) *combining; formation of outer approx. from outer approx. etc.*	:	$c_2 \times K \times K + b_2$
(3) *evaluation of finest level inner approx.*	:	$c_3 \times m \times K + b_3$

These models depend upon values of the constants $b_0, c_0, b_1, \ldots, b_3, c_3$. We call these formulas models, because they are not exact - peculiarities in specific computational hardware or compilers may mean that the constants are not truly constants. However, on the machines the author has worked with, these formulas capture rather well the actual amount of computational time for a wide range of the parameters n, m and K. In order to estimate the values for these constants, a small program was written which times the specific operations. Using the results of these timing estimates for several values of n, m etc., the constants in the models were determined by a least squares procedure.

In three dimensions the models are slightly different because the finite sum in the integral kernel in Poisson's formula is not represented by a single closed formula. Assuming n computational particles, m evaluation points, K integration points and M terms in the Poisson kernel, we use the following computational models,

(0) *direct interaction*	:	$c_0 \times n \times m \ + b_0$
(1) *formation of outer approximations*	:	$c_1 \times n \times \mathrm{K} \ + b_1$
(2) *combining - formation of outer approx. from outer approx. etc.*	:	$c_2 \times \mathrm{K} \times \mathrm{K} \times \mathrm{M} \ + b_2$
(3) *evaluation of finest level inner approx.*	:	$c_3 \times m \times \mathrm{K} \times \mathrm{M} \ + b_3$

Here, as before, values for these constants are obtained by using a least squares fit to the output of a program which times the particular operations.

With the computational model constants determined, complete timing estimates are obtained in two sweeps. In the first sweep the amount of computational time to construct the hierarchy of outer ring approximations is estimated. Let n_f denote a finest level of discretization chosen for the timing. (This level should always be greater than the finest level chosen for the actual computation.) Starting at this finest level the time it takes to construct the outer ring approximation is estimated by using the model (1) and information in the array S_n at the finest level. If there is a non-zero value in any element of this array, then a outer ring approximation will be constructed for the particles associated with the corresponding box. The work to construct this outer approximation is estimated using (1) with the value of n being the value in that particular array element of S_n (i.e. the particle count for the box). Each of these estimates is added together to get a complete timing estimate for the construction of the outer approximations at the finest level. One then proceeds to the next coarser level. At this level two computational time estimates are computed. The first estimate is that of the time necessary to construct all of the outer approximations assuming that this particular level is the starting level - the same procedure as for the finest level. The second estimate is that of the time necessary for combining outer ring approximations from the finer level into the outer ring approximations at the current level. In this second estimate, the current level of S_n is swept through and if any element is non-zero, then an outer ring approximation must be constructed for the corresponding box. The time compute this construction is obtained by using the model (2) and determining the number of finer level outer ring approximations which must be used in the construction. (This information can be obtained from the presence of a non-zero value in the next finer level elements of S_n.) One then proceeds to successively coarser levels, forming these same two estimates, until level one is reached. At this point one has information about the time necessary to form outer ring approximations starting from any level as well as the time to create outer ring approximations from outer ring approximations at previous levels. By summing appropriate combinations of these timings, one can compute the time necessary to construct the complete hierarchy of outer ring approximations assuming one starts at any level. As a specific example, assume that 4,000 points are uniformly distributed in a unit square. Then if the finest level chosen is $n_f = 5$ one obtains timing estimates such as those given in the second column of Table 3. Each entry in this column is the

Level	Outer Ring Const. Time (Secs.)	Inner Ring Const. and Eval. Time (Secs.)	Total Time (Secs.)
Direct	0.0	829.0	829.0
2	1.6	331.5	333.1
3	2.6	122.1	124.7
4	6.6	142.2	148.8
5	22.5	509.4	531.9

TABLE 3
Estimated CPU time for a hierarchical element method based on ring approximations. The timings are for 4000 particles distributed uniformly in a unit square.

time it takes to compute the complete hierarchy of outer ring approximations starting from the given level.

In the second sweep estimates of the required time are made for the construction of the hierarchy of inner ring approximations as well as the time for direct local interaction. Starting at level one, the amount of work is estimated which is required for a direct calculation. This is obtained using model (0) and computing n and m by summing the values in the elements of the array S_1 and T_1 associated with level 1. One then proceeds to higher (finer) levels. At each finer level two estimates are computed. The first is an estimate of the time necessary to form an inner ring approximation from nearby outer ring approximations at the same level and the inner ring approximation at the previous coarser level. This time can be estimated by sweeping through the array T_n and determining whether or not an inner expansion will be constructed for any of the boxes in the computational domain. A construction for any box will be necessary if there is a non-zero value in the corresponding element in this array. One then employs the model (2) and determines the number of outer ring approximations and coarse inner ring approximations used in this construction from values in S_n at current level and values in T_n at the previous coarser level. Secondly, an estimate of the time which is necessary if one stops at the current level (i.e. the time to evaluate the inner expansions at this level as well as the local direct computation time) is formed. This time can is computed using the models (0) and (3) and population information for the evaluation points contained in the array T_n. Both of these timing estimates are performed for successively finer and finer levels. The computation is stopped at the finest level n_f. After this computation, we have, for every level, the time it takes to construct the inner ring approximations and the time it takes to use that level for the evaluation of the potential from the inner ring approximations and local direct calculations. By summing the appropriate values of these timings, it is possible to compute the time it takes to construct and evaluate the hierarchical set of inner approximations for each level from the first to the finest. In the example above with 4,000 points, the timings for this second step are presented in the third column of Table 3.

The complete estimate of the time it takes for the whole method consists of adding the amount of work to construct the outer ring approximations and the amount of work to construct and evaluate the inner ring approximations. The level one chooses for the actual computation is the level for which this total is the least. For the example, an estimate of the total time is obtained by summing the elements (across the row) in the second and third column of Table 3. These times are presented in fourth column of that table. One should choose level 3 for this computation.

5. Discussion and Conclusions.

The model problem discussed in this paper has been the computation of the potential induced by N charged particles. In many calculations one is interested in calculating the derivatives of this potential and not the potential itself. There are two ways to use a hierarchical element method to determine the derivatives of the potential. One way is to first create the complete hierarchy of inner ring and outer ring approximations and then, at the last step, compute the derivatives at an evaluation point by summing the derivatives of the potential induced by nearby particles (particles which are not contained in well separated boxes) and the derivative of the potential induced by the inner ring expansion associated with the finest level box the evaluation point resides in. The local interaction is computed by using the analytic derivative of (3) or (4). The derivative of the inner ring expansion is computed using the derivative (either numerical or analytical) of the approximation (13).

The other way to evaluate derivatives consists of computing each derivative by a separate hierarchical element computation. Each derivative of the potential induced by N charged particles is identical to a potential induced by N dipoles. Since the hierarchical element method requires only that the interaction be governed by solutions to Laplace's equation, we can use the method to compute the potential defined by a distribution of N dipoles. Each component of the derivative is therefore obtained by a separate computation and the formulas used for the initial ring construction and local direct evaluation are the analytic expressions for the derivatives of the potential.

In two dimensions, it seems best to use the first technique - compute the potential and then differentiate the result. While there is a loss of one order of accuracy in this procedure, this can be compensated for by increasing the accuracy with which the potential is calculated. (This is accomplished by increasing the number of integration points in the ring approximations, or by using more terms in a multipole approximation.) The extra cost to preserve accuracy is very small compared to having perform completely separate computations for each derivative. The first approach is also likely to be better in three dimensions if one just wants the derivatives of a scalar potential. However, if one is interested in computing the velocity induced by a distribution of vorticity in three dimensions, then the second approach is more efficient. In this case, the velocity field is determined by the curl of a vector potential. To implement the first method, one must determine the three separate components of the vector potential by three different computations and then evaluate the velocity by taking curl of the resulting function. In the second approach, each component of the velocity is determined directly via separate computations and there is no need to differentiate the result. Thus, one saves on both work and accuracy by using the second approach.

One is of course interested in a comparison between the multipole method and the method based on the the use of Poisson's formula. Since the method which uses Poisson's formula is in some sense the discrete transform of the multipole method, one cannot expect there to be any great computational advantages in using one method over the other, i.e. they basically give the same results. The reason for considering using a method based on Poisson's formula is that it is perhaps more convenient to program and use. Since the operations for combining and constructing the hierarchical elements involve only function evaluation, they are easy to formulate and to program. Furthermore, these operations are essentially the same in two and three dimensions and this makes the implementation of the method in three dimensions very easy. A hierarchical element method based on Poisson's formula can also easily treat the problem of determining the potential induced by collections of sources which are more general than point charges. For example, one might be interested in computing the potential induced by collections of segments with some given charge distribution on them. The difficulty with using the multipole method for this computation occurs in the first step. In this step one needs to form a multipole approximations induced by collections of segments. While for point charges, this initial multipole approximation can be analytically determined from the locations of the charges and their strengths, this may not be possible, (or if possible, then rather complicated), for segments. To get an initial ring approximation for such elements one need only to evaluate the potential induced by these distributions at the integration points of the ring approximation.

For either the method presented here, or the original multipole method, one can and should take advantage of the inherent multigrid structure. This inherent structure allows one to take advantage of the variety of techniques which have been formulated for programming multigrid. We have used the techniques for programming on single processor machines, but there is no doubt some benefit to using multigrid techniques for multiprocessor computers. Also, as seen in the third section the timing of the hierarchical element method is very important. From the results of that section it is easy to see why some early experimenters with the method (ourselves included) were not particularly pleased with the hierarchical element method performance. Fortunately, the recursive nature of the method allows for efficient and accurate timing estimates which can be used to select the appropriate level of refinement. The three contributions of the paper, the use of Poisson's formula, the use of multigrid programming strategy, and the use of timing estimates to obtain good level selection, certainly do not exhaust the set of tasks which could be done to improve on the hierarchical element method. Other improvements could come by making the method more adaptive, or improving the speed at which the component operations can be carried out. Such procedures will hopefully lead to even more efficient algorithms.

REFERENCES

[1] C.R. Anderson, 'HELM2 : A Hierarchical Element Method in Two Dimensions', UCLA Report CAM 90-15, Dept. of Mathematics, UCLA, Los Angeles, CA., 1990.

[2] C.R. Anderson, 'HELM3 : A Hierarchical Element Method in Threee Dimensions', UCLA Report CAM 90-16, Dept. of Mathematics, UCLA, Los Angeles, CA., 1990.

[3] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Vol. I, Interscience Publishers, New York, 1953.

[4] L. Greengard and V. Rokhlin, *A Fast Algorithm for Particle Simulations*, J. Comp. Phys., 73(1987), pg. 325.

[5] L. Greengard and V. Rokhlin, *Fast Methods in Three Dimensions* in *Vortex Methods*, C. Anderson and C. Greengard, (Eds.), Lecture Notes in Mathematics, 1360, Springer-Verlag, 1988.

[6] J. Katzenelson, *Computational Structure of the N-Body Problem*, SIAM. J. of Sci. Statist. Comput., 4(1989), pp. 787-815.

[7] A.D. McLaren, *Optimal Numerical Integration on a Sphere*, Math. Comput., 17(1963), pp. 361-383.

[8] Z.P. Nowak, *Panel Clustering Technique for Lifting Potential Flows in The Three Space Dimensions*, in *Panel Methods in Mechanics*, J. Ballman, R. Eppler, W. Hackbusch, (Eds.), Notes in Numerical Fluid Mechanics, 1360, Vieweg-Verlag, Braunschewig/Wiesbaden, 1987.

[9] V. Rokhlin, *Rapid Solutions of Integral Equations of Scattering Theory in Two Dimensions*, J. Comp. Phys. 86(1990), pp. 414-439.

[10] Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Inc. New Jersey, 1971.

[11] K. Stuben and U. Trottenberg, *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications, in Multigrid Methods*, W. Hackbusch and U. Trottenberg, (Eds.), Lecture Notes in Mathematics, 960, Springer-Verlag, 1982.

[12] L. Van Dommelen and E. Rundensteiner, ' *Fast Adaptive Summation of Point Forces in Two-Dimensional Poisson Equations*, J. Comp. Phys., 83(1989), pp. 126-147.