An Overview of Fast Multipole Methods

Alexander T. Ihler

April 23, 2004

Sing, Muse, of the low-rank approximations
Of spherical harmonics and $\mathcal{O}(N)$ computation...

Abstract

We present some background and results from the body of work collectively referred to as *fast multipole methods* (FMM). These comprise a set of techniques for speeding up so-called N-body problems, in which a potential function composed of a sum of pairwise interaction terms from N points is evaluated at an equally large number of locations. We present these methods from the viewpoint of low-rank block matrix approximations, first discussing a heuristic block-matrix approximation method [1], then moving into the analytic expansions which form the basis of both the original and new versions of the fast multipole method [2]. We attempt to provide sufficient background to understand and compute all the relevant results, yet present the material in sufficient generality that it is easy to understand the relationship to similar algorithms such as the fast Gauss transform [3].

1 Introduction

In many problem domains and applications, simulation of physics-based interaction models or other direct methods result in substantial computational overhead. One such class of problems are described by *pairwise interaction* models, in which the function of interest (perhaps a cost or energy function) can be decomposed into sums of pairwise interactions with a set of "source points" x_i:

$$f(y) = \sum_{i=1}^{N} q_i K(y - x_i) \tag{1}$$

where the q_i are scalar values. In particular, these types of problems are found in such wide-ranging fields as physics (computation of electrostatic or gravitational potentials and fields), fluid and molecular dynamic simulations, and density estimation. They are sometimes called "N-body" problems, since they involve N "bodies" of charge, mass, or probability.

We have not specified the dimensionality of the variables y, x_i; in typical applications for physics they are three- (or possibly two-) dimensional values. Although this is not necessarily required it is certainly convenient for gaining intuition. We will use the variables x, y, z to denote vectors of arbitrary dimension, and use $|y|$ to denote the length of y. Otherwise (unless specified) all variables refer to scalar values. In order to provide intuition we will often refer to the x_i as "point charges" and q_i as their associated charge, and give examples couched in terms of computing an electrostatic potential. However, precisely the same equations apply equally well to e.g. the gravitational potential (with the additional requirement that the q_i be positive), and the development for many other problems is quite similar.

Typically the applications of interest involve evaluating $f(\cdot)$ at an equally large number of target locations $\{y_j\}$. For many cases, we desire to evaluate f at the *same* locations x_i (with the self-interaction term omitted):

$$f(x_j) = \sum_{i \neq j} K(x_j - x_i) \tag{2}$$

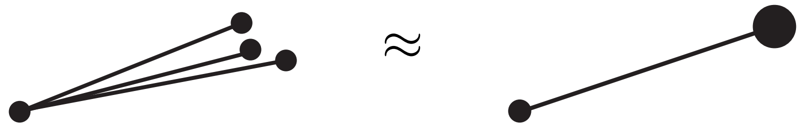

Figure 1: The far-field effect from several point charges can be approximated as a single function capturing their net effect, rather than requiring that each contribution be calculated individually.

Notationally, it is convenient to assume at least that the evaluation locations $\{y_j\}$ and point charge locations $\{x_i\}$ are approximately equal in size. This makes a direct, exact evaluation of the function f parameterized by point charges $\{q_i, x_i\}_{i=1}^N$ at locations $\{y_j\}_{j=1}^N$ nominally a quadratic time operation, $\mathcal{O}(N^2)$. Fast multipole methods (FMM) can be used to approximate this calculation to some pre-specified level of accuracy in only $\mathcal{O}(N)$ time.

2 Far-field approximation

Although the function f may be quite complex in the near-field (when y is close to the set of charges at $\{x_i\}$), if the function K is smooth far from zero, f will be well-behaved in the far-field[1]. In particular, this means that if the location y is sufficiently far from a set of charges at $\{x_i\}$, we may compute the *aggregate* effect of the charges, and not need to resort to computing every interaction.

As an example, consider the electrostatic field due to several point charges examined from very far away. In particular, we assume here that the origin of our coordinate system is near to the point charge locations ($|x_i|$ is small for each i) and far from the location of evaluation ($|y|$ is large). This situation is depicted in Figure 1. We may recall from classical physics that, sufficiently far away and up to first order, we may approximate their net effect by a single point with total charge given by the sum of the q_i. We make this approximation more precise in later sections.

It is generally the case that for any value of y, the sum (2) is dominated by only a few of the x_i's terms (those nearby). The rest of the charges' contributions are sufficiently far that an aggregate approximation is reasonable. By approximating increasingly larger sets as their distance grows, we may greatly reduce the total computational burden. Such "divide-and-conquer" style calculations underlie many computationally efficient algorithms.

3 Matrix Formulation

To construct a precise generalization of such far-field approximations and to gain additional intuition, we first describe the generic problem in terms of matrix multiplication. Let us denote the $N \times N$ matrix of pairwise interactions by M, i.e.

$$M_{ji} = K(y_j - x_i) \tag{3}$$

3.1 Low-rank matrix approximations

We are interested in the computational cost of matrix-vector products Mv for some vector v; in particular, the values of the function f as discussed in Section 1 at the locations $\{y_j\}$ are exactly $M\mathbf{q}$ where $\mathbf{q} = [q_1, \ldots, q_N]$.

If M is an arbitrary matrix, these matrix-vector products are $\mathcal{O}(N^2)$ to compute. However, for certain types of matrices they may be performed more efficiently. For example, this is the case if M is *low-rank*. The *rank* of M (which we denote by r) is the number of linearly independent rows or columns in M [4]. Considering the singular value decomposition of M,

$$M = U'SV \tag{4}$$

[1]Often, the functions K are symmetric and approach zero as distance increases; however, the techniques applied do not require these conditions and we do not impose them. We will, however, require the function to become arbitrarily smooth with increasing distance; this rules out, for example, purely oscillatory kernel functions.

where U and V are size $N \times N$ unitary matrices ($UU' = VV' = I_N$, the $N \times N$ identity matrix) and S is a diagonal matrix of the singular values, the rank of M is the number of non-zero elements of S. Typically, the elements of s are ordered from largest to smallest, $s_1 \geq s_2 \geq \ldots \geq s_r$ (the rest being zero). The limited number of independent rows means that any matrix-vector product may be taken in only $\mathcal{O}(rN)$ time.

Often, a real-world matrix M will be technically full rank; in this event we may instead consider the *effective* rank of M. The effective rank is defined in terms of some numerical tolerance ϵ, as the number of singular values which are greater than $\epsilon \cdot s_1$; we denote this quantity by r_ϵ. While the cost of exact computations scale with the rank r, the cost of approximate computations depends on the effective rank r_ϵ.

Given a matrix M we may construct a low-rank approximation M_ϵ by retaining only those singular values greater than r_ϵ, i.e. $M_\epsilon = U' S_\epsilon V$, where S_ϵ is a thresholded version of S. For such a matrix M_ϵ, one may derive error bounds on an arbitrary matrix-vector product Mv versus the (computationally easier) $M_\epsilon v$. Let $|v|$ be the standard vector norm, $|v| = \sqrt{\sum_{i=1}^{N} (v_i)^2}$; then we have that

$$|Mv - M_\epsilon v| \leq r_\epsilon |v| \tag{5}$$

See e.g. [4] for more detail on low-rank matrix approximation.

3.2 Block approximations

Unfortunately, it is typical that the matrices of interest in real-world N-body problems also have full effective rank. However, they often have *blocks* which are low rank (or low effective rank); this corresponds to the observation that, while y is unlikely to be in the far-field for *all* the x_i, it will be far from many of them. For example, several clustered points x_I (where $I \subset \{1, \ldots, N\}$ may have nearly identical influence on another set of far-away locations y_J. Then, the block M_{JI} is nearly constant (effective rank one), and computations involving it are correspondingly less expensive.

This leads us to consider a natural tradeoff in the computational efficiency of the matrix-vector products. In particular, we may choose the size of the blocks we approximate; we would like each block's size to be large enough that there are few total blocks, but small enough that each block needs only a relatively low rank to achieve the required tolerance level.

In [1], this tradeoff is optimized by heuristically joining blocks of the (pre-specified) matrix M, and approximating the block with a low-rank matrix which meets some error tolerance ϵ. This joining procedure continues until the amount of work required to perform a matrix-vector product is at its local optimum. Algorithmically, this can be done reasonably efficiently:

1. Sort the points so as to put strongly interacting points close together (near the diagonal), and weaker interactions farther apart (off-diagonal). One is free to choose precisely how this is done.

2. Grid the matrix into $k \times k$ blocks (where k is a parameter of the algorithm), and let $l = 0$. Compute rank r_ϵ approximations of each block, and associate with each block an optimal level (initialized to l) at which its products will be computed.

3. Re-grid the matrix with block size $2k$; call these blocks *parents*, and the four sub-blocks enclosed the *children*. Increment $l = l + 1$.

4. For each parent block all of whose children have optimal level $l - 1$,

   (a) Compute a low-rank approximation to the parent block. Note that this can be done more efficiently by first merging pairs horizontally, then vertically (Figure 2); see [1] for more detail.

   (b) Compare the computational burden of using the parent to that of using all four children — if the rank of the parent is less than one-half the summed ranks of the children, set the optimal level of this block to l; otherwise, set it to $l - 1$.

5. Otherwise, set the parent's optimal level to the largest level of its children.

6. Set $k \leftarrow 2k$ and repeat from Step 3.

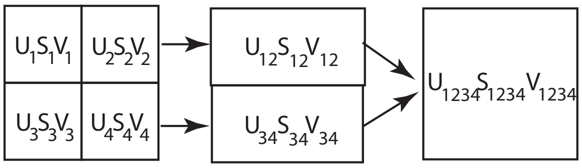

Figure 2: Low-rank approximations for larger blocks may be constructed efficiently by merging the approximations of the constituent smaller blocks; from [1].

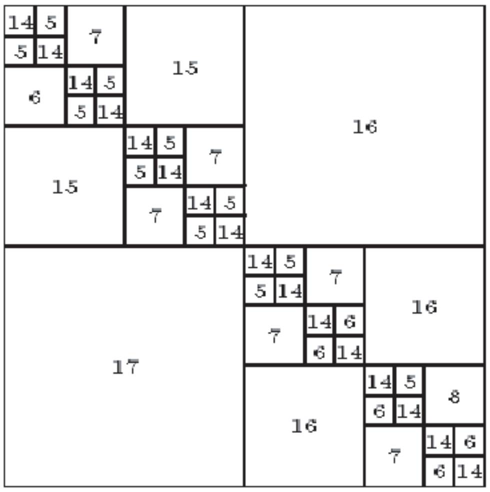

Figure 3: Block-wise low-rank approximation to a 256×256 inductance matrix M from [1]; the number displayed indicates the retained rank of each block to precision $\epsilon = .001$.

This yields a hierarchy of block matrix approximations with bounded error (see Figure 3). While the arbitrary nature of M means that it is difficult to say anything analytic about the algorithm's performance, in practice it has been observed [1] to be slightly super-linear (compared to quadratic for the naive computation).

Of course, none of the presented analysis takes into account the amount of effort required to *find* low-rank approximations to the blocks of M. For the heuristic joining algorithm discussed, this construction cost is about $\mathcal{O}(N^2)$. Thus, if we need to evaluate many matrix-vector products Mv successively for various vectors v, pre-processing M can make these much more efficient. However, if we only need to perform the evaluation a few times, it is no better (and potentially worse) than direct computation.

However, we have as yet done nothing to exploit the known analytic structure of the kernel function K. In fact, we may find a low-rank approximation directly from an analytic series expansion of K, relieving us of the relatively high cost involved in the matrix decomposition step above.

4 Analytic approximation of the kernel

4.1 Degenerate kernel functions

Suppose that our kernel function K is in fact *degenerate*, i.e.

$$K(y-x) = \sum_{k=0}^{p-1} \Phi_k(x)\Psi_k(y) \tag{6}$$

Then the sum (1) becomes

$$\begin{aligned} f(y) &= \sum_{i=1}^{N}\sum_{k=0}^{p-1} \Phi_k(x_i)\Psi_k(y) \\ &= \sum_{k=0}^{p-1}\left(\sum_{i=1}^{N} \Phi_k(x_i)\right)\Psi_k(y) \end{aligned} \tag{7}$$

effectively decoupling the influence of the particle locations x_i and evaluation locations y_j, and allowing all N evaluations at $\{y_j\}$ to be performed in $\mathcal{O}(pN)$ time. Equation (6) indicates a rank-deficiency inherent in the function K, and thus present in the matrix M; specifically, M has rank at most p.

Just as in Section 3, the kernels of interest are not likely to be exactly or approximately degenerate for arbitrary values of x and y. However, once again we expect that the far-field effects of K will be smooth, and thus may be analytically shown to be approximately degenerate so long as x and y are sufficiently far apart. We describe some of these analytic expansions in the next section; we then describe some data structures which may be used to determine appropriate groupings of points, analogous to the choice of which potential indices to group for low-rank block approximation. This leads directly to an $\mathcal{O}(N \log N)$ algorithm; however, with some additional effort we may improve this to $\mathcal{O}(N)$.

4.2 Expansion of the electrostatic potential

We next describe one analytic series expansion of a particular kernel and its associated error residual bounds, corresponding to a multipole approximation to the electrostatic potential due to a number of point charges. The electrostatic potential due to a set of charges q_i located at x_i is given by

$$f(y) = \sum_i q_i \frac{1}{|y-x_i|} \tag{8}$$

We would like to find a summation expansion of the kernel function

$$K(y-x) = \frac{1}{|y-x|} \approx \sum_{k=0}^{\infty} \Phi_k(x)\Psi_k(y) \tag{9}$$

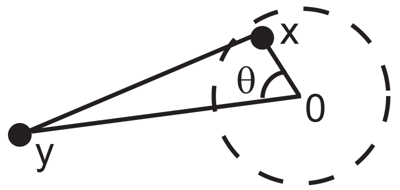

Figure 4: Constructing a series expansion of the function $\frac{1}{|y-x|}$; we consider their positions in terms of the spherical coordinates around some origin 0.

If (for a given set of x_i and y_j) this summation is well-approximated by a small number of terms, we have a situation similar to (7); we will be able to sum the contributions of the x_i separately and decouple the otherwise quadratic cost.

It will turn out to be convenient to perform this approximation in spherical coordinates, essentially because our selected kernel is (trivially) separable in a spherical coordinate frame. Consider, therefore, the situation depicted in Figure 4 — a point charge located at x (taken to be near the origin), whose influence we wish to compute at another location y (taken to be far from the origin). If we denote the angle between the two points by θ, by simple trigonometry we have

$$|y-x|^2 = |x|^2 + |y|^2 - 2|x||y|\cos\theta \tag{10}$$

and thus

$$\frac{1}{|y-x|} = \frac{1}{|y|}\frac{1}{\sqrt{1-2uv+v^2}} \qquad \text{where} \quad u = \cos\theta \quad , \quad v = \frac{|x|}{|y|} \tag{11}$$

We may use this to define a generating function $g(u,v)$ which we expand it into its power series in v

$$g(u,v) = \frac{1}{\sqrt{1-2uv+v^2}} = \sum_{n=0}^{\infty} P_n(u)v^n \tag{12}$$

which defines the *Legendre polynomials* $P_n(\cdot)$; specifically our kernel function is given by

$$\frac{1}{|y-x|} = \frac{1}{|y|}\sum_{n=0}^{\infty} P_n(\cos\theta)\left(\frac{|x|}{|y|}\right)^n \tag{13}$$

In developing this numerical approximation, we have used a coordinate system similar to the one shown in Figure 4 — the charges x_i are near the origin, and evaluation point y is far away. The resulting series (12) is convergent for $|v| < 1$, corresponding to the distance assumption $|x| < |y|$ made at the beginning; thus the series converges very quickly in the far-field. However, sometimes it will be useful to consider the origin to be near y, and the contributing points x_i to be far away. For reasons which will become clear, approximations made in the first regime are called *multipole* expansions, while approximations in the latter will be referred to as *local* expansions. We concentrate initially on multipole expansions (the coordinate system is centered near the charge locations x_i), and discuss later why local expansions are important and how to construct them.

Unfortunately, although Equation (13) is close to our goal of a separable expansion, we have not yet achieved it — the two radii $|x|$ and $|y|$ are coupled by the $\cos\theta$ argument to P_n. However, it will be instructive to linger for a moment on the Legendre polynomial expansion form before constructing a truly separated expansion.

4.2.1 Legendre polynomials

The Legendre polynomials $P_n(\cdot)$ are in fact interesting in their own right. For example, the polynomials form a complete orthogonal basis of functions defined on the interval $[-1, 1]$. They arise in a number of contexts. However, leaving aside these broader contexts, we consider only what we need for the FMM. The

$P_0(u) = 1$
$P_1(u) = u$
$P_2(u) = \frac{1}{2}(3u^2 - 1)$
$P_3(u) = \frac{1}{2}(5u^3 - 3u)$
$P_4(u) = \frac{1}{8}(35u^4 - 30u^2 + 3)$

Figure 5: The first several Legendre polynomials $P_n(u)$.

polynomials themselves may be described analytically in a number of different ways, some of which may be more or less convenient than others. The first few are listed in Figure 5.

Perhaps the most numerically convenient form is given by the recurrence relations,

$$(2n+1)uP_n(u) = (n+1)P_{n+1}(u) + nP_{n-1}(u) \tag{14}$$

which allows the $n+1^{st}$ polynomial to be evaluated in terms of the n^{th} and $n-1^{st}$. Another convenient form is given by Rodrigues' formula,

$$P_n(u) = \frac{1}{2^n n!}\frac{\partial^n}{\partial u^n}(u^2-1)^n \tag{15}$$

The series (13) (and its further expansion into spherical harmonics, described in the next section) is known as a multipole expansion. This nomenclature comes from the physical meaning behind each term in the series — suppose that we have a number of charges q_i arrayed in some small region compared to $|y|$. For sufficiently large $|y|$ this sum is dominated by its first non-zero term. Retaining only the first term of (13) we have that the field is given by

$$f(y) \approx \frac{1}{|y|}\sum_i q_i \tag{16}$$

This is precisely the field due to the q_i's *monopole moment*, the effect of treating the charge distribution defined by the q_i as a single point charge. If the q_i are chosen such that their sum is zero and we examine the second term of the sum, we have

$$f(y) \approx \frac{1}{|y|^2}\sum_i \cos\theta_i \; q_i \; |x|, \tag{17}$$

the field due to the *dipole moment* of the q_i. If, for instance, we have a two-charge symmetric configuration, i.e. $q_1 = -q_2$ and $x_1 = -x_2$, we obtain the classic electrostatic dipole potential. Similarly, if the dipole moment is zero, the third term dominates and is known as the quadrupole moment, and so on.

4.3 Spherical harmonics

While Equation (13) hasn't quite produced the separation of variables we initially hoped, it has come close. As it turns out, we may further expand the Legendre polynomial into its own sum:

$$P_n(\cos\theta) = \sum_{m=-n}^{n} Y_n^{-m}(x)Y_n^m(y) \tag{18}$$

giving the series expansion

$$\frac{1}{|y-x|} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} |x|^n Y_n^{-m}(x)\frac{Y_n^m(y)}{|y|^{n+1}} \tag{19}$$

The functions Y_n^m are called the spherical harmonics; specifically, $Y_n^m(y)\;|y|^{-n-1}$ is the spherical harmonic of degree $-n-1$. Although spherical harmonics may be defined for higher dimensions, they are typically developed for three dimensional space. We will proceed in as general terms as possible, but the reader should note that the definition of Y must be modified in higher dimensions. In $d \leq 3$ dimensions, of course, the formulae hold as presented.

For the same reasons discussed in Section 4.2.1, the spherical harmonics of negative degree are also called multipoles and their coefficients (in this case given by $|x|^n Y_n^{-m}(x)$) are called the *moments* of the expansion. In particular, the moments M_n^m due to a set of charges $\{x_i\}$ is given by

$$M_n^m = \sum_i q_i |x_i|^n Y_n^{-m}(x_i) \tag{20}$$

Note that in fact, the functions Y depend only on the angular portion (in spherical coordinates) of their argument; however we sometimes suppress this independence for convenience in the notation. Interestingly, the spherical harmonic functions Y_n^m themselves form an orthogonal and complete basis for functions on the sphere, subject to "sufficient continuity properties" [5].

The spherical harmonics Y are defined in terms of the associated Legendre functions $P_n^m(u)$. Specifically, if θ_y denotes the polar angle of y, and ϕ_y denotes its azimuthal angle, then

$$Y_n^m(y) = (-1)^m \sqrt{\frac{2n+1}{4\pi} \frac{(n-m)!}{(n+m)!}} P_n^{|m|}(\cos\theta_y) e^{im\phi_y} \tag{21}$$

(where i, when not used in a sub- or super-script, denotes the imaginary number, $i^2 = -1$). Sometimes (as in [2]) the normalization constant $\frac{2n+1}{4\pi}$ is discarded. The associated Legendre functions P_n^m are in turn defined by

$$P_n^m(u) = (1-u^2)^{\frac{m}{2}} \frac{\partial^m}{\partial u^m} P_n(u) \tag{22}$$

Note that for $m = 0$ we recover the Legendre polynomials, $P_n^0(u) = P_n(u)$, and since the highest-order term of P_n is u^n, for $m > n$ we have $P_n^m \equiv 0$. Although it might appear from their definition that m should be positive, the associated Legendre functions for negative m, denoted P_n^{-m}, may be defined by using Rodrigues' formula (15) as the definition of P_n; one may then show that

$$P_n^{-m}(u) = (-1)^m \frac{(n-m)!}{(n+m)!} P_n^m(u) \tag{23}$$

As before, it may be convenient (particularly numerically) to define the associated Legendre functions (and thus the spherical harmonics) in terms of a recurrence relation. Since the associated Legendre functions are doubly indexed, there are in fact a wide variety of recurrences to choose from [5]. A sufficient set is given by the two formulae

$$(2n+1)uP_n^m(u) = (n+m)P_{n-1}^m(u) + (n-m+1)P_{n+1}^m(u) \tag{24}$$

$$(2n+1)(1-u^2)^{\frac{1}{2}} P_n^m(u) = P_{n+1}^{m+1}(u) + P_{n-1}^{m+1}(u) \tag{25}$$

and that $P_0^0(u) = P_0(u) = 1$.

Returning to our primary concern of creating a separable expansion for the kernel function, we see that the expansion (19) is finally beginning to resemble the degenerate form (7); when the distance $|y|$ is far compared to the $|x_i|$ this series will converge quickly and we can hope to neglect all but a few terms. To do so, we need only bound the error incurred by a finite truncation of the series. It can be shown [2] that if the radii $|x_i|$ are bounded by some value $|x|$ and we approximate f using p terms,

$$\hat{f}_p(y) = \sum_{n=0}^{p-1} \sum_{m=-n}^{n} M_n^m \frac{Y_n^m(y)}{|y|^{n+1}} \tag{26}$$

(with moments M_n^m defined as before) then we have that the residual error is bounded by

$$\left| f(y) - \hat{f}_p(y) \right| \leq \frac{Q}{|y| - |x|} \left(\frac{|x|}{|y|} \right)^p \tag{27}$$

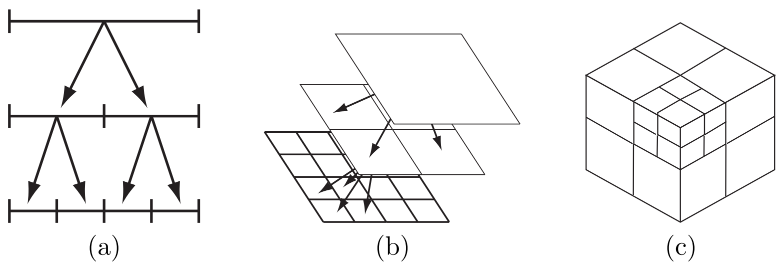

Figure 6: Equal-size hierarchical division of space — a binary tree (a) in one dimension, a quadtree (b) in two dimensions, and an octree (c) in three dimensions.

where $Q = \sum_i |q_i|$ is the total magnitude of charge located at the set of source points x_i.

Although the details of this derivation have been fairly complicated, the general picture should be quite intuitive. Equation (26) is a Taylor series expansion of the function f which is nearly degenerate (has low effective rank, as expressed by (27)). Although the coefficients themselves are less than intuitive, they may be computed using relatively simple formulae and recurrence relations.

This approximation leads directly to an $\mathcal{O}(N \log N)$ algorithm, similar in many ways to our previous low-rank matrix approximation method but with the retained (effective) rank determined analytically from location constraints. Note that (27) demonstrates the same tradeoff discussed earlier for block matrix approximation. Specifically, precision of our approximation $\hat{f}$ may be achieved in two ways — increasing the number of terms retained (p), or increasing the minimum distances involved (reducing $\frac{|x|}{|y|}$). We will consider this tradeoff further shortly. However, before enumerating the algorithm, we must describe some data structure for grouping sets of points which we hope to approximate together.

5 Multi-resolution data structures

As was the case with the block matrix approximation of Section 3, we require some means of deciding which points to group together. The fact that our kernel function becomes increasingly smooth with distance suggests that we should group points by spatial proximity. Similarly, we would like to create groupings at a variety of scales, since we can accurately approximate large sets so long as they are sufficiently far.

One simple way of creating such a hierarchical description of locale is given by successively dividing the space in half along each dimension, giving a tree-structured sequence of increasingly fine partitions. In one dimension, this leads to a binary tree; in two dimensions, a quad-tree (see Figure 6). In three dimensions, it becomes an octree (and in d dimensions, a tree with 2^d children per node).

Of course, not all of these regions will be equally populated. There is clearly no reason to consider further refinement of an empty partition (a region of space in which no points are located); if we stop the refinement of each sub-tree when it contains one or no points we obtain an adaptive binary tree (and quadtree/octree in higher dimensions) [6]. More generally, one may stop when the number is below some fixed value N_0. Similar schemes for creating adaptive tree structures include [7, 8].

Another possibility for adaptive refinement of space is the *k-dimensional tree* (kd-tree for short) [9]. This method again consists of a binary tree decomposition of space, but adaptively chooses both a single dimension along which to divide the space and a location at which to do so, and keep track of bounding boxes or spheres for the contained points. Typically, this requires fewer spatial subdivisions to achieve the same level of clustering than an spatially equally divided tree. However, the cost of constructing such a tree is asymptotically $\mathcal{O}(N \log N)$ (due to a sorting procedure on the points), which means the overall algorithm cannot be less than this. In practice, the asymptotic regime where this cost dominates is well beyond most reasonable data set sizes, but there has been relatively little work in adapting the fast multipole approximations to make use of their advantages.

6 An $\mathcal{O}(N \log N)$ algorithm

At this point, we have all the tools necessary to develop an $\mathcal{O}(N \log N)$ algorithm for computing the interaction potential f at all N locations. The basic idea is to subdivide our domain space using one of the hierarchical descriptions of Section 5, and for all particles which are "sufficiently far" from a given particle, use the multipole expansion to summarize their total interaction.

For analysis of the (typical) binary/quad/octree case, a convenient definition of "sufficiently far" is a binary relationship between pairs of boxes at a particular level, and specified in terms of the length of each box's sides (denoted w). Specifically, sufficient distance is a statement about the ratio of $|x|$ to $|y|$; for all points x_i in a given d-dimensional box with side length w we know that no point is further than $\sqrt{d} \cdot w/2$ from an origin placed at the box's center. Furthermore, if the box containing y is separated by s boxes (i.e. $s = 0$ if y is in a neighboring box, $s = 1$ in the next farther set, and so forth) then $|y|$ is at least $(s + 1/2)w$. Any pair of boxes which are far enough apart (more than s boxes) will be called "well separated", while those which are not are called "near neighbors".

Diagram of above paragraph?

We begin by describing the algorithm under the common assumption of uniformly distributed points, then discuss relaxing this assumption. First, notice that the error bound (27) depends on the ratio of total charge to box size: $\frac{Q}{|y|-|x|}$. If $|y|$ and $|x|$ are both specified in terms of the box length w, this depends on the density of charge within a given box. If the distribution of point charges is relatively uniform, this becomes a constant and we may essentially ignore it.

Next we consider the number of terms required. This depends on the number of boxes we would like to be able to summarize at a given level of the tree hierarchy. A typical choice is that only the box itself and its immediate neighbors comprise the near neighbor set (so that well-separated boxes have $s \geq 1$). In this case (and for $d = 3$ dimensions), it is easy to see that $|x|/|y| \leq \frac{\sqrt{3}}{2}/\frac{3}{2} = 1/\sqrt{3}$. Thus precision ϵ may be achieved using $p \propto -\log \epsilon / \log \sqrt{3}$ (where the proportionality constant depends only on the charge density Q/w). More generally, it will be

$$p \propto -\log \epsilon / \log(\frac{2s+1}{\sqrt{d}}). \tag{28}$$

Typically this overestimates the number of terms required by about a factor of two; the required number of terms for various accuracies cited in [2] seem to be chosen so as to remove this gap. However, for the calculations in this paper we will confine ourselves to the stated, bounding values of p.

The algorithm then operates as follows. We begin with a single box, the region of space in which all points lie. This space is subdivided into 2^d more boxes (e.g. 8 in three dimensions); for each box, we determine which (if any) other boxes may be dealt with immediately using the multipole expansion. After only one subdivision, no boxes are sufficiently far, but after two subdivisions there will be several box pairs which may be summarized. We then proceed recursively by subdividing each box; for each (new, smaller) box b_1 we process any box b_2 whose parent was not processed (the parents of b_1 and b_2 were not well-separated) but which is now well-separated from b_1 (see Figure 7). This recursion proceeds for approximately $\log_{2^d}(N)$ steps, since at this point (by our assumption of uniformity) there will be approximately one point per box and we may simply evaluate the remaining near neighbors directly.

The cost of this algorithm is approximately $N \log N$: to evaluate the expansion of a single box b requires $\mathcal{O}(p^2 N_b)$ operations, where N_b is the number of points inside the box, since we compute the triple sum $\sum_{n=0}^{p} \sum_{m=-n}^{n} \sum_{i \in b} Y_n^m(x_i)$. Computing this for all boxes at a given level requires $\mathcal{O}(p^2 N)$ work; since there are approximately $\log_{2^d}(N)$ levels, we have $\mathcal{O}(p^2 N \log N)$. In fact however, we may do more detailed accounting to evaluate the computational cost.

To estimate the constant in front, we may ask how many boxes which were near neighbors of a given box become well-separated at the next level. Again, denoting by s the required separation in number of boxes, it is easy to see that the number of near neighbors of a box at any level is approximately $(2s+1)^d$. Then, these boxes are subdivided into $2^d(2s+1)^d$ subregions, of which $(2s+1)^d$ are once again not well-separated from a given subregion. At the finest level, we compute the $(2s+1)^d$ interactions directly. Thus the algorithm requires approximately

$$(2^d - 1)(2s+1)^d p^2 N \log_{2^d}(N) + (2s+1)^d N \tag{29}$$

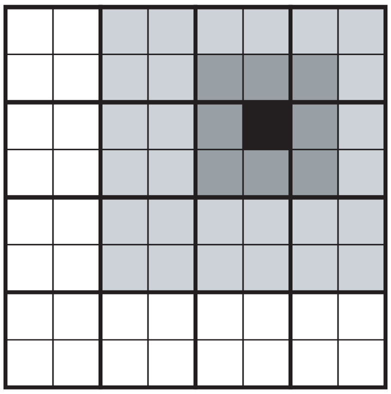

Figure 7: An illustration of the near-neighbors and well-separated boxes in two dimensions. From a given box b's perspective (black), there are $(2s+1)^d = 9$ near-neighbors for $s = 1$, shown in dark gray. At each level, we compute the interaction terms from the children of the near neighbors of b's parent which are not near neighbors of b; these boxes are shown in light gray, and there are $(2^d - 1)(2s+1)^d = 27$ of them. The boxes in white are well-separated from b's parent box, and thus their contribution is calculated at a previous scale.

operations. For $d = 2$ and $s = 1$ (the most commonly considered version) this becomes approximately $27p^2 N \log N$; for $d = 3$ it is $\approx 189p^2 N \log N$.

This analysis has been predicated on the fact that the distribution of points is approximately uniform. If it is *not*, using an adaptive hierarchical structure (such as one which does not continue to refine regions which have fewer than some N_0 points contained) becomes imperative. It also becomes much more difficult to analyze the resulting algorithmic performance. Although [10] demonstrates similar behavior to the improved FMM developed in Section 7 for kd-tree structures, its proof of asymptotic performance appears flawed. In practice, however, the performance of adaptive data structures on non-uniform data is quite similar to the above analysis.

6.1 How can this be improved?

In the Introduction, we promised an $\mathcal{O}(N)$ algorithm; however we have only yet seen an $\mathcal{O}(N \log N)$ one. How may the above algorithm be improved so as to require only $\mathcal{O}(N)$ computations?

The difference in performance can be seen as arising from the fact that each point must be visited at each level of the tree structure, both to create a box's series expansion and evaluate the expansion from each well separated box. If we could instead visit only every *box* at each level, caching sufficient information about the expansion, we would require only $\mathcal{O}(1 + 2^d + 4^d + \ldots + N) = \mathcal{O}(N)$ operations.

The dependence on the actual point locations enters in two places. The first is in computing the source expansion — naively, we compute moments for each box b by evaluating a sum N_b terms. We may avoid this by designing a means of converting the finer scale's moment expansions to an expansion at the next coarser scale. The second dependence is in evaluating at the target locations. Evaluating at a target location depends on the angle from the expansion's origin to the target. We would like to accumulate terms from several source boxes and combine them so that they may be evaluated at the target only once. This requires two things — a translation of several expansions' origins to a common point (enabling them to be summed), typically the target box center, and a translation of an expansion about the target box center to expansions about each of its subregions (enabling expansions from different levels to be combined easily). In the next section, we present three theorems which enable this decoupling, and show that it results in an $\mathcal{O}(N)$ algorithm.

7 An $\mathcal{O}(N)$ algorithm

7.1 Local translation operations

As discussed, in order to develop an $\mathcal{O}(N)$ fast multipole algorithm, we require three things:

1. A means of combining fine-scale multipole expansions into a single coarse-scale expansion. This will be used to compute the multipole expansions of each box in the tree using a single fine-to-coarse pass, and takes the form of a translation from child box centers to parent box centers.
2. A means of combining several multipole expansions into a single expansion about a local origin in the target box. This again takes the form of a translation, but this time between source boxes and target boxes at the same scale.
3. A means of translating the target box's local series expansion to an origin within each of the box's subdivisions at the next level of the tree. This will be used to combine the approximations at all levels in a coarse-to-fine pass.

Each of these is addressed with a corresponding theorem.

7.1.1 Translation of the multipole expansion

This step allows us to combine several multipole expansions about different origins at one scale to a single expansion at the next coarser scale (the boxes' parent), by translating each of them to a common origin. Suppose that we have a multipole expansion about the origin, and we shift the expansion to be located about some point z, for example the center of the parent box (see Figure 8). Then, our original multipole expansion looks like

$$\hat{f} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{M_n^m}{|y|^{n+1}} Y_n^m(y) \tag{30}$$

This may be expanded into the coarser scale form

$$\hat{f}^+(y) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{\tilde{M}_n^m}{|y-z|^{n+1}} Y_n^m(y-z) \tag{31}$$

where

$$\tilde{M}_n^m = \sum_{j=0}^{n} \sum_{k=-j}^{j} M_{n-j}^{m-k} \frac{i^{|m|}}{i^{|k|} i^{|m-k|}} \frac{A_j^k A_{n-j}^{m-k}}{A_n^m} |z|^j Y_j^k(z) \tag{32}$$

where the constant A_n^m is defined by

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}. \tag{33}$$

However, our error bounds are loosened slightly by the coarser expansion, from the original error bound

$$\left| f(y) - \hat{f}_p(y) \right| \leq \frac{Q}{|y| - |x|} \left(\frac{|x|}{|y|} \right)^p$$

to

$$\left| f(y) - \hat{f}_p^+(y) \right| \leq \frac{Q}{|y-z| - |x| - |z|} \left(\frac{|x| + |z|}{|y-z|} \right)^p \tag{34}$$

Note also that, since there are p^2 moment terms $\tilde{M}_n^m$, each of which is a linear function of the p^2 original moments M_n^m, this operation takes about p^4 operations.

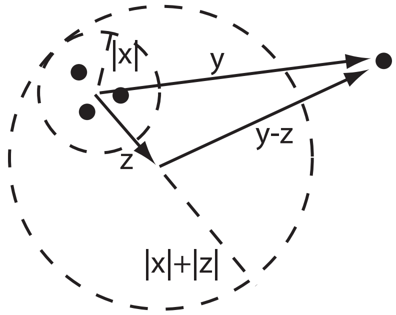

Figure 8: A multipole expansion around the source points (small circle) is translated to an expansion about the point z, in order to fuse several fine-scale expansions into a single coarse-scale one.

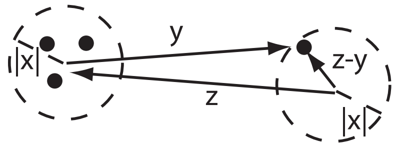

Figure 9: A multipole expansion around the source points (left) is translated to a local expansion about the point $-z$ (in the original coordinate system), in order to fuse several multipole expansions into a single local one.

7.1.2 Conversion to a local expansion

The next step is to convert each source box's multipole expansion at a particular tree level into a single expansion about the target box center. Given a multipole expansion of the form (30) which we intend to evaluate at some point y, it may be converted to a local expansion about the point $-z$ (see Figure 9). We assume that $|z|$ is large compared to the bounding sphere of radius $|x|$ about the sources x_i, specifically that there exists some constant $c > 1$ such that $|z| > (c+1)|x|$, and also that $|z-y| < |x|$ (that y is contained in an equivalent-sized bounding sphere about z).

This local expansion has the form

$$\hat{f}^*(y) = \sum_n \sum_{m=-n}^{n} L_n^m |z-y|^n Y_n^m(z-y) \tag{35}$$

where

$$L_n^m = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{(-1)^j} \frac{i^{|m-k|}}{i^{|k|} i^{|m|}} \frac{A_j^k A_n^m}{A_{j-n}^{k-m}} \frac{Y_{j+n}^{k-m}(z)}{|z|^{j+n+1}} \tag{36}$$

and A_j^k is defined as before.

This is not quite the result we need — specifically, it does not tell us what additional error is introduced in this transformation by the fact that our moments M_j^k have been truncated (retaining p^2 terms). According to [2] it is "straightforward but tedious" to derive a precise bound on the error. One may think of this as a slight accumulation of error; using only p^2 moments causes slight errors in even the retained values of L_n^m. Luckily only this step introduces additional error, since the upward and downward translations of the multipole expansion are exact, and this step occurs only once for each contributing box.

Furthermore, finite truncation of this series to $p \geq 1$ terms conforms to the error bound

$$\left| f(y) - \hat{f}_p^*(y) \right| \leq \frac{Q}{(c-1)|x|} \left(\frac{1}{c} \right)^p . \tag{37}$$

As with the multipole translation, the construction of a truncated local expansion requires p^4 operations.

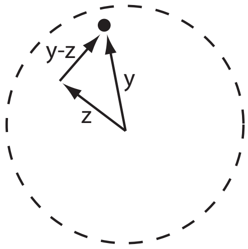

Figure 10: A local expansion create for the coarse scale region can be converted into local expansions at each finer scale child by translating the expansions' origins to the centers of the child regions.

Notice that if this expression is used for converting expansions from well-separated boxes (with number of separating boxes $s \geq 1$ and box width w), then $|z| \geq 2w$ and $|x| \leq \sqrt{d} \cdot w/2$, and thus we know that

$$c \geq \frac{4}{\sqrt{d}} - 1 \tag{38}$$

Unfortunately this is a considerably weaker rate of convergence (in p) than we obtained for the original $N \log N$ algorithm. For example, in $d = 3$ dimensions our original error bound was slightly faster than $(.6)^p$, but it is now about $(.75)^p$. One possible solution [11] is to require further separation between approximated boxes, but (as we shall see) this may be ineffective since it also impacts the number of boxes which must be processed at each level.

7.1.3 Translation of a local expansion

Finally, we would like to shift the local expansion at a box center to a local expansion at the center of each of its children (see Figure 10). Let a finite local expansion be given by

$$\hat{f}_p^*(y) = \sum_{n=0}^{p-1} \sum_{m=-n}^{n} L_n^m |y|^n Y_n^m(y) \tag{39}$$

Then we may equivalently write

$$\hat{f}_p^*(y) = \sum_{n=0}^{p-1} \sum_{m=-n}^{n} \tilde{L}_n^m |y-z|^n Y_n^m(y-z) \tag{40}$$

where

$$\tilde{L}_n^m = \sum_{j=n}^{p-1} \sum_{k=-j}^{j} \frac{L_j^k}{(-1)^{j+n}} \frac{i^{|k|}}{i^{|k-m|} i^{|m|}} \frac{A_{j-n}^{k-m} A_n^m}{A_j^k} Y_{j-n}^{k-m}(-z)|z|^{j-n} \tag{41}$$

where A_j^k is defined as before. This entails no additional approximation (merely a translation of origin), and again requires p^4 operations.

7.2 The (original) fast multipole algorithm

We are now ready to describe the original Fast Multipole Method (FMM). As outlined previously, this will proceed as follows:

1. Form multipole expansions (moments) at the tree's finest scale
2. Merge expansions to form the expansions at the next coarser scale until the coarsest level is reached
3. Starting at the coarsest level, for each (target) region convert the multipole expansions at any well-separated (source) regions into a local expansion around the target center.

4. For each region, translate the local expansion to the center of each of its 2^d children.
5. Go to (3) until the finest scale is reached; then evaluate the series expansion at each target location and compute the remaining near neighbor interactions directly.

Once again, let us assume uniformity of the source and target points, and construct a tree hierarchy with $\log_{2^d}(N/N_0)$ levels (so that each finest-scale box has approximately N_0 points each). This means that there are approximately N/N_0 total boxes in the hierarchy. Analyzing the costs of each of the above steps, we see that the first requires pN computations, the second and fourth each require at most $p^4(N/N_0)$ (p^4 for each node in the tree), the third $p^4(2^d-1)(2s+1)^d(N/N_0)$, and the last $(2s+1)^d NN_0 + pN$ computations (the remaining pairwise interactions plus the expansion's sum at each target). Thus we have a total number of computations given by

$$p^4((2^d-1)(2s+1)^d+2)(N/N_0) + 2pN + (2s+1)^d NN_0 \quad (42)$$

If, as in [2], we choose $N_0 \approx 2p^2$ and take $d = 3$ dimensions and $s = 1$ as our neighborhood region, this becomes $\approx 150p^2N$.

Notice that the increase in required number of terms p from Section 7.1.2 is unfortunate, as the "constant" in front of our $\mathcal{O}(N)$ algorithm depends on p^2. To give some intuition, for $\epsilon = 10^{-4}$ and unit charge density, we need $p = 17$ terms for the $\mathcal{O}(N \log N)$ algorithm but $p = 32$ terms for the $\mathcal{O}(N)$ one. However, the $\mathcal{O}(N)$ version is almost always preferable, since for this example, we can compute the break-even point between the two by $189(17)^2 N \log_8 N \approx 150(32)^2 N$, which is attained around $N \approx 400$. In this regime the direct, quadratic computation is actually more efficient; we are primarily interested in much higher values of N.

Furthermore, consider increasing the required separation distance s to, say, $s = 2$ as proposed in [11]. Then for $\epsilon = 10^{-4}$ we requires only about $p = 11$ terms, but the constant in front increases, making our cost $689p^2N$. For this choice of ϵ this gives only about a factor of 2 improvement. In order to be practical for the three-dimensional problems of interest, we require something more.

8 Improvements in three dimensions

Unfortunately, although the asymptotic performance is $\mathcal{O}(N)$, the high value of the constant factor means that the standard FMM becomes much slower with increasing dimension. In particular, three dimensional problems are particularly common, yet the breakpoint for FMM versus direct computation in three dimensions is quite high. For example, if $\epsilon = 10^{-4}$ we have $p \approx 32$ and the cost is $\approx 150 * p^2 N$, so the break-even point versus direct N^2 computation is around N=150,000.

One possible way to reduce this cost is by applying coordinate system rotations to translate the multipole expansion in a source box to the center of a target box. Naively, the multipole to local translation cost p^4 operations; by rotational methods it can be performed in only $3p^3$ operations. For details on how this may be done, see e.g. [2]. However, this is still not a sufficient improvement to make the method truly practical, and we omit details.

8.1 Exponential plane waves

Recent work by Greengard and Rokhlin [2] uses somewhat more complicated mathematical apparatus and improves the FMM's performance for reasonable values of N. The essential observation is that (in three dimensions) the multipole expansion may be converted into six directional plane-wave expansions (one in each cartesian direction, $\pm x, \pm y, \pm z$). These plane-wave expansions can be translated very easily, and may then be converted into local expansions as before. Although mathematically more difficult, this results in a computationally lower complexity procedure.

8.1.1 Integral expansion

We require a different expansion for each of the six cartesian directions. For example, letting r be a vector, the positive z direction can be derived using the integral identity [2]

$$\frac{1}{|r|} = (r_x^2 + r_y^2 + r_z^2)^{-\frac{1}{2}} = \int_0^\infty e^{-\lambda r_z} J_0(\lambda\sqrt{r_x^2 + r_y^2})d\lambda \quad (43)$$

where J_0 is a Bessel function of integral order zero. This Bessel function has a number of nice integral representations [5], for example

$$J_0(u) = \frac{1}{2\pi} \int_0^{2\pi} \cos(u \ \sin \alpha) d\alpha \tag{44}$$

or

$$J(\lambda \sqrt{r_x^2 + r_y^2}) = \frac{1}{2\pi} \int_0^{2\pi} e^{i\lambda(r_x \cos \alpha + r_y \sin \alpha)} d\alpha \tag{45}$$

Clearly, these integrals may be estimated using two summations, such as

$$\frac{1}{|r|} \approx \sum_k w_k e^{-\lambda_k r_z} J_0(\lambda_k \sqrt{r_x^2 + r_y^2}) \tag{46}$$

$$J_0(\lambda_k \sqrt{r_x^2 + r_y^2}) = \frac{1}{2\pi} \sum_j \omega_{jk} e^{i\lambda_k (r_x \cos \alpha_{jk} + r_y \sin \alpha_{jk})} \tag{47}$$

For any finite number of terms in each sum, the problem of determining a good set of coefficients $w_k, \lambda_k, \omega_{jk}, \alpha_{jk}$ falls into the category of quadrature approximation.

8.1.2 Quadrature approximation

Quadrature is the well-studied field of approximating the definite integral of an unknown function by a weighted sum of n evaluations at pre-specified locations. While a full explanation of quadrature approximations goes beyond the scope of this document, we briefly describe some of the relevant background for applying quadrature methods to the double integral form (43).

Quadrature approximations [5] come in many forms. The most straightforward involve evaluating the integrand at a number of equally-spaced locations. Equally weighting these evaluations is often called "trapezoidal" quadrature, since it approximates the integral by the area under a linear interpolation. The error in this estimate is bounded by a constant factor times the second derivative of the integrand (as seems intuitive for such an approximation).

By optimizing the n weights, one may create a better fit and make the quadrature estimate exact for low-degree polynomial functions. Even more sophisticated methods optimize both the weights and evaluation locations in the sum, $2n$ parameters in all, selecting the best possible interpolating polynomial. Collectively, these $2n$ parameter methods are known as Gaussian quadrature, and can be shown to be exact for all polynomials up to degree $2n - 1$. Their estimate error is at most proportional to the $2n^{th}$ derivative of the integrand.

Of course, we have a double integration problem, and thus must worry about the interaction between the two sources of error. Luckily, the bounded nature of the Bessel function J_0 means that our errors cannot interfere too catastrophically. Specifically, if we have a function of two variables $f(u, v)$ and both approximations (individually) have errors bounded by ϵ:

$$\int f(u, v) du - \sum_{i=1}^{n_u} w_i^u f(u_i, v) < \epsilon \qquad\qquad \int f(u, v) dt - \sum_{i=1}^{n_v} w_i^v f(u, v_i) < \epsilon \tag{48}$$

and at least one function's evaluates have bounded absolute sum, e.g. $\sum_{i=1}^{n} |f(u, v_i)| < \eta$, then the total error in the double quadrature approximation is bounded by

$$\int \int f(u, v) du dv - \sum_{i=1}^{n_u} \sum_{j=1}^{n_v} w_i^u w_j^v f(u_i, v_j) < \epsilon(1 + \eta). \tag{49}$$

If we then choose the number of terms in our quadrature approximation to be high enough, we may achieve any level of desired accuracy. We are free to choose the exact *nature* of this quadrature formula as we please; for example, one might choose trapezoidal quadrature for the inner integral (44) (involving α), since it is on a finite range; all weights are equal and locations equally spaced in $[0, 2\pi]$. One may easily bound the second derivative of J_0 to obtain the required number of terms.

As for the outer integral (43) (involving λ), an appealing choice is Gauss-Laguerre quadrature. The locations are then given by the n roots of the n^{th} Laguerre polynomial [5]

$$L_n(\lambda) = \sum_{s=0}^{n} (-1)^{n-s} \frac{n!}{s!} \frac{\lambda^{n-s}}{(n-s)!(n-s)!} \tag{50}$$

and weighting by w_i where

$$w_i = \frac{\lambda_i}{(n+1)^2 \left(L_{n+1}(\lambda_i)\right)^2}. \tag{51}$$

Again, the error is proportional to the $2n^{th}$ derivative of the integrand.

Although we list these options as reasonable possibilities, they may not be the optimal choices. Indeed, recently methods have been proposed by [12, 13] which numerically optimize the weights and locations for a pre-specified class of possible functions in the integrand. This may reap considerable benefits in the FMM, since we know a considerable amount about each function. Specifically, we know not only the analytic structure of the function, but also about the possible range of its arguments (since in the FMM, the boxes being considered at any point are neither extremely close nor far). In [13], a numerical search is proposed and demonstrated by optimizing coefficients to minimize potential error (whose maximum, they show, may be evaluated for arguments in the pre-specified region). While this involves a procedurally complex search which must be repeated if the desired tolerance level changes, it results in an extremely efficient quadrature formulation for the specific problem under consideration. Tables for several reasonable choices of ϵ are given in [2]. Note, however, that because these estimates are obtained by assuming a particular argument range (the values of which are appropriate for a unit-length cube), we will need to rescale each box before converting it to a plane wave expansion, then rescale again after converting back to a local expansion.

Finally, the exact number of terms required in the inner integral's trapezoidal approximation, denoted m_k, depends on the *value* of the particular λ_k in the outer integral's quadrature sum (46). Given m_k, of course, the locations α_{jk} of (47) are simply $\frac{2\pi j}{m_k}$ and the weights $\omega_{jk} \equiv 1$.

The double integral expressing r^{-1} is thus approximated by the sum

$$\sum_{k=1}^{s} \sum_{j=1}^{m_k} \frac{w_k}{m_k} e^{-\lambda_k r_z} e^{-i\lambda_k (r_x \cos\alpha_{jk} + r_y \sin\alpha_{jk})} \tag{52}$$

resulting in an overall approximation to f (in the $+z$ direction)

$$\hat{f}^{+z} = \sum_{k=1}^{s} \sum_{j=1}^{m_k} W_{jk} e^{-\lambda_k r_z} e^{-i\lambda_k (r_x \cos\alpha_{jk} + r_y \sin\alpha_{jk})} \tag{53}$$

where the W_{jk} will be defined shortly in terms of the multipole expansion.

8.1.3 Exponential translation operations

At this point it might appear that we have merely created more work for ourselves; however, the plane wave expansion has one extremely nice property — that translation (step 3 of the original FMM, Section 7.1.2) becomes a diagonal operation and thus may be performed in only $\mathcal{O}(p^2)$ time (rather than $\mathcal{O}(p^4)$ for the original algorithm or $\mathcal{O}(p^3)$ by coordinate system rotation).

This requires three steps:

1. Conversion of a multipole expansion to (six) plane-wave expansions
2. Translation of a plane-wave expansion
3. Conversion of a plane-wave expansion to a local expansion

The first is accomplished by expressing the spherical harmonics as derivatives of r^{-1}; this gives the result that the coefficients of the exponential plane wave may be calculated by first rescaling the box to unit volume (if the box has side length b, we set $M_n^m \leftarrow M_n^m / b^{n+1}$). We may then apply the transformation

$$W_{jk} = \frac{w_k}{m_k} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{(-i)^{|m|} e^{im\alpha_j} M_n^m (\lambda_k)^n}{\sqrt{(n-m)!(n+m)!}} \tag{54}$$

As was the case in (36) for the original algorithm, although this involves an infinite sum its truncation to the p^2 terms typically retained by the multipole expansion introduces only a small additional error to the computation.

The translation step (to any sufficiently distant box in the $+z$ direction) is quite simple: to translate the coefficients W_{jk} for an expansion at the origin to coefficients W_{jk}^d for an expansion centered at $d = (d_x, d_y, d_z)$ we again rescale by the box size ($d \leftarrow d/b$), then simply apply the diagonal operation

$$W_{jk}^d = W_{jk} e^{-\lambda_k d_z} e^{-i\lambda_k (d_x \cos\alpha_{jk} + d_y \sin\alpha_{jk})} \tag{55}$$

Finally, converting the plane wave expansions back into local expansion coefficients L_n^m at the target box can be done by the following transformation:

$$L_n^m = \frac{(-i)^{|m|}}{\sqrt{(n-m)!(n+m)!}} \sum_{k=1}^{s} (-\lambda_k)^n \sum_{j=1}^{m_k} W_{jk} e^{im\alpha_j} \tag{56}$$

We may then rescale back to our original box size b by setting $L_n^m \leftarrow L_n^m b^n$.

In [2] it is asserted that $s \approx p$ and that $m_k \approx p$ on average, and thus the total cost of performing the multipole to plane wave and plane-wave to local conversions is approximately $3p^3$. Computationally, this is a small additional price, and the translation to every well-separated region is correspondingly cheaper, costing only $p^2(2^d - 1)(2s + 1)^d = 189p^2$. Choosing the minimum number of particles per box to be $N_0 \approx 2p$, our operation count becomes $\approx 150pN + 5p^2 N$. For $\epsilon \approx 10^{-4}$ we have a break-even point of $N \approx 150 * 32 + 5 * (32)^2 = 10^5$, more than an order of magnitude better than the original FMM.

As it turns out, a few more tricks including close examination of symmetry relationships can speed things up even further. However, we do not explore these options; for a short list of the possibilities see [2].

9 Gaussian kernel expansion (the fast Gauss transform)

While most of the previous discussion has been in terms of the kernel $K(z) = \frac{1}{|z|}$, an almost identical development may be performed for other kernel functions using their series expansions. For example, the fast Gauss transform (FGT) [3] can be used for efficient simulation of diffusion models and more generally evaluation of Gaussian sum models. Using the same toolkit, it is relatively easy to derive. Suppose that our kernel function is described by a Gaussian distribution,

$$K(z) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{|z|^2}{2\sigma^2}} \tag{57}$$

We are aided considerably in this case by the fact that the Gaussian distribution (57) is separable in cartesian coordinates, since if z is d-dimensional, $z = (z_1, \ldots, z_d)$, we have $K(z) = \prod_i K_i(z_i)$. Then, a separable kernel in d dimensions may be constructed by the product of separable kernels in each individual dimension:

$$\begin{aligned} K(y - x) &= \prod_{i=1}^{d} \left(\sum_{k=0}^{p-1} \Phi_k(x^i) \Psi_k(y^i) \right) \\ &= \sum_{k_1=0}^{p-1} \cdots \sum_{k_d=0}^{p-1} \left(\prod_{i=1}^{d} \Phi_{k_i}(x^i) \Psi_{k_i}(y^i) \right) \end{aligned}$$

Note, however, that for a given rank p the above sum has p^d terms, exponential in the problem's dimension. Although we will see that the residual error decays very rapidly in p, this is still the main computational

$$
\begin{aligned}
H_0(u) &= 1 \\
H_1(u) &= 2u \\
H_2(u) &= 4u^2 - 2 \\
H_3(u) &= 8u^3 - 12u \\
H_4(u) &= 16u^4 - 48u^2 + 12
\end{aligned}
$$

Figure 11: The first several Hermite polynomials $H_n(u)$.

limitation of the fast Gauss transform for high dimension. Once again, exponential plane waves may be used to speed up the process somewhat [3]; but here will we simply cover the original FGT.

Therefore we may consider only the one-dimensional case, for which $(y-x)^2 = y^2 + x^2 - 2xy$. In parallel with Equation (12), define the *Hermite polynomials* H_n by their generating function

$$
g(u, v) = e^{-v^2+2uv} = \sum_{n=0}^{\infty} H_n(u) \frac{v^n}{n!} \tag{58}
$$

Then clearly the Gaussian kernel is given by $u = \frac{x}{\sqrt{2\sigma^2}}$, $v = \frac{y}{\sqrt{2\sigma^2}}$, and

$$
K(y-x) = \frac{1}{Z_\sigma} \sum_{n=0}^{\infty} H_n(u) e^{-u^2} \frac{v^n}{n!} \tag{59}
$$

where Z_σ is the Gaussian normalization constant. Again, it is usually more convenient to express the Hermite polynomials in terms of a recurrence relation,

$$
H_{n+1}(z) = 2zH_n(z) - 2nH_{n-1}(z) \tag{60}
$$

with $H_0(z) = 1$, $H_1(z) = 2z$, and so on; the first several Hermite polynomials are listed in Figure 11. Assuming the box size is sufficiently small ($\leq \sqrt{2}\sigma$), the series' finite truncation error is bounded by [3]

$$
\left| f(y) - \sum_{k=0}^{p-1} \Phi_i(y) \Psi_i(x_i) \right| \leq 2.75^d \; Q \; \left(\frac{1}{p!}\right)^{d/2} \left(\frac{1}{2}\right)^{(p+1)d/2} \tag{61}
$$

where (as previously) $Q = \sum |q_i|$ and d is the dimension. This error bound decreases much more rapidly in p than the corresponding bound in the original FMM, but does not make use of the separation distance between source and target boxes, perhaps indicating that even tighter bounds might be derived.

In principle, there is nothing to stop one from deriving similar algorithms for any other kernel functions using a series expansion (if known); however, the FMM and FGT are each widely applicable and thus the best studied. For another example, see e.g. [14], which extends the FMM to kernels of the form

$$
K(y-x) = \frac{e^{-\lambda|y-x|}}{|y-x|} \tag{62}
$$

called "screened" Coulombic potentials.

10 Conclusions

We have given an overview of the fast multipole method and related algorithms, which are used for approximate computation for otherwise quadratic complexity problems. The key to these algorithms lies in creating low-rank block approximations to the $N \times N$ matrix of interactions. This may be done directly, but not without incurring the quadratic cost; thus it is only useful if the matrix is to be reused many times.

For many analytic interaction potentials, however, the function itself can be shown to have low effective rank under certain distance constraints. Exploiting these low-rank forms results in an $\mathcal{O}(N)$ algorithm, where the constant depends on the desired accuracy.

Unfortunately, the size of this constant is quite important and may dominate even for the large data sets of interest. Reducing its effect can be quite involved. Directional exponential expansions can lower the cost considerably, due to the simplicity of translating their series from source to destination regions (the main computational bottleneck of the original FMM).

Although we have discussed the algorithms mainly in terms of the Coloumbic potential function r^{-1}, the same ideas may be applied to other potentials as well. Gaussian sums, for instance, may be evaluated using the fast Gauss transform, which is developed from precisely the same principles. Other functions may be readily used given a series expansion and a bound on its finite truncation error.

References

[1] Sharad Kapur and Jinsong Zhao. A fast method of moments solver for efficient parameter extraction of MCMs. In *Design Automation Conference*, pages 141–146, 1997.

[2] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the laplace equation in three dimensions. *Acta Numerica 6*, pages 229–269, 1997.

[3] L. Greengard and X. Sun. A new version of the fast Gauss transform. *Documenta Mathematica*, Extra Volume ICM(III):575–584, 1998.

[4] G. Strang. *Linear Algebra and its Application, 3rd. Ed.* Academic Press, 1985.

[5] George Arfken. *Mathematical Methods for Physicists.* Academic Press, Boston, 3 edition, 1985.

[6] J. Barnes and P. Hut. A Hierarchical $O(NlogN)$ Force Calculation Algorithm. *Nature*, 324:446–449, 1986.

[7] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal of Scientific and Statistical Computing*, 9(4), 1988. Yale University Technical Report, YALEU/DCS/RR-496 (1986).

[8] Leon van Dommelen and Elke A. Rundensteiner. Fast, adaptive summation of point forces in the two-dimensional poisson equation. *J. Comput. Phys.*, 83(1):126–147, 1989.

[9] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18, 1975.

[10] A. G. Gray and A. W. Moore. Very fast multivariate kernel density estimation via computational geometry. In *Joint Stat. Meeting*, 2003.

[11] L. Greengard and V. Rokhlin. The rapid evaluation of potential fields in three dimensions. In C. Greengard (Eds.) C. Anderson, editor, *Vortex Methods*, volume 1360 of *Lecture Notes in Mathematics*, pages 121–? Springer-Verlag, Berlin, 1988.

[12] J. Ma, V. Rokhlin, and S. Wandzura. Generalized gaussian quadrature rules for systems of arbitrary functions. *SIAM J. Numer. Anal.*, 33(3):971–996, 1996.

[13] N. Yarvin and V. Rokhlin. Generalized gaussian quadratures and singular value decompositions of integral operators. *SIAM J. Sci. Comput.*, 20(2):699–718, 1998.

[14] J. Huang and L. Greengard. A new version of the fast multipole method for screened coulomb interactions in three dimensions. *J. Comput. Physics*, 180(2):642–658, 2002.