JOURNAL OF COMPUTATIONAL PHYSICS **126,** 274–285 (1996)
ARTICLE NO. 0137

A Multipole-Based Algorithm for Efficient Calculation of Forces and Potentials in Macroscopic Periodic Assemblies of Particles

CHRISTOPHE G. LAMBERT,* THOMAS A. DARDEN,† AND JOHN A. BOARD JR.‡

*Department of Computer Science, Duke University, Durham, North Carolina 27708, †National Institute of Environmental Health Sciences, MD A3-06, P.O. Box 12233, RTP, North Carolina 27709, and ‡Department of Electrical Engineering, Duke University, Durham, North Carolina 27708

Received January 24, 1995; revised October 27, 1995

A new and efficient algorithm based on multipole techniques is presented which calculates the electrostatic forces and potentials in macroscopic periodic assemblies of particles. The fast multipole algorithm (FMA) can be used to compute forces within the n-particle unit cell in $O(n)$ time. For the cubic lattice, forces due to a $3^k \times 3^k \times 3^k$ lattice of images of the unit cell, containing 3^{3k} n particles, can be computed in $O(nk^2 + k^3 \log k)$ time to arbitrary precision. The algorithm was readily added onto an existing FMA implementation, and computational results are presented. Accurate electrostatic computations were done on a $3^8 \times 3^8 \times 3^8$ region of 100000-particle unit cells, giving a volume of 28 quadrillion particles at less than a twofold cost over computing the forces and potentials in the unit cell alone. In practice, a $k = 4 \cdots 6$ simulation approximates the true infinite lattice Ewald sum forces (including the shape-dependent dipole correction) to high accuracy, taking 25–30 % more time to compute than only the unit cell. The method extends to noncubic unit cell shapes, and noncubic macroscopic shapes. Simple code modifications allowed computation of forces within macroscopic spheres and ellipsoids, and within near-infinite square, circular, and elliptical surfaces formed of unit cubes replicated along two of the three axes. In addition to efficient periodic simulations, the method provides a powerful tool to study limiting behavior of various finite crystal shapes, as well as surface phenomena in molecular dynamics simulations.

1. INTRODUCTION AND MOTIVATION

Much of molecular dynamics today concentrates on biomolecules. One particular challenge is the computation of electrostatic forces with included solvent effects. Typically the experimentalist is interested in the dynamics of a solvated protein molecule or DNA strand. Surrounding water molecules must be simulated to properly deal with shielding of Coulomb interactions within the biomolecule. However, neglecting the larger aqueous environment outside the simulation box results in poor simulation of water molecules at the box boundaries (i.e., there is a nonphysical vacuum there), ultimately compromising the results of the simulation. Periodic boundary conditions are perhaps the best approach to address this problem. The atoms within the unit cell feel forces due to near neighbors within the cell, as well as forces due to a lattice of periodic copies of the unit cell.

Early work on simulation of particle systems with periodic boundary conditions was developed by condensed matter physicists in their simulation of materials with repeated lattice-like properties. In crystals and other solids, a unit cell of particular symmetry is the building block of the material. Once the unit cell is specified, the structure of the solid can be described in terms of copies of these building blocks except for small deviations due to atomic vibrations. Thus, for the purposes of simulation, an arrangement of the n charged particles within the unit cell need only be simulated, and long-range forces from a suitably large number of replicas of the unit cell simulate the bulk phase behavior of the system.

The Ewald sum technique is the most widely used method for computing electrostatic potential due to an *infinite* lattice of repeated unit cells [7]. Various algorithms have been developed to compute the Ewald sum. The first to do so in better than $O(n^2)$ time, where n is the number of particles in the unit cell, was the $O(n^{3/2})$ algorithm of [15]. Since then, improved $O(n \log n)$ algorithms have been developed [3, 13, 21].

The Ewald sum is increasingly being used in the biochemical community for solvent simulation [1, 16], especially as truncation of the long-range electrostatic contributions has been shown to cause problems [11, 18, 22].

Much work has been done on the so-called N-body problem of computing the forces within only the unit cell. Early work dealt with small systems where a brute force $O(n^2)$ all-pairs computation was sufficient. However, as simulation sizes have increased, some algorithms have resorted to cutoffs, where interactions outside of a given radius are neglected. A large amount of work has been done on the N-body problem, culminating in the $O(n)$ fast multipole algorithm (FMA) [8]. The FMA derives its speed from using series expansions to represent the potential due to distant groups of particles at local particle sites.

Both the FMA and Ewald lattice sum concept were coupled together in [17]. In this work, the coefficients for the series expansion of the potential of the unit cell must

0021-9991/96 $18.00

be translated from all infinite lattice sites (except the innermost 27) to the center of the original unit cell. Because the coefficients are identical, they can be factored out of the sum, and an Ewald sum over the lattice sites computes the transformation matrix which is constant and independent of the contents of the unit cell. This matrix is applied to the multipole moments to get the local expansion for the infinite lattice except the central 27 cells. The innermost 27 sites are not included in the sum, as the cells are not *well-separated* from the unit cell. These cells and the unit cell itself are computed with the FMA, and the contribution of the infinite surrounding lattice of unit cells is incorporated at level 0 before the downward pass (see Section 3.2.6). The limitation of this method is that only infinite sums that are readily evaluated by Ewald-type methods can be computed. One could also do small finite sums by explicitly summing the transformation matrices of surrounding unit cells, but until the present work, no general method for swiftly summing large lattices of arbitrary shape has been derived.

A new and efficient algorithm based on multipole expansions is presented for computing the potentials and forces due to a finite but extremely large lattice of periodic unit cells. The method is not restricted to infinite sums, the unit cell need not be centered, and an infinite variety of macroscopic shapes is possible. The algorithm does not require that the unit cell be charge neutral. Expressions for error bounds are derived, and the forces and potentials can be computed to the desired accuracy with no truncation of long-range forces.

The method is first illustrated in 2D for expository purposes and is then extended to 3D systems of particles. In 2D, we obtain an algorithm to compute forces and potential due to a $3^k \times 3^k$ square lattice of n-particle unit cells in $O(nk + k^2 \log k)$ time. In 3D our algorithm computes a $3^k \times 3^k \times 3^k$ cubic lattice of unit cells in $O(nk^2 + k^3 \log k)$ time. We show that for small (constant) values of k, the infinite Ewald sum force values are approximated to high precision, effectively making the algorithm linear time in n.

We then illustrate extension of the algorithm to arbitrary noncubic macroscopic shapes, several of which have been implemented.

2. MATHEMATICS OF POTENTIAL ENERGY IN 2D

2.1. Introduction

In two dimensions, the potential at $z \in \mathbb{C}$ due to a point charge of intensity q at z_0 is given by

$$\Phi(z) = -q\Re(\log(z - z_0)), \tag{1}$$

where $\Re$ () is the real part of a complex number. Expressing coordinates as complex variables makes for simple series descriptions for the electrostatic potential. Efficient evaluation of potentials (and forces) for many point charges is done with a truncated series expansion.

2.2. Multipole Expansions

The well-known multipole expansion describes the potential due to a set of charges within a disk of radius r centered at c, at a point z outside the disk.

LEMMA 2.1 (Multipole expansion) (Adapted from [8]). *Suppose that n charges of strengths $\{q_i, i = 1, ..., n\}$ are located at points $\{z_i, i = 1, ..., n\}$, with $|z_i - c| < r$. Then for any $z \in \mathbb{C}$ with $|z - c| > r$, the potential $\Phi(z)$ induced by the charges is given by*

$$\Phi(z) = -\Re(\log(z - c)) \sum_{i=1}^{n} q_i - \Re \sum_{k=1}^{\infty} \frac{\sum_{i=1}^{n} - q_i(z_i - c)^k}{k(z - c)^k}.$$

For computational purposes, the infinite series is truncated to p terms. The *multipole coefficients array* is an array of p complex numbers, $\{a_0, ..., a_{p-1}\}$, where a_0 is real and is the sum of all of the charges,

$$a_0 = \sum_{i=1}^{n} q_i, \tag{2}$$

and complex elements a_k for $k = 1, ..., p - 1$ are given by

$$a_k = \frac{\sum_{i=1}^{n} - q_i(z_i - c)^k}{k} \tag{3}$$

which is the multipole expansion of Lemma 2.1 with the $1/(z - c)^k$ terms factored out.

LEMMA 2.2 (Multipole expansion translation)(see [8]). *Let a multipole expansion due to n charges in a disk of radius r be centered about c as in Lemma* 2.1. *Then the center of the expansion can be translated to a new center, c', and have the resulting multipole expansion converge outside a radius of $r' = r + |c - c'|$.*

The main consequence of this lemma for this work is that multipole expansions for several regions can be grouped together into a single net multipole expansion describing the potential due to the collective region. One simply translates the multipole expansion for each small region in turn to a common center and then adds the multipole coefficients, a_k, component-wise. The resulting multipole expansion converges outside of a circle centered at the given center with radius enclosing all of the smaller regions.

LEMMA 2.3 (Multipole to local expansion conversion) (See [8]). *Suppose that n charges of strengths $q_1, q_2, ..., q_n$ are located inside a disk D_a of radius a and center at z_a,*

with $|z_a| > a + b$ and $a \geq b$. Then the corresponding multipole expansion converges inside a disk D_b of radius b centered about the origin. At any point inside D_b, the potential due to the charges in D_a can be described by a Taylor series, or local expansion translated to the origin. The worst-case error of evaluating the potential via this p-term local expansion can be bounded by

$$A\left(\frac{a+b}{|z_a|}\right)^p, \tag{4}$$

where $A = \sum_{i=1}^{n} |q_i|$.

Note that this bound differs from Greengard's original derivation as it explicitly includes the radii of the multipole and local expansions [5]. This result also holds in 3D.

3. MULTIPOLE EXPANSIONS AND PERIODIC CELLS IN 2D

3.1. Aggregate Multipole Expansions

The following lemma provides the insight into how to compute efficiently the potential of repeated unit cells.

LEMMA 3.1. *The multipole coefficients due to a set of points $z_1, ..., z_n$ centered at c are identical to the multipole coefficients due to a set of points $(z_1 + y), (z_2 + y), ..., (z_n + y)$ centered at $(c + y)$, for any $y \in \mathbb{C}$.*

Proof. The coefficient a_0 defined by Eq. (2) remains the same, and coefficients a_k of Eq. (3) are given by

$$a_k = \frac{\sum_{i=1}^{n} - q_i((z_i + y) - (c + y))^k}{k} = \frac{\sum_{i=1}^{n} - q_i(z_i - c)^k}{k}.$$

That is, the multipole coefficients for a cell with a given arrangement of particles is independent of the position of the cell. Hence the multipole coefficients for the unit cell are identical to those of all of its copies. Furthermore, because we can translate and add multipole expansions via Lemma 2.2, we can group sets of identical adjacent cells together to form a larger area described by a single new multipole expansion. This can be repeated with the new larger area with corresponding expansion to form still larger areas. Figure 1 illustrates this process. Given this tool, we devise an algorithm for grouping together unit cells into larger cells, in order to quickly evaluate the forces and potential in a large area about the central unit cell.

3.2. Algorithm

We present an algorithm for computing the long-range potential within the unit cell due to all repeated unit cells in a surrounding $3^k \times 3^k$ square lattice of unit cells. We need some notation to describe our algorithm.

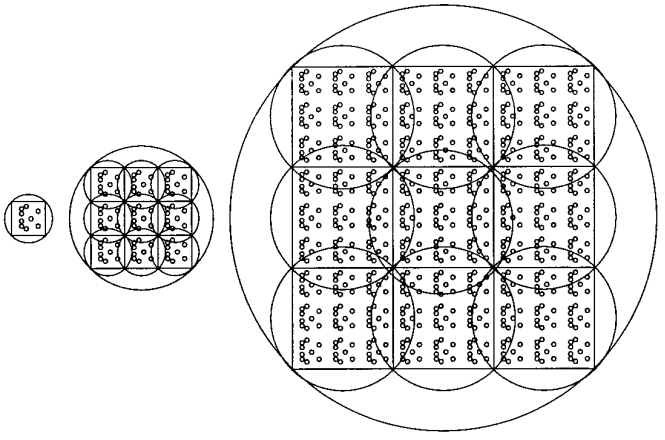

FIG. 1. Aggregation of multipole expansions. The multipole coefficients for the square unit cell on the left are identical to each of the nine on the right. The nine on the right can be shifted and added together to form a single net multipole expansion, and the process can be repeated. Thus if there are n particles in the unit square, a multipole expansion for a square containing 9^k unit squares with $9^k n$ particles can be computed in $9k$ steps of shifting and adding multipole expansions. That is, it is not necessary to compute the multipole expansion directly for all $9^k n$ particles.

S_0 is a positionless square identical to the unit square. We normalize the dimensions of S_0 to 1×1. S_0 can be enclosed in a circle of radius $\sqrt{2}/2$.

S_i is defined recursively as follows: S_i is formed of 9 copies of S_{i-1} in a square of length 3^i with enclosing radius $3^i\sqrt{2}/2$.

M_0 is the multipole coefficients array for S_0.

M_i is the multipole coefficients array for S_i. M_i is recursively formed by shifting multipole coefficients M_{i-1} centered about all nine (x, y) positions, for $(x, y) \in \{-3^i, 0, 3^i\}$ to $(0, 0)$ and adding them together via Lemma 2.2.

3.2.1. *Recursive Step*

The algorithm is as follows:

1. Compute a p-term multipole coefficients array, M_0, centered at $(0, 0)$ for the n particles in the unit square, S_0. Sections 3.2.2 and 3.2.4 present two different schemes for choosing p.

2. Compute each M_i, for $i = 1 \cdots k - 2$ using the recursive definition of M_i above.

3. Take the full repeated particle assembly of $3^k \times 3^k$ unit squares, centered at $(0, 0)$. Let $i = k$ and invoke the following recursive routine. If the current region of side $3^i \times 3^i$ is *too close* (defined below) to the central unit square (or contains it), then recursively subdivide it into nine squares of side $3^{i-1} \times 3^{i-1}$, corresponding to nine copies of S_{i-1}. Otherwise, convert the multipole expansion, M_i, to a local expansion about the unit cell via Lemma

2.3, component-wise, adding the coefficients of the resulting local expansion to a net local expansion due to all macroscopic squares about the unit square. All 1×1 squares, S_0, that are too close, we denote the *innermost boxes* and these will be dealt with in Section 3.2.6.

4. A net local expansion is now computed about the unit cell for all macroscopic expansions that are not *too close*. Process the innermost boxes and then continue with the downward pass of the FMA [8], as detailed in Section 3.2.6.

For a given error, ε, using Eq. (4), we say a square S_i is *too close* if:

$$\varepsilon < 9^i A \left(\frac{r_i + \frac{\sqrt{2}}{2}}{z_i} \right)^p, \tag{5}$$

holds, where $r_i = 3^i\sqrt{2}/2$ is the radius of the circle enclosing S_i and z_i is the distance from the center of S_i to the center of the unit square at (0, 0). The 9^i A comes from there being 9^i times as much charge, A, in S_i as S_0.

Clearly it is desirable to do multipole to local conversions on as few macroscopic expansions as possible. Our notion of *too close* is dependent on the ratio of the macroscopic expansion size to the distance to the unit cell, as well as the parameters ε and p, only the latter of which is adjustable. We derive two schemes for choosing p. The first scheme is theoretically optimal, but practical implementation considerations lead to a second scheme.

3.2.2. *Scheme* 1

We derive a scheme for choosing p which results in the decomposition of Fig. 2. Note that there are 72 unit-sized squares in a square annulus surrounding the black central unit square. In general there are a constant 72 squares of size $3^i \times 3^i$ surrounding the next smaller annulus of 72 squares. Let us denote the square annulus of 72 squares of side 3^i by R_i.

In each R_i, by symmetry, there are four squares that are equally closest to the unit cell. Each such square can be enclosed by a disk of radius $r_i = 3^i\sqrt{2}/2$, with corresponding multipole expansion. The distance from the center of the expansion to the origin is $z_i = 2 \cdot 3^i$. Substituting these values into Eq. (5), we find that for a given precision ε, we must choose p satisfying

$$\varepsilon \geq 9^i A \left(\frac{1}{\sqrt{8}} + \frac{1}{3^i\sqrt{8}} \right)^p, \tag{6}$$

for all $i = 0 \cdots k - 2$. The maximal values for p occur for the innermost and outermost annuli, R_0 and R_{k-2}. We thus obtain

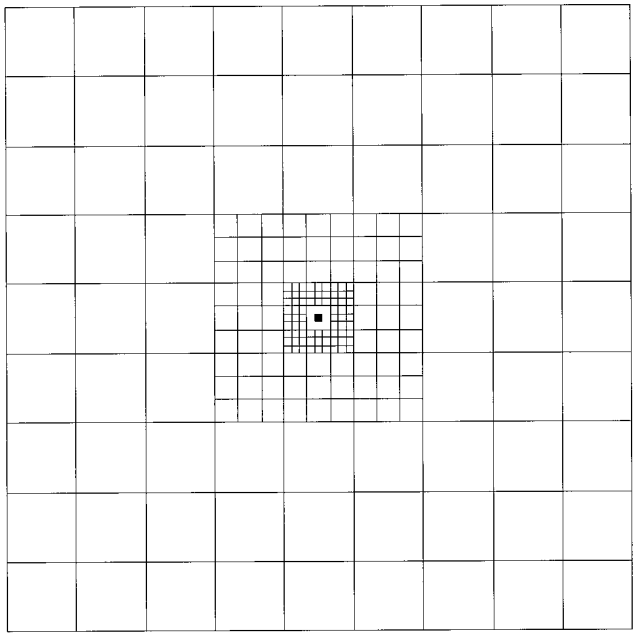

FIG. 2. Scheme 1 spatial decomposition. Squares get exponentially larger as they are exponentially further from the unit cell, in concentric square annuli. Illustrated is a decomposition with $k = 4$, giving a $3^4 \times 3^4$ grid of unit cells. There are $k - 1 = 3$ concentric annuli of 72 boxes, R_0, R_1, R_2. The unit square in the center is shaded. Its eight surrounding squares have to be further subdivided into sub-boxes.

$$p = \max \left\{ \left\lceil \frac{\log\left(\frac{\varepsilon}{9^{k-2}A}\right)}{\log\left(\frac{1}{\sqrt{8}} + \frac{1}{3^{k-2}\sqrt{8}}\right)} \right\rceil, \left\lceil \frac{\log\left(\frac{\varepsilon}{A}\right)}{\log\left(\frac{2}{\sqrt{8}}\right)} \right\rceil \right\}. \tag{7}$$

In general, the former will be the maximum. Boxes in the outermost annulus will have the most error if k is large because the number of particles increases swiftly, while the separation ratio approaches a constant $1/\sqrt{8}$. However, if k is small and ε is small there can be more error in the expansions of ring R_0 than R_{k-2}, and the latter term will be the maximum because the $1/3^i\sqrt{8}$ term makes for slower series convergence for small i.

Table I shows the number of terms 1 that are theoretically necessary in the multipole expansions for some representative values of ε and k.

3.2.3. *Scheme* 1 *Computational Cost*

The multipole to local conversion, as well as the multipole translation needed to compute the initial expansions, M_i, can be formulated in terms of a convolution operation on polynomials. In 2D the $O(p^2)$ cost of the multipole to local conversion can be reduced to an $O(p \log p)$ operation via a 1D FFT-like convolution [9].

TABLE I

Expansion Terms for Given Accuracy

ε	k	p
2.2*e*-16	20	73
2.2*e*-16	10	60
1.*e*-6	20	51
1.*e*-6	10	30
1.*e*-4	5	16

Note. Illustrated are the number of terms, p, needed in the multipole expansions to attain a given error ε for a $3^k \times 3^k$ area of unit cells using Eq. (7). Machine epsilon on a Sun-4 is 2.2*e*-16. Most current simulations need less accuracy.

The cost of computing the initial expansions, M_i for $i = 0 \cdots k-2$ is proportional to $np + kp \log p$ as it takes np steps to compute the initial multipole expansion, M_0, and $9p \log p$ steps each time nine multipole expansions are shifted and added to create successive expansions $M_1 \cdots M_{k-2}$ via FFT techniques. In practice, FFT techniques are unlikely to be used for this step, however, as it represents a very minor proportion of the running time and it is difficult to implement.

We can perform all of the multipole to local conversions of the macroscopic expansions about the unit cell in time proportional to $kp \log p + np$. It costs $p \log p$ operations to do a multipole to local conversion via FFT techniques, which has to be done for a constant 72 boxes in each of $(k-1)$ rings. It then costs a final np steps to evaluate the resulting local expansion at all n points in the unit cell.

In the limit of large k with fixed ε,

$$\lim_{k\to\infty} \frac{1}{k} \frac{\log\left(\frac{\varepsilon}{9^{k-2}A}\right)}{\log\left(\frac{1}{\sqrt{8}} + \frac{1}{3^{k-2}\sqrt{8}}\right)} = \frac{\log 9}{\log(\sqrt{8})}. \tag{8}$$

That is, asymptotically as $k \to \infty$, $p \asymp (\log 9/\log(\sqrt{8}))k \approx 2.113\ k$. Because p grows as a constant times k, we may replace p with k in our expressions for computational cost. This gives us our asymptotically best running time of

$$\Theta(nk + k^2 \log k). \tag{9}$$

Note that, practically, k will be a small constant, certainly less than 20, making the algorithm of order n.

It has been found in practice that the number of terms p does not have to be chosen as large as Eq. (7) prescribes. The worst-case error bounds of the multipole to local conversion arise only when all of the charge has the same sign and is placed at a point on the outermost edge of the box. This will never happen for a macroscopic expansion. No matter how nonuniform the unit cell charge is, the larger macroscopic cells become increasingly uniform because they are formed of unit cells replicated over an evenly spaced lattice. An analysis was done of the most optimistic case where macroscopic boxes are modeled as continuous plates of uniform density $\rho = \sum_{i=1}^{n} q_i$, and a somewhat more realistic bound was derived for the growth of p [12].

3.2.4. *Scheme* 2

The spatial decomposition of Scheme 1 can result in an excessively large number of terms, p, in the multipole expansions. Computations with large p are impractical due to runtime speed and numerical and storage considerations. If we are forced to fix p at some constant, then a spatial decomposition of the form depicted in Fig. 3 results.

Substituting $3^i\sqrt{2}/2$ for r_i in Eq. (5), we say a box is too close if the following holds:

$$\varepsilon < 9^i A \left(\frac{\frac{\sqrt{2}}{2}(1 + 3^i)}{z_i} \right)^p. \tag{10}$$

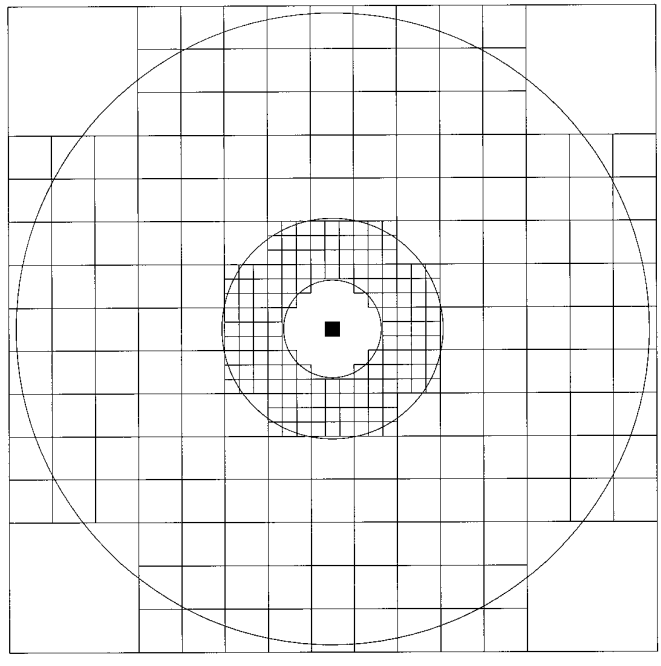

FIG. 3. Scheme 2 spatial decomposition. We illustrate Scheme 2 for $k = 4$, giving us a $3^4 \times 3^4$ lattice of unit cells. The unit cell is at the center in black. Circles, C_i, of radius given by Eq. (11) are drawn with $A = 1$, $\varepsilon = 1e-6$, and $p = 16$. For $i = 0, 1, 2$ we have respective radii of 3.35, 7.69, and 22.03. Squares S_i must be subdivided if their center is within circle C_i.

Solving this for z_i, we find a box S_i is too close if its center lies within the circle C_i. That is, if

$$z_i < C_i = \frac{\sqrt{2}}{2}(3^i + 1)\left(\frac{A}{\varepsilon}\right)^{1/p} 9^{i/p}. \tag{11}$$

3.2.5. *Scheme 2 Computational Cost*

To estimate the number of boxes of size S_i, we need to determine the number that will fit in the annulus formed between the circle of C_i and C_{i+1}. An easier estimate is simply the number that fit in the circle C_{i+1}. We determine the number of boxes, S_i of length 3^i that span C_{i+1}:

$$\frac{C_{i+1}}{3^i} = \frac{\frac{\sqrt{2}}{2}(3^{i+1} + 1)\left(\frac{A}{\varepsilon}\right)^{1/p} 9^{(i+1)/p}}{3^i}. \tag{12}$$

We then square that, multiply by π, and simplify:

$$\pi\left(\frac{\sqrt{2}}{2} 9^{(i+1)/p}\left(\frac{A}{\varepsilon}\right)^{1/p}\left(3 + \frac{1}{3^i}\right)\right)^2 = \frac{\pi}{2} 81^{(i+1)/p}\left(\frac{A}{\varepsilon}\right)^{2/p}\left(3 + \frac{1}{3^i}\right)^2. \tag{13}$$

In general, moving away from the center, the number of boxes of a given size $3^i \times 3^i$ increases exponentially and then drops off as i approaches k. The drop-off phenomenon is due to the fact that only so many boxes of size $3^i \times 3^i$ fit in a $3^k \times 3^k$ box. The upper bound is simply $(3^k/3^i)^2$.

Looking at large simulations, if values of $p = 16$, $i = 15$, $\varepsilon = 1e - 6$, and $A = 1$ are used in Eq. (13), we estimate that the number of multipole expansions of box size 3^{15} is less than 6440. When a real simulation was done, we saw our estimate was conservative, as only 5900 boxes of size 3^{15} had to have multipole expansions evaluated for a $k \gg 15$ simulation. We must sum over all sizes of boxes and multiply the sum by $p \log p$, the cost of a multipole to local conversion, to obtain the total running time. Asymptotically, the algorithm is $\Theta(81^k p \log p)$ for any fixed p, but the assumption is that a good balance can be struck between p and k to make the algorithm efficient in practice.

3.2.6. *The Innermost Boxes*

In both schemes, a certain constant number (dependent on ε) of unit cells will be too close to the central unit cell to perform multipole to local conversions there. We denote these cells, the *innermost boxes*.

One can think of the algorithm as a variant of the FMA, where the interaction lists within the unit cell and its subboxes are augmented to respectively include macroscopic cells and nearby image cells. Scheme 1 and Scheme 2 differ in what comprises the macroscopic interaction list.

In the FMA [8], the unit cell is composed of subcells; in 2D the unit square is recursively subdivided into four subsquares, which in turn are split into four, recursively down to a depth where a small number of particles are within each square. The FMA consists of an upward pass and a downward pass. The upward pass repeatedly translates the multipole expansions of sets of four children together to form larger parent multipole expansions, until multipole expansions for every subcell up to the unit cell are computed. The downward pass then computes the net potential due to all boxes in a given parent's interaction list before shifting the local expansion down to its four children.

The entire algorithm for macroscopic multipole expansions comes in before the downward pass. All macroscopic cells that are not *too close* will be in the interaction list of the central unit cell. The multipole expansions for these cells are converted to local expansions and shifted on top of the central unit cell. When an innermost box is found to be too close to the central unit cell, however, both squares must be subdivided into four children, and multipoles for the four children in the distant square are shifted onto the four children in the unit cube if they are not too close, according to whatever *well-separated* criterion is used for the FMA in the unit cell. If they are not well-separated, recursively subdivide the near and the far cells until either all the children have had their multipole expansions shifted, or else the deepest level of the recursive decomposition is reached. At the deepest level, if two cells are not well-separated, then compute the forces directly. Once all the macroscopic cells are dealt with, the downward pass of the FMA can proceed. Finally, local expansions are evaluated at the leaves of the spatial decomposition tree, giving the net potentials and forces due to the entire macroscopic space.

The computational cost of computing the innermost boxes may not be entirely obvious, as it involves recursively partitioning them to various depths. Because the decomposition of the innermost boxes is identical in form to that of Greengard's $O(n)$ FMA, the runtime cost is still $O(n)$ as there are only a constant number of innermost boxes surrounding the unit cell. There is less actual work than a full FMA on all the innermost boxes, as the forces and potentials need not be computed within the image cells.

4. EXTENDING THE METHODS TO 3D

4.1. Multipole Expansions in 3D

The expressions for multipole expansions in 3D become more complex, as we may no longer resort to expressing coordinates as complex variables. The details of the multipole math will be omitted, but see [6, 10] for the

lemmas for 3D multipole expansions, multipole expansion translations, and local expansions.

4.2. Algorithm

4.2.1. *Recursive Step*

The decomposition is very analogous to the 2D case. The entire computational region is a $3^k \times 3^k \times 3^k$ cube, which is recursively divided into 27 subcubes based on whether each cube is *too close* to the unit cube at center. Analogously to Eq. (5), in 3D a cube S_i is too close if

$$\varepsilon < 27^i A \left(\frac{r_i + \frac{\sqrt{3}}{2}}{z_i} \right)^p. \tag{14}$$

4.2.2. *Scheme* 1

Instead of annuli of 72 squares, we obtain concentric 3D shells of 702 cubes, exponentially increasing in size. Cubes in the innermost shell, corresponding to R_0 in the 2D case will end up being too close in 3D, so the inner 729 boxes will be dealt with as the innermost boxes. For shells $R_1 \cdots R_{k-2}$, there will be six cubes equally closest to the unit cube in each shell. These cubes have $r_i = 3^i\sqrt{3}/2$, and $z_i = 2 \cdot 3^i$, and to obtain a given error, ε, the following must hold:

$$\varepsilon \geq 27^i A \left(\frac{\sqrt{3}}{4} + \frac{1}{3^i\sqrt{8}} \right)^p. \tag{15}$$

Thus p is chosen to satisfy:

$$p = \max \left\{ \left\lceil \frac{\log\left(\frac{\varepsilon}{27^{k-2}A}\right)}{\log\left(\frac{\sqrt{3}}{4} + \frac{1}{3^{k-2}\sqrt{8}}\right)} \right\rceil, \left\lceil \frac{\log\left(\frac{\varepsilon}{A}\right)}{\log\left(\frac{\sqrt{3}}{4} + \frac{1}{\sqrt{8}}\right)} \right\rceil \right\}. \tag{16}$$

It should also be noted that in the 3D case for a given number of expansion terms, p, we obtain a coefficients matrix which by symmetry ends up having $p(p + 1)/2$ elements instead of only p as in 2D. This increases the cost of performing multipole expansion operations as we shall see in the next section.

4.2.3. *Scheme* 1 *Computational Cost*

The cost of computing the initial expansions, M_i for $i = 0 \cdots k - 2$ is proportional to $(np^2 + kp^2 \log p)$ as it takes np^2 steps to compute the initial multipole expansion, M_0, and $27p^2 \log p$ steps each time 27 multipole expansions are shifted and added to create successive expansions $M_1 \cdots M_{k-2}$ via FFT techniques [6, 9].

Doing the multipole to local conversion on all macroscopic cubes takes time proportional to $kp^2 \log p + np^2$ steps. It costs $O(p^2 \log p)$ operations to do a multipole to local conversion, which has to be done for a constant 702 cubes in $(k - 2)$ shells. It costs a final np^2 steps to evaluate the resulting local expansion at all n points in the unit cell. Analogously with the 2D case, as $k \to \infty$, we find that p increases linearly as

$$\left[\log(27)/\log\left(\frac{4 \cdot \sqrt{3}}{3}\right) \right] k \approx 3.94k. \tag{17}$$

Our 3D algorithm thus runs in time

$$\Theta(nk^2 + k^3 \log k). \tag{18}$$

4.2.4. *Scheme* 2

Using Eq. (14), we say a cube, S_i, is too close if

$$\varepsilon < \mathrm{A}27^i \left(\frac{\sqrt{3}\,(3^i + 1)}{2z_i} \right)^p \tag{19}$$

holds, where z_i is the distance from the center of the cube to the center of the unit cube at (0, 0). Solving this for z_i, we find a box S_i is too close if its center lies within the sphere C_i. That is, if

$$z_i < C_i = \frac{3^{i+1/2}}{2} \left(\frac{A}{\varepsilon} \right)^{1/p} 27^{i/p}. \tag{20}$$

4.2.5. *Scheme* 2 *Computational Cost*

Using a similar analysis to the 2D case, we can estimate the number of cubes S_i that fit in a sphere of radius C_{i+1}, which is

$$\frac{4\pi}{3} \left(\frac{C_{i+1}}{3^i} \right)^3. \tag{21}$$

The key to not having too many cubes is to ensure that the fraction k/p remains small.

4.2.6. *The Innermost Boxes*

The innermost boxes are dealt with analogously to the 2D algorithm. The only differences are that the unit cell is recursively divided into eight subcubes as compared to the four squares of the 2D case, and a different criteria for well-separatedness is used in 3D (see [2] for a discussion of such criteria).

4.3. Other Macroscopic Shapes

It is relatively straightforward to extend the method to compute potentials and forces of unit cubes in macroscopic shapes other than a cube. Additionally, the unit cube may lie anywhere within the macroscopic shape. The algorithm for an arbitrary shape is essentially a preprocessor for the regular macroscopic algorithm, and is as follows:

1. Choose k so that the macroscopic cube of size $3^k \times 3^k \times 3^k$ centered about the unit cube completely encloses the shape of interest. The algorithm is recursive, starting with the entire macroscopic cube.

2. Each cube is recursively subjected to the following three rules until the algorithm terminates:

(I) If some, but not all, of the cube is outside of the shape, subdivide it into 27 cubes of side 3^{k-1}. If $k = 0$, discard the cube.

(II) If the cube is completely inside the shape, pass the cube to the macroscopic algorithm, allowing it to compute the potential due to that cube at the coarsest possible level. Terminate processing this cube.

(III) If the cube is completely outside the shape, discard it.

The algorithm is efficient, in that one need only recursively subdivide large cubes when they cross shape boundaries, so all the interactions are at the coarsest possible level. The running time for this algorithm is, of course, shape dependent, and at worst it is proportional to the surface area of the shape divided by the surface area of the unit cell.

Additionally, it is easy to periodically replicate unit cubes in only one or two out of three dimensions. When recursively forming macroscopic aggregate expansions, one simply omits cubes along the appropriate axis. So for instance, to simulate a macroscopic membrane, one would form aggregate expansions from increasingly larger cubes only in the $x - y$ plane.

5. IMPLEMENTATION

5.1. Implementation Details

In two dimensions, the implementation of multipole expansions is rather simple as compared to the 3D case. Therefore, to test the algorithm, a 2D implementation was first done of Scheme 1, omitting consideration of the innermost boxes. Scheme 2 was later devised and implemented in 2D, when it became apparent that using upwards of 40 terms in the multipole expansions in Scheme 1 resulted in numerical instabilities.

Since a goal of this project was the simulation of electrostatic interactions in 3D systems of atoms, a complete implementation of the 2D case was foregone in favor of a full 3D implementation. Typically electrostatic packages based on the fast multipole algorithm have been major undertakings involving many months of work. Given that the FMA is a subalgorithm needed as a part of the macroscopic expansion algorithm, it was desirable to integrate the new algorithm presented here into an existing multipole-accelerated electrostatics code. The local PMTA [2] program was the logical platform to start from.

It turned out to be relatively straightforward to re-tool PMTA to incorporate macroscopic expansions, especially as no new multipole expansion routines had to be written. Most of the new code lay in two recursive routines, one to deal with the macroscopic expansions and one to deal with the innermost boxes outside of the unit cell. Only Scheme 2 was implemented in 3D.

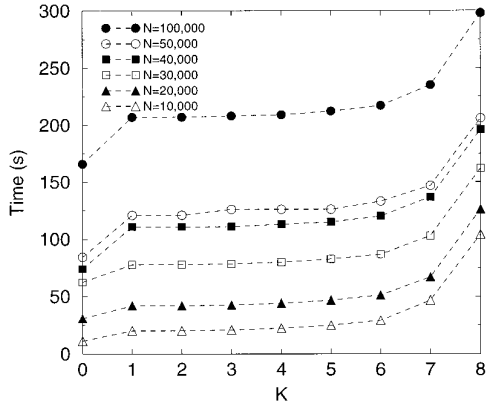

FIG. 4. Running times for medium accuracy simulations. Running times are depicted for simulation sizes ranging from 10,000 to 100,000 particles, uniformly distributed within the unit cube. The algorithm was run at medium ($p = 8$) accuracy for a $3^k \times 3^k \times 3^k$ macroscopic volume of unit cells. We implemented Scheme 2 with FFT-based multipole to local conversions for optimum performance. Within the unit cell, either a four or five level uniform decomposition was used (depending on the system size) corresponding to 8^3 or 8^4 cubes, respectively, at the lowest level of the decomposition. For comparison, it takes 10,500 s to do a 100,000 particle direct $O(n^2)$ computation on the unit cell alone.

5.2. Running Times

All computations were done on a Hewlett–Packard 735/125 workstation. Figure 4 shows running times for the new macroscopic multipole algorithm (3D Scheme 2) for various sizes of moderate accuracy simulations. In these trials, eight expansion terms were retained in the expansions ($p = 8$), which in free-space simulations corresponds to four-to-five significant figures in the potentials, and three-to-four figures in the forces. For a given number k, the potential and forces due to a $3^k \times 3^k \times 3^k$ space of unit cells are computed at all n atom locations in the unit cell. Hence $k = 0$ corresponds to running the FMA on

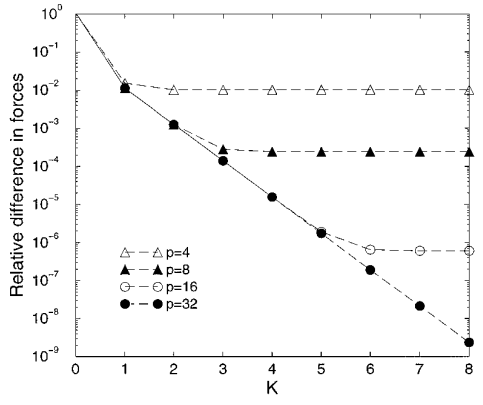

FIG. 5. Ewald sum comparison for a small box of water. The exact Ewald sum forces were computed for an equilibrated cube of 216 water molecules, and then compared with the forces computed by the macroscopic multipole algorithm with $p = 4$, 8, 16, and 32 for increasing values of k. Bonded interactions are excluded. The mean of the relative differences in the forces falls off rapidly as k is increased; then it hits a point of no improvement as the error threshold from truncating the expansions to a fixed number of terms is reached.

only the unit cell, while $k = 8$ is a $6561 \times 6561 \times 6561$ volume of unit cells. For the 100,000 atom unit cell, for $k = 2 \cdots 6$, it takes about 25–30% more time to compute the potentials/forces due to the macroscopic volume (including the unit cell and innermost boxes) as it does to compute them within the n-particle unit cell alone. The curves lose their flatness at $k = 7$, due to the $27^{i/p}$ factor in Eq. (20) which leads to an increase in the number of macroscopic expansions as estimated by Eq. (21). As the system size is increased, the macroscopic part of the algorithm makes up a smaller and smaller proportion of the running time, as there are a fixed number of multipole to local conversions for any given decomposition depth.

With a very high accuracy simulation ($p = 16$) the curves remain relatively flat at $k = 8$, and for a low accuracy simulation ($p = 4$) they lose their flatness at $k = 4$ (results not shown). Later in Figs. 5 and 6, the method is compared to the infinite Ewald sum. We see that for a $p = 4$ simulation, it is pointless to compute k greater than 2, because the error in the expansions exceeds the error in approximating the infinite Ewald sum with a $3^2 \times 3^2 \times 3^2$ macroscopic cube. Similarly, for $p = 8$ the threshold for k is 4, and for $p = 16$, the threshold for k is 5. Thus the $27^{i/p}$ blowup appears to never figure into practical simulations.

5.3. Asymptotic Behavior of Forces and Potentials

It is interesting to examine how the potential and forces behave as k is increased. A small system was run with k ranging from $0 \cdots 7$, and the potential and vector components of the force upon representative particle positions were examined. For a given k, we computed the forces and potentials at the n particle positions due to a $3^k \times 3^k \times 3^k$ volume of unit cells, minus the contribution of the central $3^{k-1} \times 3^{k-1} \times 3^{k-1}$ volume, giving the contribution of the kth shell to the particle forces and potentials.

With a unit cell whose charges sum to zero, as is required for the Ewald sum to converge, the magnitude of the contribution to the potentials and forces due to the kth shell decreases exponentially as k is increased. For a unit cell that is not charge neutral, the potentials increase exponentially with k, but the forces *still* fall off exponentially. For $k \geq 2$, log-plots of the force and potential contributions are linear, making it possible to integrate the decreasing exponentials to infinity. Thus potentials can be calculated due to the infinite lattice when the unit cell is neutral, and the forces can be computed irregardless of the charge neutrality of the system, and without resorting to adding fictitious charges. In practice a relatively small k approximates the infinite Ewald sum quite well without this extrapolation.

5.4. Comparison to the Ewald Sum

To explicitly compare an exact Ewald sum computation to the results of the macroscopic multipole algorithm, we used the Ewald code from [3] and compared the forces generated by the two algorithms. At first attempt, the answers were not agreeing, but this was traced back to the Ewald sum code neglecting to add in the so-called *dipole correction*. If the unit cell has a nonzero net dipole moment, $\mathbf{d} = \sum_{i=}^{n} q_i \mathbf{r}_i$, then the surface of the infinite lattice will be charged, which contributes to the potentials and forces within the unit cell. This effect is often ignored when computing the Ewald sum, but a correct treatment adds a term that is dependent on the limiting shape of the infinite volume and the volume and dipole moment of the unit cell.

Deem et al. [4] derive a general dipole correction expression for arbitrary shapes. Applying this expression to the cubic-shaped infinite volume and taking the gradient, we obtain a force correction for a given point charge q_i:

$$\mathbf{f}_{\mathrm{tot}_i} = \mathbf{f}_{\mathrm{Ewald}_i} - \frac{4\pi q_i}{3V}\mathbf{d}, \tag{22}$$

where V is the volume of the unit cell. Interestingly, this correction is the same as for the infinite sphere.

Correction of the Ewald sum forces brought them into agreement with the forces computed by the macroscopic algorithm. As k is increased, the macroscopic algorithm better approximates an infinite cubic volume and the agreement with the Ewald sum improves. Figure 5 shows the mean of the relative differences between the forces as generated by the two methods on a small cube of water as k is increased. The formula used for the mean of relative force differences is

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\|\mathbf{f}_{\text{Ewald}_i}-\mathbf{f}_{\text{Macro}_i}\|_2}{\|\mathbf{f}_{\text{Ewald}_i}\|_2}. \tag{23}$$

Simulations are run for values of p from 4–32, and the curves go flat where it is pointless to increase k, as the error threshold from truncating the expansions is reached. In practice 32 terms take too long to compute, and the accuracy from a $p = 16$ and $k = 4$ run is already on the upper end of the scale. This is especially true considering that most Ewald codes neglect the dipole correction which can be several orders of magnitude larger than these discrepancies.

To provide a more realistic test of the algorithm on large biomolecular systems we calculated the electrostatic potential and forces in an ~80 Å cubic unit cell at the center of a $3^k \times 3^k \times 3^k$ cluster of replicas. The electrostatically neutral unit cell consisted of a protein, the catalytic domain of the β-1,4-glycanase Cex from *Cellulomonas fimi*, eight sodium counterions, and 14455 water molecules, for a total of 48097 atoms. The simulation system was built using the AMBER 4.1 package [14]. Coordinates for the protein were taken from entry pdb2exo.ent [20] in the Brookhaven protein data bank. Hydrogen atoms were added using the AMBER Edit module, and these were minimized in vacuo, holding the rest of the protein fixed. After this the TIP-3P waters were added, again using the Edit module, and eight waters were chosen at random from those which were at least 6 Å distant from the protein, to be replaced by sodiums. The waters and counterions were then subjected to 40 ps of MD simulation while holding the protein fixed. Finally the entire system was subjected to 100 ps of unconstrained MD under constant pressure conditions, using Ewald summation via the PME option [3] for long-range electrostatics. The equilibrated unit cell was a cube 78.9 Å on a side. Figure 6 shows the RMS agreement of the Ewald computed forces to those computed with the macroscopic algorithm with increasing k. The convergence is about the same as for the small box of water. Note the error in the $k = 0$ error is larger for the small water box as surface effects are more dominant.

In both simulations, potential energies were typically an additional digit closer to the Ewald sum, following the same curves as the forces. The standard deviations in force differences were between 1 and 3 times the means, with the worst-case outliers being two digits less accurate than the average. The relative error in the total energies for the solvated protein were good to 10 decimal places with $k = 8$ and $p = 32$.

A small unit cell without a dipole moment was also run, and the forces were in excellent agreement, with no dipole correction necessary.

5.5. Other Macroscopic Shapes

To study a system replicated in only two of the three dimensions (as for some membrane studies), about five

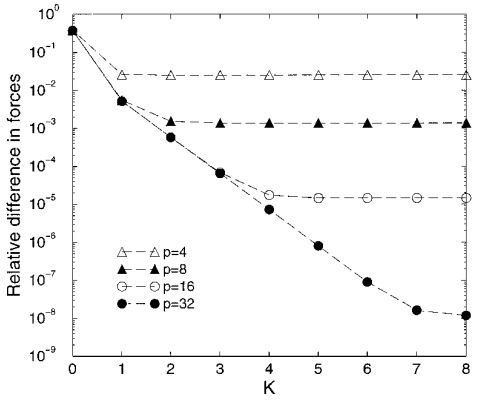

FIG. 6. Ewald sum comparison for a large solvated protein. The exact Ewald sum forces were computed for a large solvated $\alpha - \beta$ barrel protein and then compared with the forces computed by the macroscopic multipole algorithm with $p = 4$, 8, 16, and 32 for increasing values of k. Bonded interactions are excluded. In comparison to Fig. 5, we see a similar rate of convergence to the infinite sum as for the small box of water.

lines of code had to be modified. Two code loops over three dimensions were reduced to loops over two dimensions. To verify correctness, a direct $O(3^{2k}n^2)$ computation was done with small k and n, and agreement was reached in the forces and potentials.

The recursive algorithm of Section 4.3 was implemented for the ellipsoid and the sphere. As is expected, with large k, the spherical results agreed with those of the macroscopic cube, since the dipole term is the same. The ellipsoid has a different dipole term, and indeed, the results differed from the cubic and spheroid cases.

By aggregation of unit cubes in only two dimensions, analogous surface computations were tested for elliptical, circular and square surfaces one unit cube thick. In the limit of large k, with a neutral unit cube, the forces and potentials are the same for square and circular macroscopic surfaces.

The running time for a 50000-atom unit cell within a macroscopic sphere of radius 364 unit cells was 2.35 times longer than that needed to compute a $3^6 \times 3^6 \times 3^6$ macroscopic cube which properly contains the sphere. Although the sphere volume is smaller, there are a greater number of cells in the spherical simulation since outlying cubes are refined to fit the boundaries of the sphere as closely as possible. An ellipsoid with semiaxis dimensions of $364 \times 182 \times 182$ took 1.51 times longer to compute than the macroscopic cube.

6. PERFORMANCE IMPROVEMENTS

The multipole to local operation takes the bulk of the running time. Much of this time is spent calculating transfer functions of the coordinates of the macroscopic expansions

which do not change from timestep to timestep. Not only can this work be eliminated by computing the transfer functions once, but they can be factored together to form one net transfer function (or perhaps k, one for each size of macroscopic cell) because the macroscopic cubes are identical. With this improvement, the runtime curve of Fig. 4 is expected to be flat for all values of k. There will be a large payoff for noncubic macroscopic shapes, as the overhead of computing transfer functions for subdivided cubes at the boundaries need only be done once—possibly storing the constants in a file for future use. This is analogous to the approach of [17], where the net transfer function for all infinite copies of the unit cell can be precomputed once with an Ewald lattice sum approach. Unlike their method, we are not restricted to infinite sums, the unit cell need not be centered, and an infinite variety of macroscopic shapes is possible.

The macroscopic algorithm was built on top of PMTA version 3.1, which is somewhat dated. Substantial performance and algorithmic improvements in PMTA 4.1 have reduced the running times of computing the $k = 0$ cell by two-to-three times. Once the macroscopic expansions are added to PMTA 4.1, substantial performance improvements are expected, particularly in computing the innermost boxes.

PMTA currently compiles to both serial and parallel versions on both distributed and shared-memory machines. We have presently only a serial implementation of PMTA with macroscopic expansions. A parallel version is not technically difficult, as the mechanism for doing many multipole to local conversions in parallel has already been optimized in PMTA. The good parallel speedups observed for PMTA are expected to carry through for PMTA with macroscopic expansions.

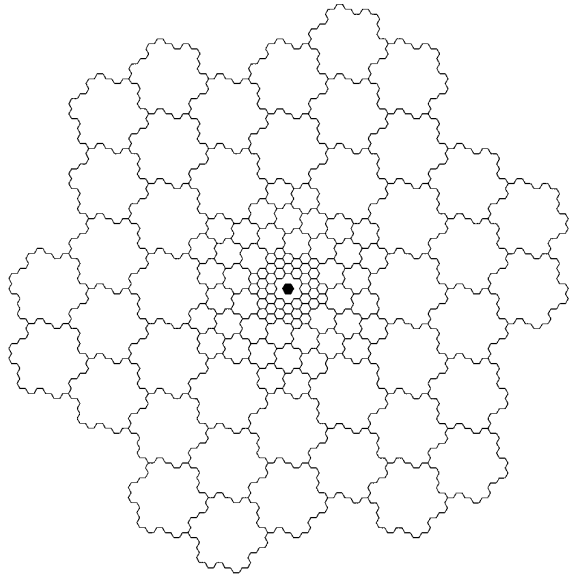

FIG. 7. Hexagonal macroscopic algorithm. Although the details would have to be worked out, it is easy to conceive of a macroscopic multipole algorithm using a hexagonal unit cell. Because the hexagon is closer to circular, one might expect better multipole convergence than with a square unit cell.

7. FUTURE DIRECTIONS

In [19] Smith showed how to calculate the electrostatic energy of a unit cell centered within a large, finite, and arbitrarily shaped crystal. If the energy of the unit cell is denoted by E_L, where L describes the size of the crystal, he showed that the limit as L tends to infinity is the usual Ewald sum plus a dipole correction term. The algorithm outlined in Section 4.3 allows one to compute E_L directly and, thus, to investigate the limit process studied analytically by Smith. Furthermore, as predicted by Deem *et al.* [4], it is now possible to discuss the magnitude of the finite size correction to the asymptotic result, as well as to discuss energies in unit cells closer to the surface of the finite crystal.

There is no need for the unit cell to be square in 2D or cubic in 3D. In 2D, hexagonal lattices group nicely into larger hexagonal regions, making an analogous algorithm easy to conceive of. This idea is depicted in Fig. 7, and it can also be extended to 3D, where it has applications in electrostatics of crystal lattices of various symmetry.

The new algorithm is appropriate to the case where identical copies of a simulation volume are copied out to a great distance. One drawback to this and all periodic techniques is that if, say, a protein is simulated in the unit cell, a conformational shift in the protein would be replicated over all copies of the unit cell, potentially creating spurious correlations within the unit cell. This can be alleviated by making the unit cell larger, as is facilitated by fast N-body algorithms. Nevertheless, if unlimited computational resources were available, one would like to simulate a macroscopic system by looking at the dynamics of every individual atom. Unfortunately, this is not likely to be a realizable goal, even in the distant future.

A future approach to this problem is to determine the difference between multipole expansions of two different large boxes of uniformly distributed particles and to see if there is a way of adding ''noise'' to multipole expansions to simulate similar but nonidentical regions. For example, one could store multipole expansions for the unit cell at various points in the trajectory and randomly pepper them throughout space to give the long-range potential contributions. Methods like this are not easily incorporated into traditional lattice sum methods. Ultimately, one may hope to model electrostatics of large biological systems in terms of sets of repetitive building blocks at differing hierarchies of detail. It is a formidable challenge to understand macroscopic biological processes at the atomic level. The new

tools presented here give us a promising start in that direction.

ACKNOWLEDGMENTS

This work was supported by NSF Grant ASC-9318159 and by NIH Grant RR-08102-01. We thank Leslie Greengard for a valuable discussion and the Theoretical Biophysics group at the University of Illinois at Urbana-Champaign for access to their Hewlett Packard workstation cluster.

REFERENCES

1. M. Belhadj, H. E. Alper, and R. M. Levy, *Chem. Phys. Lett.* **179,** 13 (1991).
2. J. A. Board Jr., Z. S. Hakura, W. D. Elliott, and W. T. Rankin, "Scalable Variants of Multipole-Accelerated Algorithms for Molecular Dynamics Applications, in *Proceedings, Seventh SIAM Conference on Parallel Processing for Scientific Computing, Philadelphia, PA, 1995.*
3. T. Darden, D. York, and L. Pedersen, *J. Chem. Phys.* **98**(12), 10089 (1993).
4. M. W. Deem, J. M. Newsam, and S. K. Sinha, *J. Phys. Chem.* **94,** 8356 (1990).
5. W. D. Elliott, Technical Report 94-008, Duke University Department of Electrical Engineering, 1994 (unpublished).
6. W. D. Elliott and J. A. Board Jr., *SIAM J. Sci. Comput.,* to appear.
7. P. P. Ewald, *Ann. Phys.* **64,** 253 (1921).
8. L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems* (MIT Press, Cambridge, MA, 1988).
9. L. Greengard and V. Rokhlin, Technical Report RR-602, Dept. of Computer Science, Yale University, 1988 (unpublished).
10. L. Greengard and V. Rokhlin, "The Rapid Evaluation of Potential Fields in Three Dimensions," in *Lecture Notes in Mathematics,* Vol. 1360 (Springer-Verlag, New York/Berlin, 1988), p. 121.
11. S. Kuwajima and A. Warshel, *J. Chem. Phys.* **89**(6), 3751 (1988).
12. C. G. Lambert, Master's thesis, Duke University Department of Computer Science, 1994; Techreport CS-1994-23 (unpublished).
13. B. A. Luty, I. G. Tironi, and W. F. van Gunsteren, *J. Chem. Phys.* **103**(8), 3014 (1995).
14. D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham III, D. M. Ferguson, G. L. Seibel, U. C. Singh, P. K. Weiner, and P. A. Kollman, *AMBER* 4.1, University of California, San Francisco, 1995.
15. J. W. Perram, H. G. Petersen, and S. W. DeLeeuw, *Mol. Phys.* **65,** 875 (1988).
16. S. W. Rick and B. J. Berne, *J. Amer. Chem. Soc.* **116,** 3949 (1994).
17. K. E. Schmidt and M. A. Lee, *J. Statist. Phys.* **63,** 1223 (1991).
18. H. Schreiber and O. Steinhauser, *Biochemistry* **31,** 5856 (1992).
19. E. R. Smith, *Proc. R. Soc. London A* **375,** 475 (1981).
20. A. White, S. G. Withers, N. R. Gilkes, and D. R. Rose, *Biochemistry* **33,** 12546 (1994).
21. D. York and W. Yang, *J. Chem. Phys.* **101**(4), 3298 (1994).
22. D. M. York, A. Wlodawer, and T. A. Darden, *Proc. Nat. Acad. Sci.* **91**(18), 8715 (1994).