

Treecode Algorithms for Computing Nonbonded Particle Interactions

Robert Krasny and Zhong-Hui Duan *

Department of Mathematics
University of Michigan, Ann Arbor MI 48109-1109, USA;
krasny@math.lsa.umich.edu, zduan@math.lsa.umich.edu

Abstract. Two new algorithms are described for computing nonbonded particle interactions in classical molecular systems, (1) a particle-cluster treecode for the real space Ewald sum in a system with periodic boundary conditions, and (2) a cluster-cluster treecode for the total potential energy in a system with vacuum boundary conditions. The first algorithm treats electrostatic interactions and the second algorithm treats general power-law interactions. Both algorithms use a divide-and-conquer strategy, adapted rectangular clusters, and Taylor approximation in Cartesian coordinates. The necessary Taylor coefficients are computed efficiently using recurrence relations. The second algorithm implements variable order approximation, and a run-time choice between Taylor approximation and direct summation. Test results are presented for an equilibrated water system, and random and sparse particle systems.

1 Introduction

Consider a molecular system described by a set of particles $\{\mathbf{x}_1(t), \dots, \mathbf{x}_N(t)\}$ and a potential energy function $V(\mathbf{x}_1, \dots, \mathbf{x}_N)$. In molecular dynamics simulations the particles evolve by Newton's equations,

$$m_j \frac{d^2 \mathbf{x}_j}{dt^2} = \mathbf{f}_j(\mathbf{x}_1, \dots, \mathbf{x}_N) , \quad (1)$$

where m_j is the mass of the j th particle and $\mathbf{f}_j = -\nabla_{\mathbf{x}_j} V$ is the force field. To investigate the physical properties of a given system, it is necessary to solve the differential equations (1) using numerical methods. This approach is widespread in biomolecular modeling [1–3].

The potential energy function and force field typically have terms accounting for bonded and nonbonded particle interactions [4]. In this article we are concerned with the efficient computation of the nonbonded interactions. The force exerted on a given particle due to nonbonded interactions with the other particles is assumed to have the general form,

$$\mathbf{f}_j(\mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_{\substack{i=1 \\ i \neq j}}^N q_i q_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) , \quad (2)$$

* This work was supported by NSF grants DMS-9506452 and DMS-9973293, a University of Michigan Rackham Faculty Fellowship, and Michigan Life Sciences Corridor grant #1515.

where q_i is a scalar weight associated with particle \mathbf{x}_i and $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ is the pairwise force function. Note that the interaction is assumed to be separable, i.e. the coefficient of interaction between particles \mathbf{x}_i and \mathbf{x}_j is expressed as the product $q_i q_j$ of weights associated with each particle. In the case of the electrostatic potential, q_i is the partial atomic charge; Lennard-Jones interactions can be treated using combination rules for different types of atoms [5].

Evaluating the force \mathbf{f}_j in (2) for $j = 1, \dots, N$ is an example of an N -body problem and the computational cost is a serious issue [6]. The simplest evaluation method is direct summation, but this requires $O(N^2)$ operations which is prohibitively expensive when N is large. Much effort has been devoted to overcoming this obstacle [7,8]. One remedy is to simply omit interactions outside a specified cutoff radius [5]. However, a serious difficulty arises in biomolecular simulations because the electrostatic potential decays slowly in space and the cutoff procedure may introduce artifacts in the simulation. Various other methods have been developed to reduce the operation count while maintaining accuracy. Particle-mesh methods transfer information between the particles and a regular mesh. The particle-particle/particle-mesh method (P3M) uses direct summation for nearby interactions and a fast Poisson solver for distant interactions [9]. The particle-mesh Ewald method (PME) [10,11] and the fast Fourier-Poisson method (FFP) [12] use the fast Fourier transform (FFT) to gain efficiency; PME is popular in biomolecular simulations and we shall refer to it again below. A recent implementation of Ewald summation uses a multigrid Poisson solver in place of the FFT [13]. Multilevel summation techniques have also been developed [14–18].

The present article concerns an alternative class of methods, *treecode algorithms*, that rely on multipole expansions. In a treecode algorithm, the particles are divided into nested clusters (or cells) and the force \mathbf{f}_j on a given particle \mathbf{x}_j is expressed as a sum of particle-cluster interactions,

$$\sum_{\substack{i=1 \\ i \neq j}}^N q_i q_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \sum_A \sum_{\mathbf{x}_i \in A} q_i q_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \quad , \quad (3)$$

where $A = \{\mathbf{x}_i, i = 1, \dots, N_A\}$ denotes a cluster of particles. The first treecode algorithms used a monopole approximation for the particle-cluster interaction and a divide-and-conquer strategy to choose the clusters; the basic idea is that the approximation is accurate when the particle \mathbf{x}_j and cluster A are sufficiently well-separated [19,20]. The Fast Multipole Method (FMM) is a more elaborate procedure that uses a higher order multipole approximation and a technique for evaluating the approximation by converting it to a local series [21]. These methods reduce the operation count to $O(N \log N)$ or $O(N)$ and they are quite important in biomolecular simulations [22–31]. However, there is great interest in further optimizing the performance of treecode algorithms [32–40].

This article describes two new treecode algorithms for computing nonbonded particle interactions. The first algorithm is a particle-cluster treecode for the real space sum in the Ewald summation method; this applies to computing potential energy and particle forces due to electrostatic interactions in a system with periodic boundary conditions [41]. The second algorithm is a cluster-cluster treecode for the total potential energy; it treats general power-law interactions in a system with vacuum boundary conditions [42]. Both algorithms use a divide-and-conquer strategy,

adapted rectangular clusters, and Taylor approximation in Cartesian coordinates to evaluate particle-cluster and cluster-cluster interactions. The necessary Taylor coefficients are computed efficiently using recurrence relations. The second algorithm implements variable order approximation, and a run-time choice between Taylor approximation and direct summation based on empirical estimates of the required CPU times. Our approach was motivated by recent developments in computational fluid dynamics [43–45]. The algorithms and test results are described in §2 and §3. A summary and conclusions are given in §4.

2 Particle-Cluster Treecode for Ewald Summation

Periodic boundary conditions are commonly used in molecular dynamics simulations to avoid surface effects. Here we assume that the particles are contained in a cube that is replicated periodically in space. In this case, the electrostatic potential energy of the system is

$$V(\mathbf{x}_1, \dots, \mathbf{x}_N) = \frac{1}{2} \sum_{\mathbf{n}} \sum_{i=1}^N \sum_{j=1}^N \frac{q_i q_j}{|\mathbf{x}_i - \mathbf{x}_j + L\mathbf{n}|}, \quad (4)$$

where the index $\mathbf{n} = (n_1, n_2, n_3)$ runs through the periodic images of the cube, the prime indicates that the $i = j$ term is omitted when $\mathbf{n} = \mathbf{0}$, q_i is the partial charge of particle \mathbf{x}_i , and L is the length of the side of the cube. The individual terms in (4) decay slowly as $|\mathbf{n}| \rightarrow \infty$ and the series is conditionally convergent. This implies that the value of the energy depends on the order in which the terms are summed. It is natural to assume that the limit is taken over finite spheres of increasing radius and that each sphere is surrounded by a medium with dielectric constant ϵ_s . In this case, the value of the energy depends on the dipole moment of the basic cube [46].

2.1 Ewald Summation

The Ewald summation method splits the point charge on a particle into a singular short-range term that is treated in real (physical) space plus a smooth long-range term that is treated in reciprocal (Fourier) space [5,46,47]. Assuming the surrounding medium is a conductor ($\epsilon_s = \infty$), this leads to the following expression for the energy,

$$V = V^{(r)} + V^{(k)} - V^{(c)}, \quad (5)$$

where $V^{(r)}$ is the real space sum,

$$V^{(r)} = \frac{1}{2} \sum_{\mathbf{n}} \sum_{i=1}^N \sum_{j=1}^N \frac{q_i q_j \operatorname{erfc}(\alpha|\mathbf{x}_i - \mathbf{x}_j + L\mathbf{n}|)}{|\mathbf{x}_i - \mathbf{x}_j + L\mathbf{n}|}, \quad (6)$$

$V^{(k)}$ is the reciprocal space sum,

$$V^{(k)} = \frac{1}{2\pi L} \sum_{\mathbf{k} \neq \mathbf{0}} \frac{1}{|\mathbf{k}|^2} \exp\left(-\frac{\pi^2 |\mathbf{k}|^2}{L^2 \alpha^2}\right) \left| \sum_{j=1}^N q_j \exp\left(\frac{2\pi i}{L} \mathbf{k} \cdot \mathbf{x}_j\right) \right|^2, \quad (7)$$

and $V^{(c)}$ is a constant self-energy term,

$$V^{(c)} = \frac{\alpha}{\sqrt{\pi}} \sum_{j=1}^N q_j^2 . \quad (8)$$

Equation (5) is exact for any value of the Ewald parameter $\alpha > 0$; the role this parameter plays will be discussed below. For convenience we recall the definition of the complementary error function,

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt . \quad (9)$$

Note that $\operatorname{erfc}(0) = 1$ and $\operatorname{erfc}(x)$ decays rapidly as $x \rightarrow \infty$. Hence the real space sum (6) is a screened version of the electrostatic potential (4). In our discussion we refer mainly to the potential energy; the forces are obtained by analytically differentiating $V^{(r)}$ and $V^{(k)}$ with respect to the particle positions [47], and essentially the same numerical techniques apply.

The advantage in expressing the energy as in (5) is that the real and reciprocal space sums are rapidly convergent with respect to the indices \mathbf{n}, \mathbf{k} . The classical Ewald method uses cutoffs r_c, k_c to evaluate $V^{(r)}$ and $V^{(k)}$, i.e. only terms satisfying $|\mathbf{x}_i - \mathbf{x}_j + \mathbf{L}\mathbf{n}| \leq r_c$ and $|\mathbf{k}| \leq k_c$ are retained in the computation. The magnitude of the Ewald parameter α controls the relative rates of convergence of the real and reciprocal space sums. When α is large, $V^{(r)}$ converges rapidly and can be evaluated to a given accuracy in $O(N)$ operations using an appropriate cutoff r_c ; however in this case $V^{(k)}$ converges slowly and $O(N^2)$ operations are required since the cutoff k_c must be large enough to attain the desired accuracy. The situation is reversed when α is small, and so in either case, the classical Ewald method requires $O(N^2)$ operations. The cost can be reduced to $O(N^{3/2})$ by optimizing the parameters α, r_c, k_c as a function of N [48,49]. The PME method reduces the operation count to $O(N \log N)$ [10,11]. This is accomplished by choosing a large value for α ; the real space sum is computed in $O(N)$ operations and the cost of evaluating the reciprocal space sum is reduced from $O(N^2)$ to $O(N \log N)$ using a particle-mesh interpolation procedure and the FFT.

Our approach is based on the observation that after cutoff, evaluating the real space sum (6) is a standard N -body problem for a screened electrostatic potential. Like PME, our algorithm reduces the operation count to $O(N \log N)$, but it is complementary to PME in that it chooses a small value for the Ewald parameter α ; the reciprocal space sum is computed in $O(N)$ operations and the cost of evaluating the real space sum is reduced from $O(N^2)$ to $O(N \log N)$ using a treecode [41]. The details of this approach are explained below.

2.2 Particle-Cluster Interaction

Consider a particle \mathbf{x}_j and cluster $A = \{\mathbf{x}_i, i = 1, \dots, N_A\}$ as in Fig. 1 (here and in other figures, a two-dimensional schematic is shown instead of the full three-dimensional structure). In the context of Ewald summation, a particle-cluster interaction is given by

$$V_{\mathbf{x}_j, A}^{(r)} = \sum_{\mathbf{x}_i \in A} \frac{q_i q_j \operatorname{erfc}(\alpha |\mathbf{x}_i - \mathbf{x}_j|)}{|\mathbf{x}_i - \mathbf{x}_j|} . \quad (10)$$

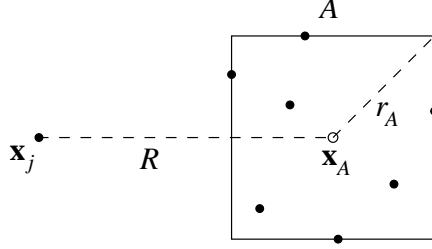


Fig. 1. A particle \mathbf{x}_j and cluster $A = \{\mathbf{x}_i, i = 1, \dots, N_A\}$. \mathbf{x}_A : cell center; r_A : cell radius; R : distance from particle to cell center.

We employ Taylor expansion in Cartesian coordinates to approximate the screened electrostatic potential,

$$\frac{\text{erfc}(\alpha|\mathbf{x}|)}{|\mathbf{x}|} = \frac{2}{\sqrt{\pi}} \sum_{\|\mathbf{n}\|=0}^{\infty} T_{\mathbf{n}}(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{n}}, \quad (11)$$

where $\|\mathbf{n}\| = n_1 + n_2 + n_3$, $T_{\mathbf{n}}(\bar{\mathbf{x}})$ is the \mathbf{n} th Taylor coefficient about an arbitrary base point $\bar{\mathbf{x}}$, and $(\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{n}} = (x_1 - \bar{x}_1)^{n_1} (x_2 - \bar{x}_2)^{n_2} (x_3 - \bar{x}_3)^{n_3}$. Substituting (11) into (10), truncating the series, and rearranging the terms, we obtain a p th order Taylor approximation for the particle-cluster interaction,

$$V_{\mathbf{x}_j, A}^{(r)} \approx \frac{2}{\sqrt{\pi}} q_j \sum_{\|\mathbf{n}\|=0}^p T_{\mathbf{n}}(\mathbf{x}_A - \mathbf{x}_j) m_A^{\mathbf{n}}, \quad (12)$$

where

$$m_A^{\mathbf{n}} = \sum_{\mathbf{x}_i \in A} q_i (\mathbf{x}_i - \mathbf{x}_A)^{\mathbf{n}} \quad (13)$$

is the \mathbf{n} th moment of the cluster. Note that for $\alpha = 0$ (no screening), the Taylor expansion (11) reduces to the classical multipole expansion of the electrostatic potential in Cartesian coordinates [4].

In practice, the approximation is employed only if the following multipole acceptance criterion (MAC) is satisfied,

$$\frac{r_A}{R} \leq s, \quad (14)$$

where r_A, R are the cell radius and particle-cell distance defined in Fig. 1, and s is a user-specified parameter for controlling the computational accuracy [20,36]. If the MAC is not satisfied, the code examines the subcells of the given cell; this reduces the cell radius, making it more likely that the MAC will be satisfied. This divide-and-conquer strategy is characteristic of treecode algorithms and the details will be explained below in §2.5.

2.3 Recurrence Relations

Before proceeding to the tree construction, we explain how the Taylor coefficients are computed. Explicit formulas for the coefficients can be developed, but we found it simpler and more efficient to use recurrence relations instead. First introduce the Taylor expansion of the following Gaussian-type function,

$$\frac{1}{2\alpha} \exp(-\alpha^2 |\mathbf{x}|^2) = \sum_{\|\mathbf{n}\|=0}^{\infty} S_{\mathbf{n}}(\bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{n}} . \quad (15)$$

It can be shown that the Taylor coefficients $S_{\mathbf{n}}$, $T_{\mathbf{n}}$ satisfy a coupled set of recurrence relations,

$$\|\mathbf{n}\| S_{\mathbf{n}} + 2\alpha^2 \sum_{i=1}^3 x_i S_{\mathbf{n}-\mathbf{e}_i} + 2\alpha^2 \sum_{i=1}^3 S_{\mathbf{n}-2\mathbf{e}_i} = 0 , \quad (16)$$

$$\|\mathbf{n}\| \cdot |\mathbf{x}|^2 T_{\mathbf{n}} + (2\|\mathbf{n}\| - 1) \sum_{i=1}^3 x_i T_{\mathbf{n}-\mathbf{e}_i} + (\|\mathbf{n}\| - 1) \sum_{i=1}^3 T_{\mathbf{n}-2\mathbf{e}_i} = \|\mathbf{n}\| S_{\mathbf{n}} , \quad (17)$$

where \mathbf{e}_i denotes the i th Cartesian basis vector, and $S_{\mathbf{n}} = S_{\mathbf{n}}(\mathbf{x})$, $T_{\mathbf{n}} = T_{\mathbf{n}}(\mathbf{x})$. It is understood that $S_{\mathbf{n}} = T_{\mathbf{n}} = 0$ whenever an index n_i is negative. The derivation of (16),(17) is straightforward using Leibniz' rule for differentiating a product of two functions [41]. To evaluate the particle-cluster approximation (12), the recurrence relations are applied with \mathbf{x} replaced by $\mathbf{x}_A - \mathbf{x}_j$ for the given particle \mathbf{x}_j and cell center \mathbf{x}_A . Figure 2 shows the associated stencil.

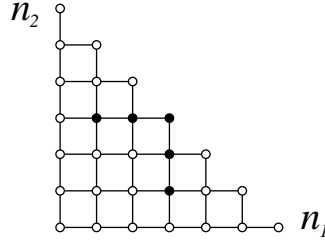


Fig. 2. Stencil (•) of the recurrence relations (16),(17). The Taylor coefficients $T_{\mathbf{n}}$ required for a p th order particle-cluster approximation (12) form a wedge (o) in the index space, $\|\mathbf{n}\| \leq p$. In practice [41], $T_{\mathbf{n}}$ was computed using a slightly different alternative form of (16),(17).

2.4 Tree Construction

The tree construction procedure divides the particles into a collection of nested clusters. Figure 3 compares the standard scheme with the adaptive scheme used here. In the standard scheme, the root cell is bisected in each coordinate direction, yielding eight subcells (or children), and the procedure is repeated recursively on the

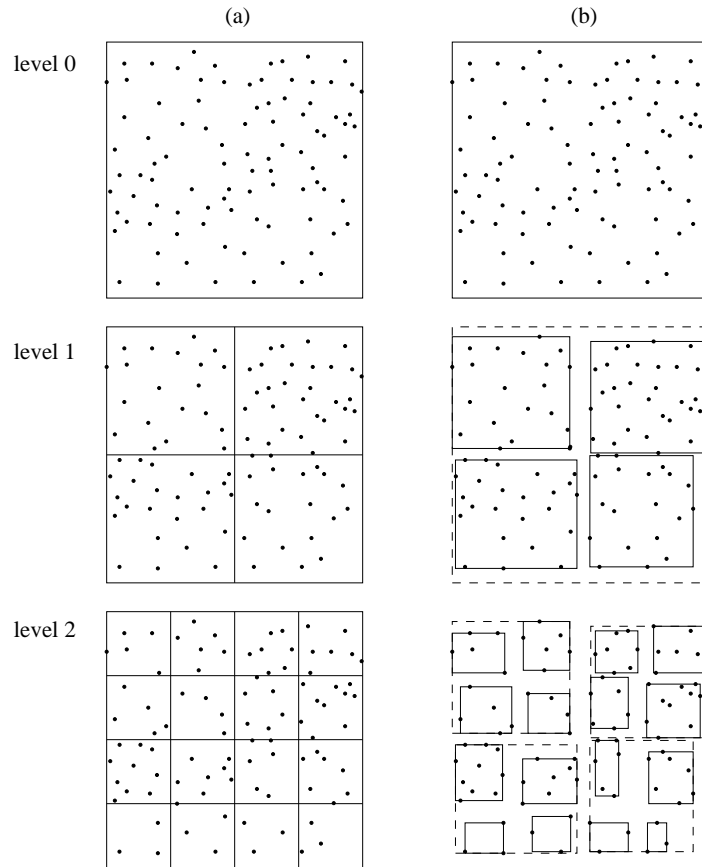


Fig. 3. Tree construction procedure; three levels of clusters are shown. (a) standard scheme (bisect); (b) adaptive scheme (bisect and shrink; the dashed-line cells are from the previous level).

subcells. A cell is left undivided if it contains fewer than a user-specified number of particles N_0 ; these are the leaves of the tree. This yields an oct-tree in which the cells on each level are identical [20,21]. In the adaptive scheme, after a cell is bisected it is shrunk to the smallest rectangular box containing its particles. This yields adapted rectangular clusters in which the cell radii r_A are smaller than they would be without shrinking. As a result, the MAC (14) can be satisfied using a lower order approximation and this leads to a reduction in CPU time. The effect of shrinking is not so dramatic for homogeneous particle distributions, but it may prove more effective in biomolecular simulations involving nonhomogeneous distributions such as an alpha helix or beta sheet.

2.5 Outline of Treecode Algorithm

Figure 4 shows an outline of the particle-cluster treecode algorithm for computing the real space Ewald sum (6). The program **main** inputs the particle data and user-specified parameters, constructs the tree, and cycles through the particles to compute the potential energy and forces. The computation is performed using a divide-and-conquer strategy based on two recursive functions [19,20]. The first function, **compute-in-cell** (\mathbf{x}, A), computes the interaction between a particle \mathbf{x} and a cluster A allowing for the possibility that \mathbf{x} is contained in A . The second function, **compute-out-cell** (\mathbf{x}, A), computes the interaction between a particle \mathbf{x} and a cluster A under the assumption that \mathbf{x} is not contained in A . Specifically, these functions compute the potential energy $V_{\mathbf{x},A}^{(r)}$ and force $\mathbf{f}_{\mathbf{x},A}^{(r)}$ associated with the real space sum for a given particle-cluster interaction. The present description corrects an error in ref. [41] in which two lines in part (c) were misplaced; this concerns only the treatment of leaf cells.

The treecode algorithm and classical Ewald method were implemented in the C programming language and the code was run on a Sun UltraSPARC-II workstation. For the classical Ewald method, the real space sum was evaluated by direct summation with cutoff radius r_c using the linked-list technique [9]. For the treecode algorithm, the real space cutoff was implemented by requiring that the particle-cluster interactions satisfy the criterion $|\mathbf{x}_j - \mathbf{x}_A| \leq r_c + r_A$ (in addition to the MAC (14)); in effect, a cluster A contributes to the real space sum only if it overlaps a sphere of radius r_c centered at the particle \mathbf{x}_j . As a result of this implementation, the treecode computation includes some additional particle interactions beyond those entering the classical Ewald computation. The parameters for the treecode computation were $N_0 = 20$ for the maximum number of particles in a leaf, $s = \frac{1}{2}$ for the MAC parameter, and $p = 6, 8, 10$ for the order of approximation; these are meant to be representative rather than optimized choices. The reciprocal space sum was also computed, using cutoff k_c , and the timing and error results below include contributions from both the real and reciprocal space sums. The enclosed volume varied with N to ensure that the particle density remains fixed.

2.6 Test Results

The test data are a set of water molecules (TIP4P water model [50]). A 1.6 picosecond molecular dynamics simulation was performed to generate equilibrated test configurations for several values of N [51]. One set of results was computed using the classical Ewald method with parameter values

$$\alpha = 6/L, \quad r_c = L, \quad k_c = 12 ; \quad (18)$$

these results are correct to double precision accuracy and they serve as a benchmark for determining the error. We compared the performance of the treecode algorithm and the classical Ewald method for parameter values

$$\alpha = 5.6/L, \quad r_c = L/2, \quad k_c = 6 ; \quad (19)$$

these are commonly used values [5,52] that provide moderate accuracy at lower cost than the values in (18).

```

(a) program main
input particle positions and weights
input user-specified parameters
   $\alpha$  : Ewald parameter
   $r_c$  : real space cutoff radius
   $s$  : MAC parameter
   $p$  : order of approximation
   $N_0$  : maximum number of particles in a leaf
construct tree
compute real space sum for potential energy and forces
  for  $j = 1 : N$ 
    compute-in-cell ( $\mathbf{x}_j, root$ )

(b) function compute-in-cell ( $\mathbf{x}, A$ )
  if  $\mathbf{x} \in A$ 
    if  $A$  is a leaf
      compute  $V_{\mathbf{x},A}^{(r)}, \mathbf{f}_{\mathbf{x},A}^{(r)}$  by direct summation (10)
    else
      for  $i = 1 : 8$ 
        compute-in-cell( $\mathbf{x}, A.child[i]$ )
  else
    compute-out-cell( $\mathbf{x}, A$ )

(c) function compute-out-cell ( $\mathbf{x}, A$ )
  if MAC is satisfied
    compute  $V_{\mathbf{x},A}^{(r)}, \mathbf{f}_{\mathbf{x},A}^{(r)}$  by Taylor approximation (12)
  else
    if  $A$  is a leaf
      compute  $V_{\mathbf{x},A}^{(r)}, \mathbf{f}_{\mathbf{x},A}^{(r)}$  by direct summation (10)
    else
      for  $i = 1 : 8$ 
        compute-out-cell( $\mathbf{x}, A.child[i]$ )

```

Fig. 4. Outline of particle-cluster treecode algorithm for computing the real space Ewald sum. (a) program **main**; (b) function **compute-in-cell** (\mathbf{x}, A); (c) function **compute-out-cell** (\mathbf{x}, A).

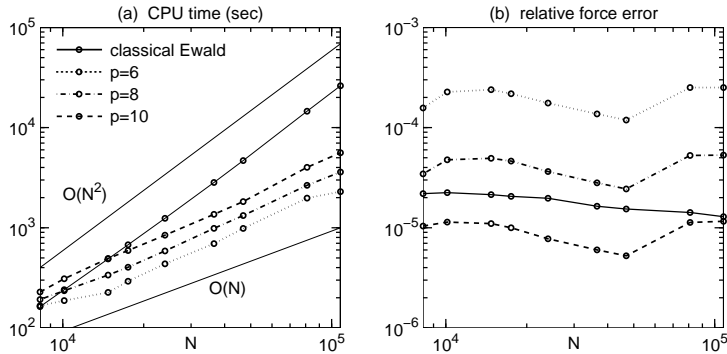


Fig. 5. Test results for a water system with periodic boundary conditions; comparison between classical Ewald method and treecode algorithm with p th order Taylor approximation.

Figure 5a displays the CPU time for evaluating the total potential energy and particle forces. Straight lines are drawn to indicate exact $O(N)$ and $O(N^2)$ behaviour. The CPU time for the classical Ewald method is $O(N^2)$, while for the treecode algorithm it is consistent with $O(N \log N)$. For large enough values of N , the treecode is faster than the classical Ewald method; for example with $N = 107,811$ and $p = 10$, the treecode is more than four times faster than the classical Ewald method. The crossover point is roughly $N = 8,000$ for $p = 6$, $N = 10,000$ for $p = 8$, and $N = 15,000$ for $p = 10$. Figure 5b displays the relative force error given by

$$\left(\frac{\sum_{j=1}^N |\mathbf{f}_j^a - \mathbf{f}_j|^2}{\sum_{j=1}^N |\mathbf{f}_j|^2} \right)^{1/2}, \quad (20)$$

where \mathbf{f}_j^a denotes the approximation to the force on particle \mathbf{x}_j computed using (19) and \mathbf{f}_j denotes the precise result computed using (18). The treecode error is reduced as the order of approximation p increases, and with $p = 10$, the treecode is slightly more accurate than the classical Ewald method.

3 Cluster-Cluster Treecode for Total Potential Energy

In this section we consider a system with vacuum boundary conditions and a potential energy function of the form

$$V(\mathbf{x}_1, \dots, \mathbf{x}_N) = \frac{1}{2} \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N \frac{q_i q_j}{|\mathbf{x}_i - \mathbf{x}_j|^\nu}, \quad (21)$$

where ν is the exponent of a general power-law interaction. As before, we assume that the particle interaction is separable. Important cases include the electrostatic

interaction ($\nu = 1$) and London dispersion ($\nu = 6$, the attractive term in a Lennard-Jones potential) [4]. A number of treecode algorithms and versions of the FMM have been developed to treat such power-law interactions [23,25,28,31].

In a molecular dynamics simulation it is necessary to compute the particle forces $\mathbf{f}_j = -\nabla_{\mathbf{x}_j} V$, but evaluating the total potential energy V is itself an important task in Monte-Carlo simulations [5] and optimization techniques for molecular conformation [53]. Here we describe a cluster-cluster treecode algorithm specifically for computing the total potential energy (21).

3.1 Cluster-Cluster Interaction

The standard procedure for computing the total potential energy is based on the expression

$$V = \frac{1}{2} \sum_{i=1}^N V_i \quad , \quad (22)$$

where

$$V_i = \sum_{\substack{j=1 \\ j \neq i}}^N \frac{q_i q_j}{|\mathbf{x}_i - \mathbf{x}_j|^\nu} \quad . \quad (23)$$

The terms V_i are computed rapidly using a treecode algorithm or FMM, and then they are summed to obtain V . We propose an alternative procedure based on the expression

$$V = \frac{1}{2} \sum_{A, B} V_{AB} \quad , \quad (24)$$

where A, B are suitable pairs of clusters and

$$V_{AB} = \sum_{\substack{\mathbf{x}_i \in A \\ \mathbf{x}_j \in B}} \frac{q_i q_j}{|\mathbf{x}_i - \mathbf{x}_j|^\nu} \quad (25)$$

is the potential energy due to interactions between the particles in cluster A and the particles in cluster B (Fig. 6). When the two clusters are well-separated, V_{AB} can be computed using a Taylor approximation; otherwise, V_{AB} can be computed either by direct summation or by subdividing one of the clusters and considering interactions with the children.

Note that the standard procedure based on (22),(23) expresses V as a sum of particle-cluster interactions, while (24),(25) expresses V as a sum of cluster-cluster interactions. The advantage of the cluster-cluster expression is that it avoids computing the N individual terms V_i in favor of computing a potentially smaller number of terms V_{AB} (in practice, the number of terms is determined adaptively). The cluster-cluster expression for V was proposed by Pérez-Jordá and Yang [54], and it is similar in spirit to Appel's approach [19] as well as to the FMM [21], but those works deal primarily with computing the particle forces rather than the total potential energy.

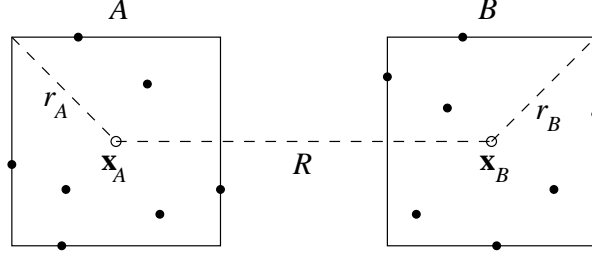


Fig. 6. Two clusters A, B define a cluster-cluster interaction V_{AB} (25). $\mathbf{x}_A, \mathbf{x}_B$: cell centers; r_A, r_B : cell radii; R : distance between cell centers.

3.2 Taylor Approximation for V_{AB}

The general power-law potential has the following expansion,

$$\frac{1}{|\mathbf{x}|^\nu} = \sum_{n=0}^{\infty} C_n^{\nu/2} \left(\frac{\bar{\mathbf{x}}}{|\bar{\mathbf{x}}|} \cdot \frac{\bar{\mathbf{x}} - \mathbf{x}}{|\bar{\mathbf{x}} - \mathbf{x}|} \right) \frac{|\mathbf{x} - \bar{\mathbf{x}}|^n}{|\bar{\mathbf{x}}|^{n+\nu}}, \quad (26)$$

where $C_n^{\nu/2}(y)$ is the Gegenbauer polynomial of degree n and order ν [55]. For $\nu = 1$ this reduces to the well-known spherical harmonics expansion of the electrostatic potential, and for $\nu > 1$ the expansion has been used to extend the FMM to general power-law interactions [25,28]. Here instead of (26) we employ a Taylor expansion in Cartesian coordinates,

$$\frac{1}{|\mathbf{x}|^\nu} = \sum_{\|\mathbf{n}\|=0}^{\infty} T_{\mathbf{n}}(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{n}}. \quad (27)$$

Substituting (27) into (25) and rearranging terms, we obtain a p th order approximation for the cluster-cluster interaction,

$$V_{AB} \approx \sum_{\|\mathbf{n}\|=0}^p T_{\mathbf{n}}(\mathbf{x}_A - \mathbf{x}_B) \sum_{\mathbf{k} \leq \mathbf{n}} \binom{\mathbf{n}}{\mathbf{k}} (-1)^{\|\mathbf{n}-\mathbf{k}\|} m_A^{\mathbf{k}} m_B^{\mathbf{n}-\mathbf{k}}, \quad (28)$$

where $m_A^{\mathbf{k}}$ and $m_B^{\mathbf{n}-\mathbf{k}}$ are cluster moments (13). As before, the Taylor coefficients satisfy a recurrence relation,

$$\|\mathbf{n}\| \cdot |\mathbf{x}|^2 T_{\mathbf{n}} + (2\|\mathbf{n}\| + \nu - 2) \sum_{i=1}^3 x_i T_{\mathbf{n}-\mathbf{e}_i} + (\|\mathbf{n}\| + \nu - 2) \sum_{i=1}^3 T_{\mathbf{n}-2\mathbf{e}_i} = 0. \quad (29)$$

The same conventions apply here as explained for (16),(17). Equation (29) is a three-dimensional analogue of the one-dimensional recurrence relation for the Gegenbauer polynomials [55].

The error in the Taylor approximation of V_{AB} was analyzed to derive an expression for the MAC [42]. For the electrostatic potential ($\nu = 1$), the MAC was chosen to be

$$\frac{1}{R} \frac{r^{p+1}}{1-r} \leq \epsilon, \quad (30)$$

where $R = |\mathbf{x}_A - \mathbf{x}_B|$, $r = (r_A + r_B)/R$, and ϵ is a user-specific parameter for controlling the computational accuracy. The order of approximation p was chosen adaptively; given a pair of clusters A, B , with geometric parameters r, R , the code selects the minimum order p satisfying (30), subject to the constraint $p \leq p_{max}$, where p_{max} is a user-specified parameter. For the dispersion potential ($\nu = 6$), the order of approximation was fixed at $p = 2$ and the MAC was chosen to be

$$\frac{1}{R^6} \frac{1}{5!} \frac{d^5}{dr^5} \left(\frac{r^8}{1-r} \right) \leq \epsilon . \quad (31)$$

The reason for using a fixed low order approximation when $\nu = 6$ is that the Taylor expansion (27) converges relatively slowly in this case and there is little advantage to be gained from higher order [25].

3.3 Outline of Treecode Algorithm

Figure 7 shows an outline of the cluster-cluster treecode algorithm for computing the total potential energy. The program **main** inputs the particle data and user-specified parameters, constructs the tree as described in section 2.4, and then computes the energy V . The computation is performed using a divide-and-conquer strategy based on two recursive functions [19,20]. The first function, **compute-one-cell**(A), computes the energy V_{AA} due to interactions among the particles in cell A . The second function, **compute-two-cells**(A, B), computes the energy V_{AB} due to interactions between the particles in cell A and the particles in cell B , assuming that A and B are disjoint. Note that the second function implements a run-time choice between Taylor approximation and direct summation using the function **direct-is-faster**(A, B); this function accesses a lookup table of precomputed CPU times and returns **true** if direct summation is faster than Taylor approximation and **false** otherwise. The CPU time for direct summation depends on the product $N_A \cdot N_B$ of the number of particles in cell A and cell B , and the CPU time for Taylor approximation depends on the order p . The precomputed CPU times in the lookup table depend on the computer hardware and coding of the algorithms; if these change, then the table should be recomputed.

Results are presented below for three test cases: the electrostatic and dispersion potentials with random particles, and the electrostatic potential with particles on a curve. The maximum order of approximation was $p_{max} = 10$ for $\nu = 1$ and the order was fixed at $p = 2$ for $\nu = 6$. The number of particles ranged from $N = 500$ to $N = 128,000$, and the simulation volume was adjusted to ensure that the particle density remains fixed. The maximum number of particles in a leaf was $N_0 = 30, 10, 20$, in the three test cases, respectively, and in each case, results are presented for three values of the MAC parameter, $\epsilon = 10^{-3}, 10^{-5}, 10^{-7}$. Loops for the recurrence relation and Taylor approximation were expanded to inline code using a separate program. A direct summation computation was the benchmark for comparing CPU times and errors.

3.4 Test Results

The first test case is the electrostatic potential ($\nu = 1$) for a set of particles distributed randomly in space. The charge is $q_i = \pm 1$ with equal probability. Figure 8a

```

(a) program main
input particle positions and weights
input user-specified parameters
   $\epsilon$  : MAC parameter
   $p_{max}$  : maximum order of approximation
   $N_0$  : maximum number of particles in a leaf
construct tree
compute total potential energy
  compute-one-cell(root)

(b) function compute-one-cell(A)
if A is a leaf
  compute  $V_{AA}$  by direct summation (25)
else
  for  $i = 1 : 8$ 
    compute-one-cell(A.child[i])
  for  $j = i + 1 : 8$ 
    compute-two-cells(A.child[i], A.child[j])

(c) function compute-two-cells(A, B)
if MAC is satisfied
  if direct-is-faster(A, B)
    compute  $V_{AB}$  by direct summation (25)
  else
    compute  $V_{AB}$  by Taylor approximation (28)
else
  if (A is a leaf) and (B is a leaf)
    compute  $V_{AB}$  by direct summation
  else if (A is a leaf) or ((B is not a leaf) and ( $r_B > r_A$ ))
    for  $i = 1 : 8$ 
      compute-two-cells(A, B.child[i])
  else
    for  $i = 1 : 8$ 
      compute-two-cells(A.child[i], B)

```

Fig. 7. Outline of cluster-cluster treecode algorithm for computing the total potential energy. (a) program **main**; (b) function **compute-one-cell**(*A*); (c) function **compute-two-cells**(*A*, *B*).

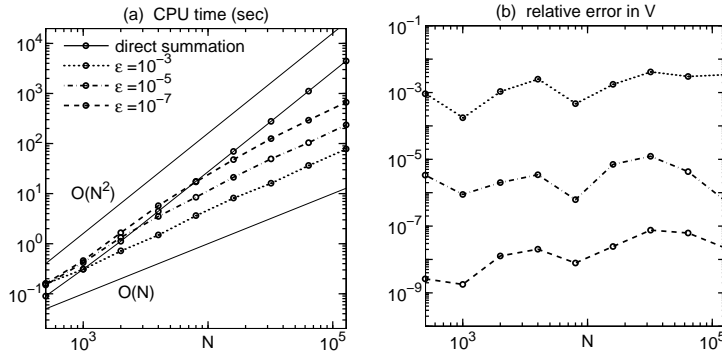


Fig. 8. First test case, electrostatic potential ($\nu = 1$), random \mathbf{x}_i , $q_i = \pm 1$; comparing direct summation and treecode with MAC parameter ϵ (30).

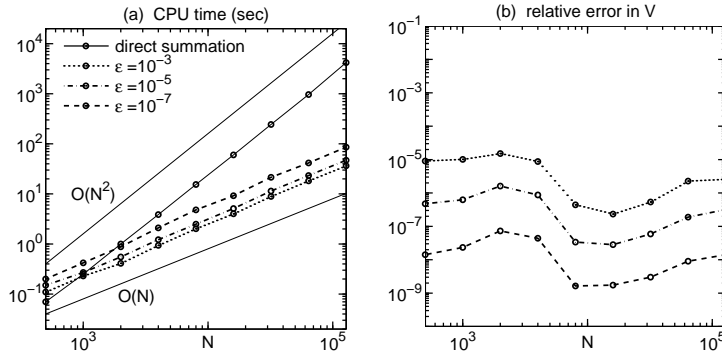


Fig. 9. Second test case, dispersion potential ($\nu = 6$), random \mathbf{x}_i , $q_i = 1$; comparing direct summation and treecode with MAC parameter ϵ (31).

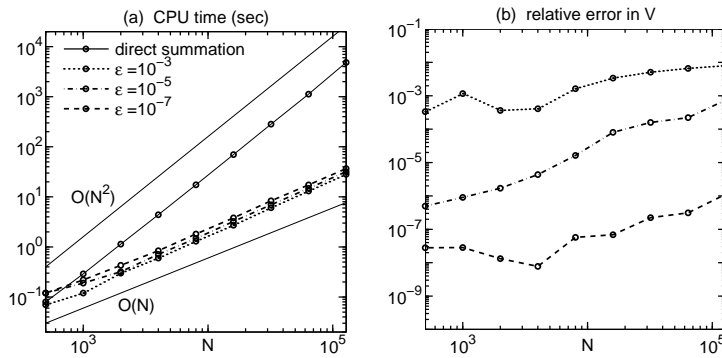


Fig. 10. Third test case, electrostatic potential ($\nu = 1$), B-spline \mathbf{x}_i , $q_i = 1$; comparing direct summation and treecode with MAC parameter ϵ (30).

plots the CPU time required to compute the total potential energy V by direct summation and by the treecode. Direct summation is faster for small systems and the treecode is faster for large systems. The crossover point depends on the MAC parameter; it is $N = 1,000$ for $\epsilon = 10^{-3}$ and increases to $N = 8,000$ for $\epsilon = 10^{-7}$. Above the crossover point, the treecode CPU time increases at a rate consistent with $O(N \log N)$. Figure 8b plots the relative error in the total potential energy computed by the treecode. The error varies slightly with the number of particles, but for each value of N it decreases as ϵ is reduced.

The second test case is the dispersion potential ($\nu = 6$) for the same particle distributions as above but with uniform weights $q_i = 1$. The results, shown in Fig. 9, display the same trends as in the first test case, but some details are different. In Fig. 9a, the direct summation CPU time is roughly the same as in Fig. 8a, but the treecode CPU time is less than in Fig. 8a. The crossover point is $N = 1,000$ for $\epsilon = 10^{-3}$ and increases to only $N = 2,000$ for $\epsilon = 10^{-7}$. The error in Fig. 9b is qualitatively similar to the results in Fig. 8b.

The third test case is the electrostatic potential ($\nu = 1$) for particles lying on a B-spline curve representing a supercoiled DNA molecule (Fig. 3b in [56]). The particles represent phosphate groups with uniform charge $q_i = 1$ and uniform spacing; the latter condition was enforced using the algorithm in [56]. In contrast to the random particle distribution in the first two cases, this is a sparse distribution in three-dimensional space. The results are shown in Fig. 10. The treecode CPU time increases only slightly as ϵ is reduced. The crossover point is less than $N = 1,000$ for all three values of ϵ . The error increases slightly with N , but for a given value of N it decreases as ϵ is reduced.

The results show that the treecode algorithm is behaving generally as expected in terms of CPU time and error (although the variation of the error with N seen in Figs. 8b, 9b, 10b is unexplained). Comparing the CPU time for the first and second test cases shows that the treecode is more effective at speeding up the computation for a short-range potential than for a long-range potential; however, it should be kept in mind that for a short-range potential, direct summation with a suitable cutoff might be competitive. Comparing the CPU time for the first and third test cases shows that the treecode is more effective for a sparse set of particles than for a random set; this is characteristic of adaptive treecode algorithms [33,57].

4 Summary and Conclusions

The cost of evaluating nonbonded particle interactions is a serious obstacle in molecular dynamics simulations. Treecode algorithms typically reduce the operation count from $O(N^2)$ to $O(N \log N)$, where N is the number of particles in the system, but there is still much interest in optimizing performance within this class of algorithms. Here we described two new treecode algorithms that use adapted rectangular clusters and Taylor approximation in Cartesian coordinates. Cartesian Taylor approximation has been used before in treecode algorithms [36] and FMM implementations [23,32]. The present work employs recurrence relations to efficiently compute the necessary Taylor coefficients, up to order $p = 10$, instead of using explicit formulas for the coefficients.

The first algorithm is a particle-cluster treecode for the real space sum in the Ewald summation method. Ewald summation poses two distinct computational

problems, the real and reciprocal space sums. The present treecode algorithm evaluates the real space sum in $O(N \log N)$ operations, while the PME method evaluates the reciprocal space sum in $O(N \log N)$ operations [10,11]. This raises the possibility of combining the two methods to obtain a more efficient hybrid scheme; however this is nontrivial since each method by itself requires a different value of the Ewald parameter.

The second algorithm is a cluster-cluster treecode for the total potential energy in a system with vacuum boundary conditions. The algorithm treats a general power-law potential. It implements variable order approximation, and a run-time choice between Taylor approximation and direct summation based on empirical estimates of the required CPU times. These adaptive techniques have not yet been implemented in the first algorithm.

Test results were presented for an equilibrated water system, and random and sparse particle systems. The CPU time was consistent with $O(N \log N)$ and the algorithms were effective in controlling the computational error.

Given the importance and difficulty of the computational N -body problem, it is not surprising that many different approaches have been developed. The FMM uses sophisticated analytical techniques to improve performance (plane wave translations, optimized Gaussian quadrature) [40]. In contrast, the present approach uses simple analytical techniques (Taylor approximation in Cartesian coordinates, recurrence relations), but combines them with enhanced adaptivity (adapted rectangular cells, variable order approximation, run-time choice between Taylor approximation and direct summation). Aside from methods based on multipole expansions, particle-mesh algorithms are another general approach [9–18]. There are various tradeoffs among these alternatives in terms of speed, accuracy, memory usage, range of applicability, and ease of implementation. Detailed comparisons have been performed [7,47,58–60] and an important development is the emergence of well-defined test problems [61]. More work along these lines will help achieve the goal of physically realistic biomolecular simulations.

References

1. Karplus, M., Petsko, G. A.: Molecular dynamics simulations in biology. *Nature* **347** (1990) 631–639
2. van Gunsteren, W. F., Hünenberger, P. H., Mark, A. E., Smith, P. E., Tironi, I. G.: Computer simulation of protein motion. *Comput. Phys. Comm.* **91** (1995) 305–319
3. Neumaier, A.: Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review* **39** (1997) 407–460
4. Stone, A. J.: *The Theory of Intermolecular Forces*, Oxford University Press, Oxford (1996)
5. Allen, M. P., Tildesley, D. J.: *Computer Simulation of Liquids*, Oxford University Press, Oxford (1987)
6. Greengard, L.: Fast algorithms for classical physics. *Science* **265** (1994) 909–914
7. Sagui, C., Darden, T. A.: Molecular dynamics simulations of biomolecules: long-range electrostatic effects. *Annu. Rev. Biophys. Biomol. Struct.* **28** (1999) 155–179

8. Schlick, T., Skeel, R. D., Brunger, A. T., Kalé, L. V., Board Jr., J. A., Hermans, J., Schulten, K.: Algorithmic challenges in computational molecular biophysics. *J. Comput. Phys.* **151** (1999) 9–48
9. Hockney, R. W., Eastwood, J. W.: *Computer Simulation Using Particles*, IOP Publishing, Bristol (1988)
10. Darden, T., York, D., Pedersen, L.: Particle mesh Ewald: an $N \cdot \log(N)$ method for Ewald sums in large systems. *J. Chem. Phys.* **98** (1993) 10089–10092
11. Essmann, U., Perera, L., Berkowitz, M. L., Darden, T., Lee, H., Pedersen, L.: A smooth particle mesh Ewald method. *J. Chem. Phys.* **103** (1995) 8577–8593
12. York, D., Yang, W.: The fast Fourier Poisson method for calculating Ewald sums. *J. Chem. Phys.* **101** (1994) 3298–3300
13. Sagui, C., Darden, T.: Multigrid methods for classical molecular dynamics simulations of biomolecules. *J. Chem. Phys.* **114** (2001) 6578–6591
14. Brandt, A., Lubrecht, A. A.: Multilevel matrix multiplication and the fast solution of integral equations. *J. Comput. Phys.* **90** (1990) 348–370
15. Brandt, A., Venner, C. H.: Multilevel evaluation of integral transforms with asymptotically smooth kernels. *SIAM J. Sci. Comput.* **19** (1998) 468–492
16. Zaslavsky, L. Y., Schlick, T.: An adaptive multigrid technique for evaluating long-range forces in biomolecular simulations, *Appl. Math. Comput.* **97** (1998) 237–250
17. Sandak, B.: Multiscale fast summation of long range charge and dipolar interactions. *J. Comp. Chem.* **22** (2001) 717–731
18. Sandak, B.: Efficient computational algorithms for fast electrostatics and molecular docking. *this volume*
19. Appel, A. W.: An efficient program for many-body simulation. *SIAM J. Sci. Stat. Comput.* **6** (1985) 85–103
20. Barnes, J., Hut, P.: A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature* **324** (1986) 446–449
21. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. *J. Comp. Phys.* **73** (1987) 325–348
22. Board Jr., J. A., Causey, J. W., Leathrum Jr., J. F., Windemuth, A., Schulten, K.: Accelerated molecular dynamics simulation with the parallel fast multipole algorithm. *Chem. Phys. Lett.* **198** (1992) 89–94
23. Ding, H.-Q., Karasawa, N., Goddard III, W. A.: Atomic level simulations on a million particles: The cell multipole method for Coulomb and London nonbond interactions. *J. Chem. Phys.* **97** (1992) 4309–4315
24. Saito, M.: Molecular dynamics simulations of proteins in water without the truncation of long-range Coulomb interactions. *Molec. Simul.* **8** (1992) 321–331
25. Elliott, W. D., Board Jr., J. A.: Fast multipole algorithm for the Lennard-Jones potential. Tech. Rep. 94-005, Duke University EECS Dept. (1994) (<http://www.ee.duke.edu/research/SciComp/Papers/TR94-005.html>)
26. Shimada, J., Kaneko, H., Takada, T.: Performance of fast multipole methods for calculating electrostatic interactions in biomacromolecular simulations. *J. Comput. Chem.* **15** (1994) 28–43
27. White, C. A., Head-Gordon, M.: Derivation and efficient implementation of the fast multipole method. *J. Chem. Phys.* **101** (1994) 6593–6605
28. Fenley, M. O., Olson, W. K., Chua, K., Boschitsch, A. H.: Fast adaptive multipole method for computation of electrostatic energy in simulations of polyelectrolyte DNA. *J. Comput. Chem.* **17** (1996) 976–991

29. Niedermeier, C., Tavan, P.: Fast version of the structure adapted multipole method - efficient calculation of electrostatic forces in protein dynamics. *Mol. Simul.* **17** (1996) 57–66
30. White, C. A., Head-Gordon, M.: Rotating around the quartic angular momentum barrier in fast multipole method calculations. *J. Chem. Phys.* **105** (1996) 5061–5067
31. Xue, G. L., Zall, A. J., Pardalos, P. M.: Rapid evaluation of potential energy functions in molecular and protein conformations. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **23** (1996) 237–249
32. Zhao, F.: An $O(N)$ algorithm for three-dimensional N-body simulations. AI-TR-995, Massachusetts Institute of Technology (1987)
33. Carrier, J., Greengard, L., Rokhlin, V.: A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Statist. Comput.* **9** (1988) 669–686
34. van Dommelen, L., Rundensteiner, E. A.: Fast, adaptive summation of point forces in the two-dimensional Poisson equation. *J. Comput. Phys.* **83** (1989) 126–147
35. Anderson, C.: An implementation of the fast multipole method without multipoles. *SIAM J. Stat. Sci. Comp.* **13** (1992) 923–947
36. Salmon, J. K., Warren, M. S.: Skeletons from the treecode closet. *J. Comput. Phys.* **111** (1994) 136–155
37. Elliott, W. D., Board Jr., J. A.: Fast Fourier transform accelerated fast multipole algorithm. *SIAM J. Sci. Comput.* **17** (1996) 398–415
38. Strickland, J. H., Baty, R. S.: A pragmatic overview of fast multipole methods. *Lect. Appl. Math.* **32**, (1996) 807–830
39. Wang, H. Y., LeSar, R.: An efficient fast-multipole algorithm based on an expansion in the solid harmonics. *J. Chem. Phys.* **104** (1996) 4173–4179
40. Cheng, H., Greengard, L., Rokhlin, V.: A fast adaptive multipole algorithm in three dimensions. *J. Comput. Phys.* **155** (1999) 468–498
41. Duan, Z.-H., Krasny, R.: An Ewald summation based multipole method. *J. Chem. Phys.* **113** (2000) 3492–3495
42. Duan, Z.-H., Krasny, R.: An adaptive treecode for computing nonbonded potential energy in classical molecular systems. *J. Comput. Chem.* **22** (2001) 184–195
43. Draghicescu, C., Draghicescu, M.: A fast algorithm for vortex-blob interactions. *J. Comput. Phys.* **116** (1995) 69–78
44. Lindsay, K.: A three-dimensional Cartesian tree-code and applications to vortex sheet roll-up. Ph.D. Thesis, University of Michigan (1997)
45. Lindsay, K., Krasny, R.: A particle method and adaptive treecode for vortex sheet motion in 3-D flow. submitted to *J. Comput. Phys.* (2001)
46. de Leeuw, S. W., Perram, J. W., Smith, E. R.: Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constant. *Proc. Roy. Soc. Lond. A* **373** (1980) 27–56
47. Toukmaji, A. Y., Board Jr., J. A.: Ewald summation techniques in perspective: a survey. *Comput. Phys. Commun.* **95** (1996) 73–92
48. Perram, J. W., Petersen, H. G., de Leeuw, S. W.: An algorithm for the simulation of condensed matter which grows as the $3/2$ power of the number of particles. *Mol. Phys.* **65** (1988) 875–893
49. Fincham, D.: Optimisation of the Ewald sum for large systems. *Mol. Sim.* **13** (1994) 1–9

50. Jorgensen, W. L., Chandrasekhar, J., Madura, J. D., Impey, R. W., Klein, M. L.: Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.* **79** (1983) 926–935
51. Rapaport, D. C.: *The Art of Molecular Dynamics Simulation*, Cambridge University Press, Cambridge, (1995)
52. Hummer, G., Pratt, L. R., Garcia, A. E.: Molecular theories and simulation of ions and polar molecules in water. *J. Phys. Chem.* **102** (1998) 7885–7895
53. Vásquez, M., Némethy, G., Scheraga, H. A.: Conformational energy calculations on polypeptides and proteins. *Chem. Rev.* **94** (1994) 2183–2239
54. Pérez-Jordá, J. M., Yang, W.: A simple $O(N \log N)$ algorithm for the rapid evaluation of particle-particle interactions. *Chem. Phys. Lett.* **247** (1995) 484–490
55. Andrews, G. E., Askey, R., Roy, R.: *Special Functions*, Cambridge University Press, Cambridge, (1999)
56. Hao, M.-H. Olson, W. K.: Global equilibrium configurations of supercoiled DNA. *Macromolecules* **22** (1989) 3292–3303
57. Boschitsch, A. H., Fenley, M. O., Olson, W. K.: A fast adaptive multipole algorithm for calculating screened Coulomb (Yukawa) interactions. *J. Comput. Phys.* **151** (1999) 212–241
58. Esselink, K.: A comparison of algorithms for long-range interactions. *Comput. Phys. Commun.* **87** (1995) 375–395
59. Pollock, E. L., Glosli, J.: Comments on P³M, FMM, and the Ewald method for large periodic Coulombic systems. *Comput. Phys. Commun.* **95** (1996) 93–110
60. Deserno, M., Holm, C.: How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines. *J. Chem. Phys.* **109** (1998) 7678–7693
61. Barth, E., Leimkuhler, B., Reich, S.: A test set for molecular dynamics. *this volume*