An Adaptive Treecode for Computing Nonbonded Potential Energy in Classical Molecular Systems

ZHONG-HUI DUAN, ROBERT KRASNY

Department of Mathematics, University of Michigan, Ann Arbor, Michigan 48109-1109

Received 8 September 1999; accepted 28 July 2000

ABSTRACT: A treecode algorithm is presented for rapid computation of the nonbonded potential energy in classical molecular systems. The algorithm treats a general form of pairwise particle interaction with the Coulomb and London dispersion potentials as special cases. The energy is computed as a sum of group–group interactions using a variant of Appel's recursive strategy. Several adaptive techniques are employed to reduce the execution time. These include an adaptive tree with nonuniform rectangular cells, variable order multipole approximation, and a run-time choice between direct summation and multipole approximation for each group–group interaction. The multipole approximation is derived by Taylor expansion in Cartesian coordinates, and the necessary coefficients are computed using a recurrence relation. An error bound is derived and used to select the order of approximation. Test results are presented for a variety of systems. © 2000 John Wiley & Sons, Inc. J Comput Chem 22: 184–195, 2001

Keywords: adaptive treecode; nonbonded potential energy; classical molecular system; multipole approximation

Correspondence to: R. Krasny; e-mail: krasny@math.lsa.umich.edu

Contract/grant sponsor: University of Michigan Rackham Faculty Fellowship

Contract/grant sponsor: National Science Foundation; Contract/grant numbers: DMS-9506452 and DMS-9973293

Introduction

Evaluating the potential energy of a molecular system is an important task in computational chemistry. In particular, this is a basic element in optimization methods for protein conformation[1] and in Monte Carlo techniques for the equilibrium properties of solvents.[2] Here, we consider classical systems represented by a set of particles $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ in three-dimensional space, and nonbonded potential energy functions of the form

$$V = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{q_i\, q_j}{|\mathbf{x}_i - \mathbf{x}_j|^\nu}, \tag{1}$$

where the q_i are scalar weights, $|\mathbf{x}|$ denotes the Euclidean norm, and the parameter ν defines the pairwise interaction potential. Examples include the long-range Coulomb potential ($\nu = 1$) and the short-range London dispersion potential ($\nu = 6$).

The cost of evaluating the potential energy is a critical issue in simulations. The direct summation method relies on the expression

$$V = \sum_{i=1}^{N} V_i, \tag{2}$$

where

$$V_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{q_i q_j}{|\mathbf{x}_i - \mathbf{x}_j|^\nu}. \tag{3}$$

The terms V_i are computed by looping over the index j, and the results are summed to obtain the total potential energy V. The operation count is $O(N^2)$ and thus direct summation is prohibitively expensive for large systems. Several methods have been developed to reduce the cost by introducing approximations. These include cutoff techniques[2] and particle-mesh algorithms.[3] The present work is concerned with an alternative class of methods known as treecodes. A treecode has three basic features: (a) the particles are divided into a nested hierarchy of groups or cells, (b) the far-field influence of a group is approximated using a multipole expansion, and (c) a recursive procedure is applied to evaluate the required force or potential. Strictly speaking, a *cell* is a region of space containing a *group* of particles, but we use the terms interchangeably.

Two of the early treecodes were developed for gravitational simulations by Appel[4] and Barnes and Hut.[5] Both methods used monopole approximations and recursive evaluation procedures, but Appel allowed the cells to have arbitrary shape while Barnes and Hut used cubical cells. Greengard and Rokhlin[6,7] developed an approach called the Fast Multipole Method, in which they retained the use of cubical cells but employed higher order multipole approximations to improve the accuracy. In addition, they evaluated the far-field multipole approximations by converting them to local Taylor series. Much effort has been devoted to extending and optimizing the performance of treecode algorithms.[8–18] In recent years, treecodes have been successfully applied in classical molecular simulations,[19–30] as well as in quantum electronic structure calculations.[31–34]

Treecodes can be applied to speed up the evaluation of the total potential energy V by using the code to rapidly compute the terms V_i and then summing the results. This is a *particle–group* procedure because each term V_i represents the interaction between a particle $\mathbf{x}_i$ and the entire group $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. This is the most straightforward way of applying a treecode to compute V. Here, we consider an alternative approach based on the expression

$$V = \sum_{A,B} V_{AB}, \tag{4}$$

where A, B are appropriate groups of particles, and

$$V_{AB} = \sum_{\substack{\mathbf{x}_i \in A \\ \mathbf{x}_j \in B}} \frac{q_i q_j}{|\mathbf{x}_i - \mathbf{x}_j|^\nu} \tag{5}$$

is the energy due to interactions between the particles in group A and the particles in group B. In (5), it is understood that if the groups A, B intersect, then the sum omits the diagonal terms $\mathbf{x}_i = \mathbf{x}_j$. In case the groups are identical, the energy V_{AB} is denoted V_{AA}. If the groups A, B are well separated, then V_{AB} can be computed using a multipole approximation.[35] Otherwise, V_{AB} can be computed recursively by subdividing the groups and considering interactions among the subgroups. This leads to a *group–group* procedure for computing the total potential energy V. This approach is closely related to Appel's algorithm,[4] although his article dealt with computing individual particle forces rather than the total potential energy of the system.

Note that the particle–group expression (2)–(3) is a special case of the group–group expression (4)–(5), obtained by setting $A = \{\mathbf{x}_i\}$ and $B = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. The advantage of a group–group procedure is that it avoids computing the N individual terms V_i in favor of computing a potentially smaller number of terms V_{AB}. Despite this potential advantage, there has been relatively little investigation of this approach. In this context we note that Pérez–Jordá and Yang[25] described a group–group procedure for computing the total Coulomb potential energy, while Xue, Zall, and Pardalos[28] developed a version of Appel's algorithm for the Lennard–Jones potential.

This article describes a treecode algorithm for computing the total potential energy V using the group–group procedure outlined above. The algorithm treats the general nonbonded potential energy function in (1). A multipole approximation for V_{AB} is derived by Taylor expansion in Cartesian coordinates, and the necessary coefficients are computed using a recurrence relation. To gain efficiency, we employ several adaptive techniques developed for particle methods in fluid dynamics.[36, 37] The tree consists of nonuniform rectangular cells adapted to the particle distribution. An error bound is derived and used to select the order of approximation for each group–group interaction. A run-time choice is made between multipole approximation and direct summation based on estimates of the required execution times. Three parameters are specified by the user: ϵ, an accuracy parameter; $p_{\max}$, the maximum order of multipole approximation; and N_0, the maximum number of particles in a leaf of the tree.

We present results for the Coulomb and London dispersion potentials. The tests include dense and sparse particle distributions, and systems with neutral and nonneutral net charge. The proposed treecode is significantly faster than direct summation for systems having a large number of particles. In itself this is not a new finding, because existing treecodes already provide a speedup over direct summation, but there is still intense interest in optimizing performance within the class of treecode algorithms. The present work aims to contribute by calling attention to the advantages of a group–group procedure for evaluating the total potential energy in classical systems. Another important issue affecting performance is the choice of coordinate system. Spherical harmonics are often preferred for high-order multipole approximation of the Coulomb potential, but we show that Taylor approximation in Cartesian coordinates can be efficiently implemented for a general class of potential functions using a simple recurrence relation. Finally, we propose several new adaptive techniques for the tree structure and error control.

Multipole Approximation

This section derives a multipole approximation for the group–group potential energy V_{AB} defined in (5). We use Cartesian coordinates $\mathbf{x} = (x_1, x_2, x_3)$ and standard multiindex notation.[38] Figure 1 shows two groups of particles A, B, with $\mathbf{x}_i \in A$, $\mathbf{x}_j \in B$. The cell associated with a group of particles is the smallest rectangular box containing the particles (the sides of the box are parallel to the coordinate axes). Given a group A, the center (denoted $\mathbf{x}_A$) is the geometric center of the cell, and the radius (denoted r_A) is the distance from the center to a vertex of the cell. The distance between the cell centers is $R = |\mathbf{x}_A - \mathbf{x}_B|$.

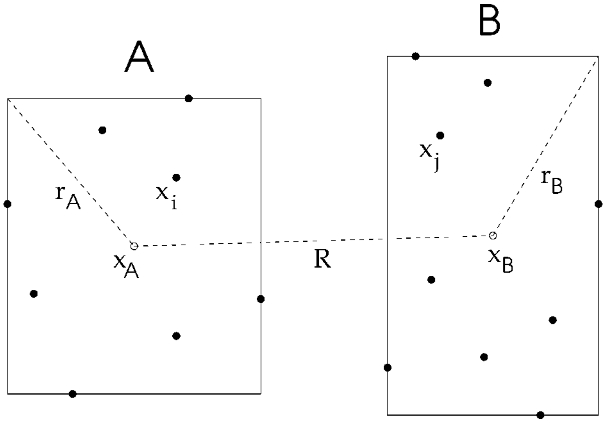

FIGURE 1. Two groups of particles A, B, with $\mathbf{x}_i \in A$, $\mathbf{x}_j \in B$. The cell associated with a group of particles is the smallest rectangular box containing the particles. The cell centers are $\mathbf{x}_A$, $\mathbf{x}_B$, the cell radii are r_A, r_B, and the distance between the cell centers is $R = |\mathbf{x}_A - \mathbf{x}_B|$.

Define the potential function

$$\phi_\nu(\mathbf{x}) = |\mathbf{x}|^{-\nu}. \tag{6}$$

Let $\mathbf{x}$, $\overline{\mathbf{x}}$ be two points and consider the pth order Taylor approximation of $\phi_\nu(\mathbf{x})$ at $\mathbf{x} = \overline{\mathbf{x}}$,

$$\phi_\nu(\mathbf{x}) \approx \sum_{||\mathbf{n}||=0}^{p} T_{\mathbf{n}}(\overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{n}}, \tag{7}$$

where $\mathbf{n} = (n_1, n_2, n_3)$, $||\mathbf{n}|| = n_1 + n_2 + n_3$, $T_{\mathbf{n}}(\overline{\mathbf{x}}) = \frac{1}{\mathbf{n}!} D^{\mathbf{n}}_{\mathbf{x}} \phi_\nu(\overline{\mathbf{x}})$ is the $\mathbf{n}$th Taylor coefficient, $\mathbf{n}! = n_1! n_2! n_3!$, $D^{\mathbf{n}}_{\mathbf{x}} = D^{n_1}_{x_1} D^{n_2}_{x_2} D^{n_3}_{x_3}$, and $(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{n}} = (x_1 - \overline{x}_1)^{n_1}(x_2 - \overline{x}_2)^{n_2}(x_3 - \overline{x}_3)^{n_3}$. Next, let $\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j$, $\overline{\mathbf{x}} = \mathbf{x}_A - \mathbf{x}_B$ in (7), apply the binomial formula, and substitute the result in (5), to obtain

$$V_{AB} \approx \sum_{||\mathbf{n}||=0}^{p} T_{\mathbf{n}}(\mathbf{x}_A - \mathbf{x}_B) \sum_{\mathbf{k} \leq \mathbf{n}} \binom{\mathbf{n}}{\mathbf{k}} (-1)^{||\mathbf{n}-\mathbf{k}||} m_A^{\mathbf{k}} m_B^{\mathbf{n}-\mathbf{k}}, \tag{8}$$

where $\mathbf{k} \leq \mathbf{n}$ means $k_i \leq n_i$ for $i = 1, 2, 3$, $\binom{\mathbf{n}}{\mathbf{k}} = \frac{\mathbf{n}!}{\mathbf{k}!(\mathbf{n}-\mathbf{k})!}$ is a binomial coefficient, and $m_A^{\mathbf{k}} = \sum_{\mathbf{x}_i \in A} q_i (\mathbf{x}_i - \mathbf{x}_A)^{\mathbf{k}}$ is the $\mathbf{k}$th multipole moment of group A (similarly for $m_B^{\mathbf{n}-\mathbf{k}}$).

Equation (8) is a pth order multipole approximation for V_{AB}. For $\nu = 1$, the term with $\|\mathbf{n}\| = 0$ is a monopole–monopole interaction between the groups, the terms with $\|\mathbf{n}\| = 1$ are monopole–dipole interactions, the terms with $\|\mathbf{n}\| = 2$ are monopole–quadrupole or dipole–dipole interactions, and similarly for higher order terms.[35] Several features contribute to an efficient computation of the expression in (8). The cell moments $m_A^{\mathbf{k}}$ can be precomputed and stored, and then used as required; they do not have to be recomputed if group A appears in more than one term V_{AB} in (4). Also, the Taylor coefficients $T_{\mathbf{n}}(\mathbf{x}_A - \mathbf{x}_B)$ depend only on the group centers and are independent of the number of particles in the groups. Explicit formulas for the high-order Taylor coefficients are cumbersome, and in the next section we derive a recurrence relation permitting rapid computation of these coefficients.

Recurrence Relation

First note that $\phi_\nu(\mathbf{x})$ satisfies the differential equation

$$|\mathbf{x}|^2 D_{x_1}\phi_\nu(\mathbf{x}) + \nu x_1 \phi_\nu(\mathbf{x}) = 0. \tag{9}$$

Applying the operator $D_{x_1}^{n_1-1}$ and using Leibniz's rule for differentiating a product, we obtain

$$\begin{aligned}|\mathbf{x}|^2 D_{x_1}^{n_1}\phi_\nu(\mathbf{x}) &+ (2n_1+\nu-2)x_1 D_{x_1}^{n_1-1}\phi_\nu(\mathbf{x}) \\ &+ (n_1-1)(n_1+\nu-2)D_{x_1}^{n_1-2}\phi_\nu(\mathbf{x}) = 0.\end{aligned} \tag{10}$$

Next, we apply the operator $D_{x_2}^{n_2}D_{x_3}^{n_3}$ to obtain

$$\begin{aligned}|\mathbf{x}|^2 D_{\mathbf{x}}^{\mathbf{n}}\phi_\nu(\mathbf{x}) &+ 2n_3x_3 D_{\mathbf{x}}^{\mathbf{n}-\mathbf{e}_3}\phi_\nu(\mathbf{x}) \\ &+ n_3(n_3-1)D_{\mathbf{x}}^{\mathbf{n}-2\mathbf{e}_3}\phi_\nu(\mathbf{x}) \\ &+ 2n_2x_2 D_{\mathbf{x}}^{\mathbf{n}-\mathbf{e}_2}\phi_\nu(\mathbf{x}) + n_2(n_2-1)D_{\mathbf{x}}^{\mathbf{n}-2\mathbf{e}_2}\phi_\nu(\mathbf{x}) \\ &+ (2n_1+\nu-2)x_1 D_{\mathbf{x}}^{\mathbf{n}-\mathbf{e}_1}\phi_\nu(\mathbf{x}) \\ &+ (n_1-1)(n_1+\nu-2)D_{\mathbf{x}}^{\mathbf{n}-2\mathbf{e}_1}\phi_\nu(\mathbf{x}) = 0,\end{aligned} \tag{11}$$

where $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ are the standard Cartesian basis vectors. Dividing (11) by $\mathbf{n}!$ and setting $T_{\mathbf{n}} = T_{\mathbf{n}}(\mathbf{x}) = \frac{1}{\mathbf{n}!}D_{\mathbf{x}}^{\mathbf{n}}\phi_\nu(\mathbf{x})$, we obtain

$$\begin{aligned}|\mathbf{x}|^2 T_{\mathbf{n}} + 2\sum_{i=1}^{3} x_i T_{\mathbf{n}-\mathbf{e}_i} &+ \sum_{i=1}^{3} T_{\mathbf{n}-2\mathbf{e}_i} \\ &+ \frac{\nu-2}{n_1}(x_1 T_{\mathbf{n}-\mathbf{e}_1} + T_{\mathbf{n}-2\mathbf{e}_1}) = 0.\end{aligned} \tag{12}$$

Two more equations are obtained by replacing the index 1 in the last term of (12) by 2 and 3. Multiplying these equations by n_1, n_2, n_3, respectively, and summing the results, we obtain

$$\begin{aligned}||\mathbf{n}|| \cdot |\mathbf{x}|^2 T_{\mathbf{n}} &+ \big(2||\mathbf{n}|| + \nu - 2\big)\sum_{i=1}^{3} x_i T_{\mathbf{n}-\mathbf{e}_i} \\ &+ \big(||\mathbf{n}|| + \nu - 2\big)\sum_{i=1}^{3} T_{\mathbf{n}-2\mathbf{e}_i} = 0.\end{aligned} \tag{13}$$

Equation (13) is the desired recurrence relation for the Taylor coefficients $T_{\mathbf{n}}$. It is understood that $T_{\mathbf{n}} = 0$ whenever an index n_i is negative. In practice, the recurrence relation is applied with $\mathbf{x}$ replaced by $\mathbf{x}_A - \mathbf{x}_B$ for specified groups A, B.

There is a connection between the recurrence relation (13) and a result from the theory of orthogonal polynomials.[39] The Gegenbauer polynomials $C_n^{\nu/2}(y)$ of degree n and order $\nu/2$ satisfy the recurrence relation

$$\begin{aligned}nC_n^{\nu/2}(y) - (2n+\nu-2)\,y\,C_{n-1}^{\nu/2}(y) \\ + (n+\nu-2)C_{n-2}^{\nu/2}(y) = 0,\end{aligned} \tag{14}$$

with $C_0^{\nu/2}(y) = 1$, $C_1^{\nu/2}(y) = \nu y$. Note the similarity between the three-dimensional recurrence relation for $T_{\mathbf{n}}$ in (13) and the one-dimensional recurrence relation for $C_n^{\nu/2}(y)$ in (14). This similarity is not surprising, because the Taylor coefficients arise by expanding the potential function $\phi_\nu(\mathbf{x})$ in Cartesian coordinates about a point $\mathbf{x} = \bar{\mathbf{x}}$, while the Gegenbauer polynomials arise by expanding the same function with respect to a radial spherical coordinate,

$$\begin{aligned}\phi_\nu(\mathbf{x}) &= \sum_{||\mathbf{n}||=0}^{\infty} T_{\mathbf{n}}(\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{n}} \\ &= \sum_{n=0}^{\infty} C_n^{\nu/2}\left(\frac{\bar{\mathbf{x}}}{|\bar{\mathbf{x}}|} \cdot \frac{\bar{\mathbf{x}}-\mathbf{x}}{|\bar{\mathbf{x}}-\mathbf{x}|}\right)\frac{|\mathbf{x}-\bar{\mathbf{x}}|^n}{|\bar{\mathbf{x}}|^{n+\nu}}.\end{aligned} \tag{15}$$

The two expansions for $\phi_\nu(\mathbf{x})$ in (15) have been used before in treecode simulations,[20, 22, 26] but as far as we know, the three-dimensional recurrence relation (13) has not previously appeared.

The pth order multipole approximation for V_{AB} in (8) requires $O(p^3)$ Taylor coefficients, and these can be computed in $O(p^3)$ operations using the recurrence relation (13). Then assuming the cell moments are available, the approximation (8) can be evaluated in $O(p^6)$ operations. The high cost of this step for large values of p is a result of using Taylor approximation in Cartesian coordinates. The treecode algorithm uses adaptive techniques to reduce the cost associated with this step; for example, in the case of the Coulomb potential, the order p is chosen to be the minimum value satisfying a certain accuracy criterion. In the next section we derive an error bound that defines this criterion.

Error Bound

Our immediate goal is to bound the error in the pth order Taylor approximation of $\phi_\nu(\mathbf{x})$ in (7). First note that the sum of the nth order terms in (15) is

$$\sum_{\|\mathbf{n}\|=n} T_{\mathbf{n}}(\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{n}} = C_n^{\nu/2}\left(\frac{\overline{\mathbf{x}}}{|\overline{\mathbf{x}}|}\cdot\frac{\overline{\mathbf{x}}-\mathbf{x}}{|\overline{\mathbf{x}}-\mathbf{x}|}\right)\frac{|\mathbf{x}-\overline{\mathbf{x}}|^n}{|\overline{\mathbf{x}}|^{n+\nu}}. \tag{16}$$

This can also be checked explicitly using the recurrence relations for $T_{\mathbf{n}}(\mathbf{x})$ and $C_n^{\nu/2}(y)$. In our application we are concerned with two groups of particles A, B with $\mathbf{x}_i \in A, \mathbf{x}_j \in B$, $\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j$, $\overline{\mathbf{x}} = \mathbf{x}_A - \mathbf{x}_B$ (see Fig. 1). This yields

$$\frac{|\mathbf{x}-\overline{\mathbf{x}}|}{|\overline{\mathbf{x}}|} = \frac{|\mathbf{x}_i - \mathbf{x}_j - (\mathbf{x}_A - \mathbf{x}_B)|}{|\mathbf{x}_A - \mathbf{x}_B|} \le \frac{r_A + r_B}{R} \equiv r, \tag{17}$$

where r_A, r_B are the cell radii and $R = |\mathbf{x}_A - \mathbf{x}_B|$ is the distance between the cell centers. Assuming $r < 1$, and using the fact that $|C_n^{\nu/2}(y)| \le \Gamma(\nu+n)/\Gamma(\nu)\,\Gamma(n+1)$ for $|y| \le 1$, we can bound the sum of the terms in (15) for $n \ge p+1$ to obtain

$$\sum_{n=p+1}^{\infty}\sum_{\|\mathbf{n}\|=n}\left|T_{\mathbf{n}}(\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{n}}\right| \le \sum_{n=p+1}^{\infty}\frac{\Gamma(\nu+n)}{\Gamma(\nu)\,\Gamma(n+1)}\frac{r^n}{R^\nu}$$
$$= \frac{1}{R^\nu}\frac{1}{\Gamma(\nu)}\frac{d^{\nu-1}}{dr^{\nu-1}}\left(\frac{r^{p+\nu}}{1-r}\right). \tag{18}$$

The expression on the right side of (18) is the desired error bound. Note that it depends on the exponent ν of the potential function, the order of approximation p, and the geometric parameters r, R specifying the size and separation of the groups A, B.

In some applications it may be desireable to use a smooth interaction potential. For example, consider the function

$$\phi_{\nu,\delta}(\mathbf{x}) = \left(|\mathbf{x}|^2 + \delta^2\right)^{-\nu/2}, \tag{19}$$

where δ is a smoothing parameter. This type of potential is used in computational fluid dynamics[36, 37, 40, 41] and a similar expression was proposed to smooth the energy surface in global optimization techniques for molecular conformation.[44] It can be shown that the Taylor coefficients of the smooth potential (19) satisfy the recurrence relation (13) with $|\mathbf{x}|^2$ replaced by $|\mathbf{x}|^2 + \delta^2$. Also, the error bound (18) is still valid. Hence, the present treecode algorithm can be applied to particle systems governed by this type of smooth interaction potential.

Accuracy Criterion

We require the error bound on the right side of (18) to be less than a user-specified parameter ϵ. If this criterion is satisfied, then the groups A, B are effectively well-separated and V_{AB} can be evaluated using the multipole approximation (8). More specifically, we employ the following strategies depending on the type of interaction.

1. *Long-range interaction:* The primary example is the Coulomb potential, $\nu = 1$, for which the criterion is

$$\frac{1}{R}\frac{r^{p+1}}{1-r} \le \epsilon. \tag{20}$$

   Given a pair of groups A, B with geometric parameters r, R, the code selects the minimum order p satisfying (20).
2. *Short-range interaction:* The London dispersion potential, $\nu = 6$, is the main example. In this case the Taylor approximation (7) converges slowly as the order p increases.[22] Hence, we fix the value $p = 2$ and then the criterion becomes

$$\frac{1}{R^6}\frac{1}{\Gamma(6)}\frac{d^5}{dr^5}\left(\frac{r^8}{1-r}\right) \le \epsilon. \tag{21}$$

   Although the order p is fixed, the algorithm can still reduce the parameter r to satisfy the criterion (21).

In summary, if the accuracy criterion ((20) or (21)) is satisfied, then V_{AB} can be evaluated using the multipole approximation with the specified order p. If the criterion is not satisfied, the code subdivides one of the groups; this has the effect of reducing the parameter r. This continues until the criterion is satisfied or until the groups are so small that direct summation is faster than multipole approximation. The procedure will be described below in more detail.

Treecodes generally use a fixed order of approximation, although we note that Petersen, Soelvason, Perram, and Smith[10] obtained a speedup using variable-order approximation in their implementation of the Fast Multipole Method. In the present approach, variable order approximation for the Coulomb potential is especially effective in combination with nonuniform adaptive cells. The next section describes the procedure for constructing these cells.

Tree Construction

The tree construction procedure divides the set of particles into a nested hierarchy of cells. Most treecodes use an oct-tree structure in which the cells on level L are uniform cubes with side length proportional to $(\frac{1}{2})^L$. In some implementations[5, 8] a cell is left undivided if it contains fewer than a user-specified number of particles N_0. This yields an adaptive tree that takes advantage of any localized gaps in the particle distribution. The present algorithm follows this approach, but enhances the adaptivity by using nonuniform rectangular cells instead of cubes. The root cell (level $L = 0$) is the smallest rectangular box containing the entire set of particles. The root is divided in half in each coordinate direction, resulting in eight subcells or children. Before further subdivision, each subcell is shrunk into the smallest rectangular box containing its particles; these cells form the next level in the tree. The process continues until the number of particles in a cell is less than N_0; these cells form the leaves of the tree. Figure 2 shows an example with $N_0 = 20$ for a dense random set of particles in two-dimensional space.

Shrinking the cells this way is not costly, and the resulting tree is well adapted to the particle distribution. The cell radii on a given level are smaller than they would be without shrinking. Hence, the accuracy criterion can be satisfied using a lower order approximation, and this leads to a speedup in execution time. This is the sense in which variable-order approximation is especially effective in combination with nonuniform adaptive cells.

Note that Clarke and Tutty[42] developed a treecode with rectangular cells in which all the cells on a given level have the same number of particles. The aim was to facilitate load balancing on a parallel computer, but there is a tradeoff because extra computation is required to determine where to split the cells. Also, their scheme might split some natural particle clusters that would be left intact by the present scheme, and this can adversely affect the code's performance.

Potential Energy Evaluation

The procedure for evaluating the total potential energy uses the two recursive functions shown in Figure 3. The first function, `V1`(A), returns the energy V_{AA} due to interactions among the particles in cell A. The initial call is `V1`(*root*). If A is a leaf, V_{AA} is computed by direct summation. If A is not a leaf, V_{AA} is computed by summing the group–group interactions among the children of A. The latter step uses the second function, `V2`(A, B), which returns the energy V_{AB} due to interactions between the particles in cell A and the particles in cell B.

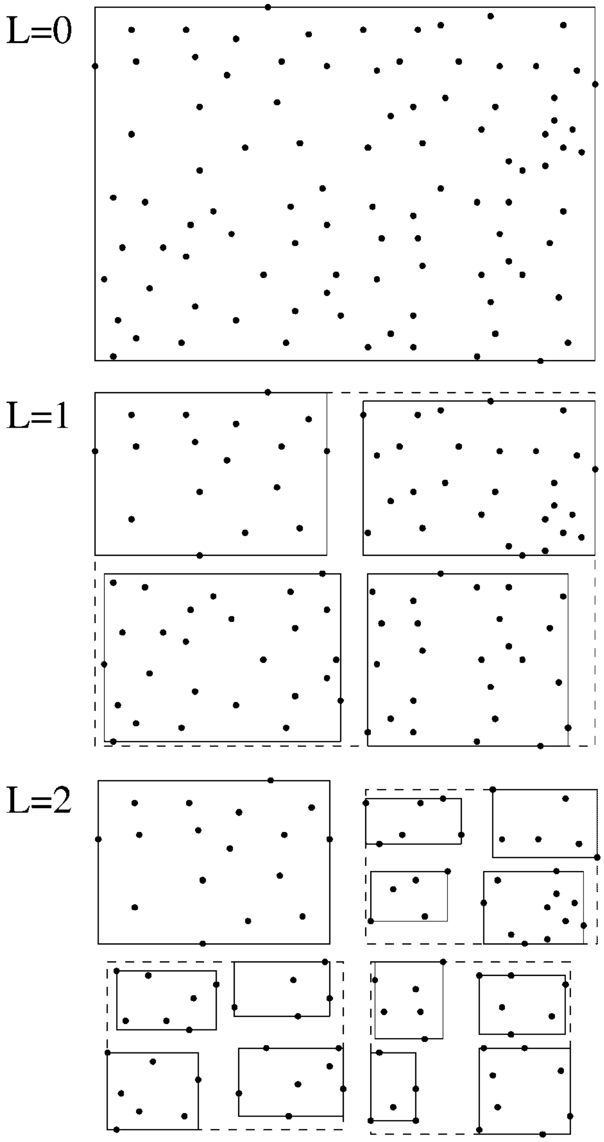

FIGURE 2. Example of tree construction for particles in two-dimensional space; three levels, $L = 0, 1, 2$; $N_0 = 20$. A cell is the smallest rectangular box containing its particles. (—):cells on a given level, (– – –): cells on the previous level.

Consider a call to `V2`(A, B). Suppose first that the accuracy criterion (20) or (21) is satisfied. In

```
(a) V1($A$)
    if $A$ is a leaf
      compute $V_{AA}$ by direct summation
    else
      for $i = 1$ to 8
        V1($A$.child[i])
        for $j = i + 1$ to 8
          V2($A$.child[i], $A$.child[j])
      compute $V_{AA}$ by summing results
    return $V_{AA}$

(b) V2($A, B$)
    if the accuracy criterion is satisfied
      if direct_is_faster($A, B$)
        compute $V_{AB}$ by direct summation
      else
        compute $V_{AB}$ by multipole approximation
    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 $k = 1$ to 8
          V2($A$, $B$.child[k])
        compute $V_{AB}$ by summing results
      else
        for $k = 1$ to 8
          V2($A$.child[k], $B$)
        compute $V_{AB}$ by summing results
    return $V_{AB}$
```

FIGURE 3. Recursive functions used in evaluating the potential energy. (a) `V1`(A) returns the energy V_{AA} due to interactions among the particles in cell A; (b) `V2`(A, B) returns the energy V_{AB} due to interactions between the particles in cell A and the particles in cell B.

TABLE I.
Execution Time Required to Evaluate the Group–Group Potential Energy V_{AB}.

p	T_{ma}	$N_A \cdot N_B$	T_{ds}
0	2.9	1	1.3
1	3.4	4	4.6
2	5.7	9	8.2
3	8.9	25	19.5
4	17.6	36	27.3
5	25.9	64	49.2
6	46.8	100	75.2
7	79.3	169	124.5
8	122.7	289	208.5
9	219.8	400	286.7
10	296.2	441	315.9

This table refers to the case of the Coulomb potential, $\nu = 1$.
T_{ma}: time required for pth order multipole approximation.
T_{ds}: time required for direct summation with $N_A \cdot N_B$ particles.
Times are in units of μs on a Sun UltraSPARC-II workstation.

this case, a pth order multipole approximation can be used, but the code checks to see whether direct summation is faster. If so, then V_{AB} is computed by direct summation, and if not, then V_{AB} is computed by multipole approximation. The decision is made with the help of the function `direct_is_faster`(A, B), which accesses a lookup table of precomputed execution times; it returns `true` if direct summation is faster and `false` otherwise. The execution time for direct summation depends on the product $N_A \cdot N_B$ of the number of particles in group A and group B, and the execution time for multipole approximation depends on the order p. Table I is the lookup table used here with $\nu = 1$.

Suppose, on the other hand, that the accuracy criterion is not satisfied. If both cells are leaves of the tree, then V_{AB} is computed by direct summation. If one cell is a leaf and the other is not, then interactions are computed between the leaf and the children of the other cell. If neither cell is a leaf, then interactions are computed between the smaller cell and the children of the larger cell.

This type of recursive evaluation procedure was introduced by Appel.[4] Note that the recursion terminates if the accuracy criterion is satisfied for a given pair of cells. As a result, the number of group–group interactions V_{AB} entering into the computation of V is not fixed in advance but is instead determined adaptively.

Implementation Details

The algorithm was coded in the C programming language, and the computations were performed in double precision on a Sun UltraSPARC-II workstation. The code first constructs the tree and then evaluates the potential energy by calling `V1(`*root*`)`. The cell moments $m_A^{\mathbf{n}}$ for $||\mathbf{n}|| \leq p_{\max}$ are computed and stored during tree construction. In the simulations below, the maximum order of multipole approximation is $p_{\max} = 10$ for the Coulomb potential and the order is fixed at $p = 2$ for the London dispersion potential. The size of the system ranges from $N = 500$ to $N = 128,000$ particles. The enclosed volume varies with N to ensure that the particle density is fixed. The maximum number of particles in a leaf is $N_0 = 30$, 10, 20, respectively, in the three test cases. In each case, results are presented for three values of the accuracy parameter, $\epsilon = 10^{-3}, 10^{-5}, 10^{-7}$. A direct summation code (using pairwise symmetry) is the benchmark for comparing errors and execution times. In the treecode, loops for the recurrence relation and multipole approximation are expanded to in-line code using a separate utility program. The particles are stored in a linear array with group members appearing in consecutive locations. The execution times in Table I were precomputed; the timings depend on the computer hardware and coding of the algorithms so each user must prepare their own version of Table I. The expression on the left side of (21) is interpolated from a look-up table containing several values of the parameter *r*.

Numerical Results

The first test case is the Coulomb potential for a dense random set of particles. The charges are $q_i = \pm 1$ with equal probability. Figure 4a plots the execution time required for direct summation and the treecode as a function of the number of particles N. As expected, the direct summation time increases at the rate $O(N^2)$. Direct summation is faster for small systems, and the treecode is faster for large systems. The crossover point depends on the accuracy parameter ϵ; it varies from $N = 1000$ for $\epsilon = 10^{-3}$ to $N = 8000$ for $\epsilon = 10^{-7}$. Past the crossover point, the treecode execution time increases at a rate close to but slightly greater than $O(N)$. For a given value of N, the treecode execution time increases as ϵ is reduced. Figure 4b plots the relative error in the energy computed by the treecode. For a given value of ϵ, the error amplitude is close to ϵ, and it varies little over the range of N. For a given value of N, the error decreases as ϵ is reduced.

The second test case is the London dispersion potential for the same particle distribution as above but with uniform weights $q_i = 1$. The results, shown in Figure 5, display the same trends as in the first test case, but some details are different. In Figure 5a, the execution time for direct summation is roughly the same as in Figure 4a, but the treecode time for large N is less than in Figure 4a. The crossover point varies from $N = 1000$ for $\epsilon = 10^{-3}$ to $N = 2000$ for $\epsilon = 10^{-7}$. The treecode time in Figure 5a increases at a rate close to $O(N)$ over the entire range of N for all three values of ϵ. The relative error in Figure 5b has similar qualitative behavior as in Figure 4b with respect to the accuracy parameter ϵ and the number of particles N.

The third test case is the Coulomb potential for particles lying on a B-spline curve representing a supercoiled DNA molecule.[43] The curve, shown schematically in Figure 6, is defined in Figure 3b of ref. 43. The particles represent phosphate groups with uniform charge and uniform spacing along the curve; the latter condition was enforced using the algorithm in ref. 43. In contrast to the dense random set of particles in the first two test cases, this is a sparse localized distribution in three-dimensional space. The results are shown in Figure 7. The treecode execution time in Figure 7a is close to $O(N)$

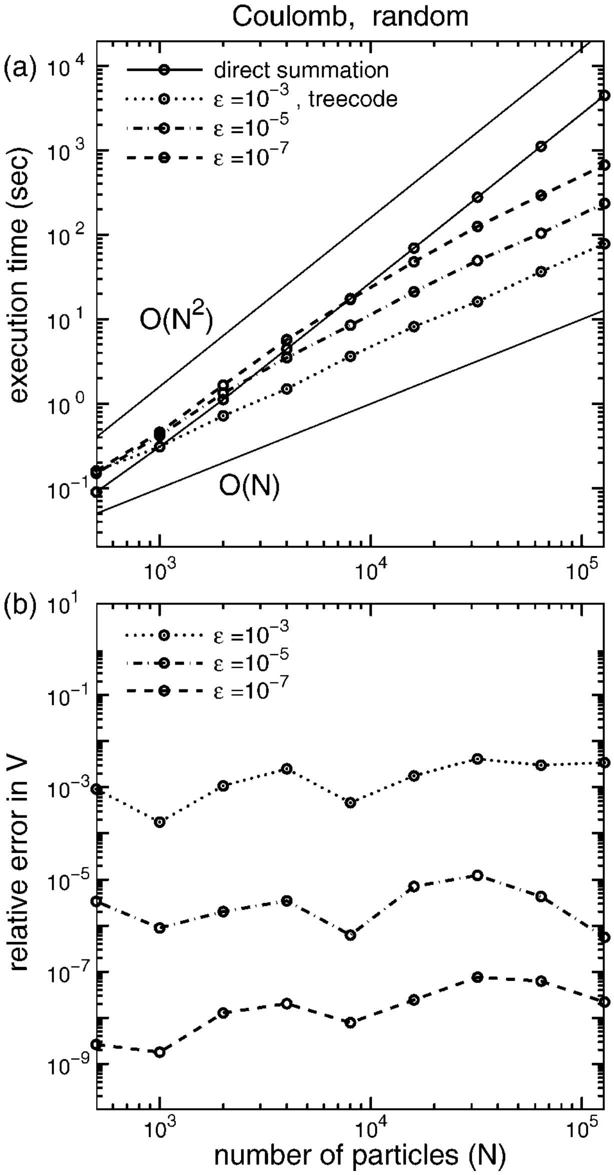

FIGURE 4. First test case: Coulomb potential, random particle distribution, random charges ($q_i = \pm 1$ with equal probability). (a) Execution time for direct summation, treecode with accuracy parameter $\epsilon = 10^{-3}, 10^{-5}, 10^{-7}$; (b) relative error in potential energy V computed by the treecode.

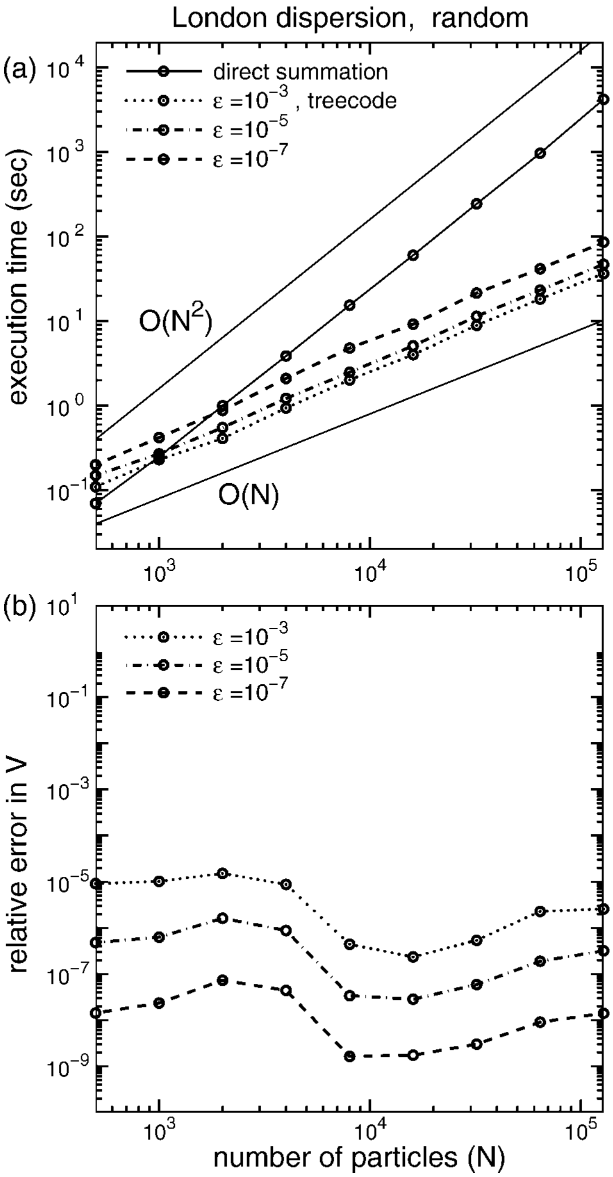

FIGURE 5. Second test case: London dispersion potential, random particle distribution, uniform weights ($q_i = 1$). (a) Execution time for direct summation, treecode with accuracy parameter $\epsilon = 10^{-3}$, 10^{-5}, 10^{-7}; (b) relative error in potential energy V computed by the treecode.

over the entire range of N, and it increases only slightly as ϵ is reduced. The crossover point is below $N = 1000$ for all three values of ϵ. The relative error in Figure 7b increases slightly with N, but for a given value of N the error decreases as ϵ is reduced.

In all three test cases there is a systematic variation in the error as a function of the number of particles N, either periodic, or increasing, or a combination of these. The source of this variation is not known. Aside from this, the results indicate that the treecode algorithm is performing generally as expected in terms of execution time and accuracy.

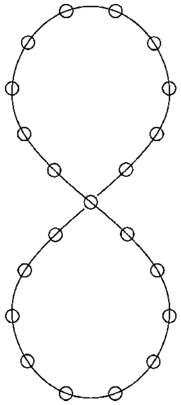

FIGURE 6. Schematic diagram of third test case, particles on a B-spline curve representing a supercoiled DNA molecule.[43] The particles have uniform charge and uniform spacing along the curve.

Table II presents the speedup, defined as the ratio of execution times for direct summation and the treecode. In each test case, the speedup improves if the system becomes larger (increasing N) or if less accuracy is requested (increasing ϵ); this is characteristic of treecodes. Comparing case 1 and case 2 shows the effect of the potential function; the speedup is better for a short-range potential than for a long-range potential. Comparing case 1 and case 3 shows the effect of the particle distribution; the speedup is better for a sparse localized set than for a dense random set (this was also observed in previous studies using adaptive treecodes[8, 30]).

Concluding Remarks

We presented a treecode algorithm for rapid computation of the nonbonded potential energy in classical molecular systems. The algorithm treats a general form of pairwise particle interaction with the Coulomb and London dispersion potentials as special cases. The energy is computed as a sum of group–group interactions V_{AB} using a variant of Appel's recursive strategy.[4] Several adaptive techniques are employed to reduce the execution time. These include an adaptive tree with nonuniform rectangular cells, variable order multipole approximation, and a run-time choice between direct summation and multipole approximation to evaluate V_{AB}. The multipole approximation is de-

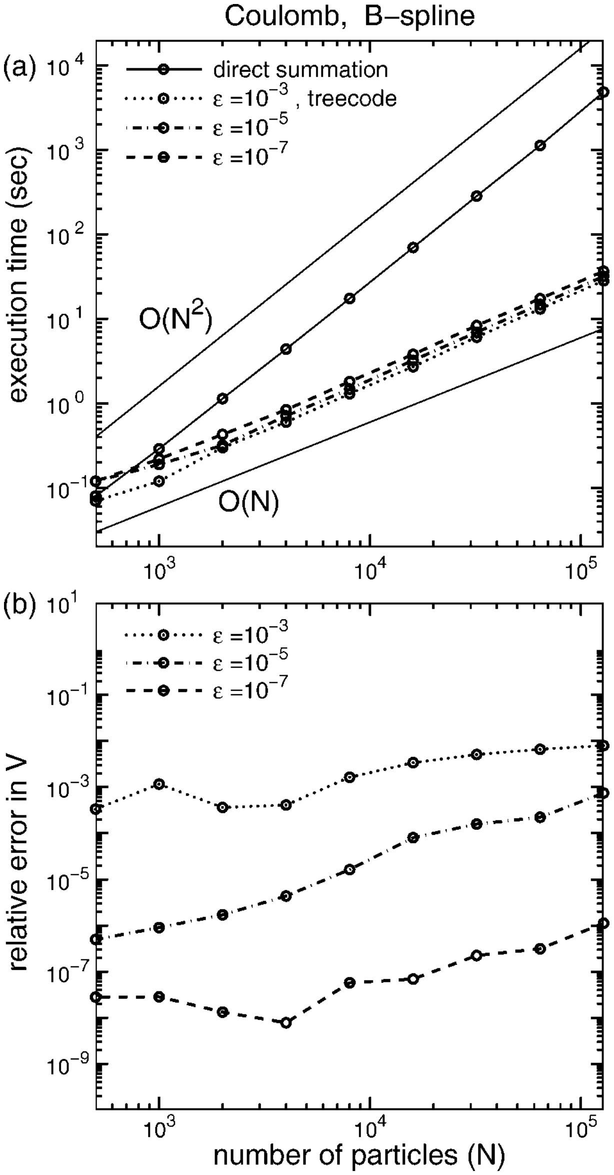

FIGURE 7. Third test case: Coulomb potential, particles uniformly spaced on a B-spline curve (Fig. 6), uniform charges ($q_i = 1$). (a) Execution time for direct summation, treecode with accuracy parameter $\epsilon = 10^{-3}$, 10^{-5}, 10^{-7}; (b) relative error in potential energy V computed by the treecode.

rived by Taylor expansion in Cartesian coordinates, and the necessary coefficients are computed using a recurrence relation. An error bound was derived and used to select the order of approximation for V_{AB}.

Tests were conducted for two types of particle distributions—a dense random set and a sparse localized set. For the London dispersion potential on a dense set (case 2) and the Coulomb potential on a sparse set (case 3), the execution time was close to $O(N)$. The performance was less favorable for the Coulomb potential on a dense set (case 1), but a significant speedup over direct summation was still achieved. It is well known that treecodes perform better than direct summation for large systems, but there is still intense interest in optimizing performance within the family of treecode algorithms. The aim of this work is to call attention to the advantages of a group–group procedure for computing the total potential energy in classical systems, to demonstrate the feasibility of using Taylor approximation in Cartesian coordinates, and to introduce several adaptive techniques. This approach has the following advantages: it is applicable to a wide class of potential functions, the analytical basis is relatively simple, and the enhanced adaptivity ensures good performance for a variety of systems.

The most expensive step in the algorithm is evaluating the pth order multipole approximation for V_{AB} in (8); due to the use of a three-dimensional Taylor expansion the operation count is $O(p^6)$. The code employs several techniques to reduce the cost associated with this step. First, a low-order p is used whenever possible; the minimum order p satisfying the accuracy criterion is used for the Coulomb potential, and the fixed order $p = 2$ is used for the London dispersion potential. Second, the recursion terminates whenever the accuracy criterion is satisfied and this limits the number of times the $O(p^6)$ step is computed. Finally, the run-time choice between direct summation and multipole approximation eliminates some of the $O(p^6)$ computations.

The Fast Multipole Method[6, 7] can also be applied to compute the total potential energy, and in this case, the most expensive step is the multipole-to-local transformation with an $O(p^4)$ operation count. Analytical techniques have been developed to reduce the cost of this step by using rotation-based translations,[16–18] the fast Fourier transform,[14] or plane wave translations.[18] The operation count has been reduced to $O(p^2)$, but these codes generally use a fixed-order p, and they traverse the tree from root to leaves with a fixed interaction list at each level. In contrast, the present approach retains an $O(p^6)$ operation count, but uses adaptive techniques to lower the order p and to reduce the number of times the costly step is computed. There are various tradeoffs between these alternatives in terms of speedup, accuracy, memory usage, range of applicability, and ease of implementation, but it is beyond the scope of this work to make a detailed quantitative comparison; this is a topic for future investigation.

TABLE II.
Speedup (Ratio of Execution Times for Direct Summation and the Treecode).

	Case 1			Case 2			Case 3		
N	$\epsilon = 10^{-3}$	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}	10^{-3}	10^{-5}	10^{-7}
500	0.6	0.6	0.6	1.6	1.2	0.9	1.1	0.7	0.7
1000	1.0	0.7	0.7	3.0	2.6	1.6	2.4	1.5	1.3
2000	1.6	0.8	0.7	6.6	4.9	3.1	3.8	3.6	2.7
4000	3.0	1.3	0.8	11.4	8.8	5.1	7.3	6.2	5.2
8000	4.8	2.1	1.0	21.0	17.1	8.9	13.4	11.8	9.6
16,000	8.5	3.3	1.5	42.2	33.1	18.4	25.7	21.7	18.4
32,000	17.2	5.6	2.2	76.1	59.4	31.6	46.7	40.8	33.9
64,000	30.5	10.7	3.8	148.8	116.4	65.2	86.0	76.1	65.0
128,000	57.1	18.9	6.7	299.4	232.2	127.2	171.3	151.9	132.0

N: number of particles, $\epsilon = 10^{-3}$, 10^{-5}, 10^{-7}: accuracy parameter.
Case 1: Coulomb potential, random particles, random charges ($q_i = \pm 1$).
Case 2: London dispersion potential, random particles, uniform weight.
Case 3: Coulomb potential, particles on a B-spline curve, uniform charge.

Acknowledgments

The authors are grateful to Professor Samuel Krimm and his research group for helpful discussions, and to the referees for suggestions that led to improvements in this article.

References

1. Vásquez, M.; Némethy, G.; Scheraga, H. A. Chem Rev 1994, 94, 2183.
2. Allen, M. P.; Tildesley, D. J. Computer Simulation of Liquids; Oxford University Press: Oxford, 1987.
3. Hockney, R. W.; Eastwood, J. W. Computer Simulation Using Particles; IOP Publishing: Bristol, 1988.
4. Appel, A. W. SIAM J Sci Stat Comput 1985, 6, 85.
5. Barnes, J.; Hut, P. Nature 1986, 324, 446.
6. Greengard, L.; Rokhlin, V. J Comput Phys 1987, 73, 325.
7. Greengard, L.; Rokhlin, V. Chem Scripta 1989, 29A, 139.
8. Carrier, J.; Greengard, L.; Rokhlin, V. SIAM J Sci Stat Comput 1988, 9, 669.
9. Anderson, C. SIAM J Stat Sci Comput 1992, 13, 923.
10. Petersen, H. G.; Soelvason, D.; Perram, J. W.; Smith, E. R. J Chem Phys 1994, 101, 8870.
11. Salmon, J. K.; Warren, M. S. J Comput Phys 1994, 111, 136.
12. White, C. A.; Head-Gordon, M. J Chem Phys 1994, 101, 6593.
13. Warren, M. S.; Salmon, J. K. Comput Phys Commun 1995, 87, 266.
14. Elliott, W. D.; Board, J. A., Jr. SIAM J Sci Comput 1996, 17, 398.
15. Wang, H. Y.; LeSar, R. J Chem Phys 1996, 104, 4173.
16. White, D. A.; Head-Gordon, M. J Chem Phys 1996, 105, 5061.
17. Greengard, L.; Rokhlin, V. Acta Numerica 1997, 229.
18. Cheng, H.; Greengard, L.; Rokhlin, V. J Comput Phys 1999, 155, 468.
19. Board, J. A., Jr.; Causey, J. W.; Leathrum, J. F., Jr.; Windemuth, A.; Schulten, K. Chem Phys Lett 1992, 198, 89.
20. Ding, H.-Q.; Karasawa, N.; Goddard, W. A. III J Chem Phys 1992, 97, 4309.
21. Saito, M. Mol Simulat 1992, 8, 321.
22. Elliott, W. D.; Board, J. A., Jr. Tech Rep 94-005, Duke Univ Dept EECS, 1994, http://www.ee.duke.edu/research/SciComp/Papers/TR94-005.html.
23. Mathiowetz, A. M.; Jain, A.; Karasawa, N.; Goddard, W. A. III Proteins Struct Funct Genet 1994, 20, 227.
24. Shimada, J.; Kaneko, H.; Takada, T. J Comput Chem 1994, 15, 28.
25. Pérez-Jordá, J. M.; Yang, W. Chem Phys Lett 1995, 247, 484.
26. Fenley, M. O.; Olson, W. K.; Chua, K.; Boschitsch, A. H. J Comput Chem 1996, 17, 976.
27. Niedermeier, C.; Tavan, P. Mol Simulat 1996, 17, 57.
28. Xue, G. L.; Zall, A. J.; Pardalos, P. M. In DIMACS Series in Discrete Mathematics and Theoretical Computer Science; Pardalos, P. M.; Shalloway, D.; Xue, G., Eds.; American Mathematical Society: Providence, 1996, p. 237, vol. 23.
29. Eichinger, M.; Grubmüller, H.; Heller, H.; Tavan, P. J Comput Chem 1997, 18, 1729.
30. Boschitsch, A. H.; Fenley, M. O.; Olson, W. K. J Comput Chem 1999, 151, 212.
31. Challacombe, M.; Schwegler, E.; Almlöf, J. J Chem Phys 1996, 104, 4685.
32. Strain, M. C.; Scuseria, G. E.; Frisch, M. J Science 1996, 271, 51.

33. Challacombe, M.; Schwegler, E. J Chem Phys 1997, 106, 5526.
34. Schwegler, E.; Challacombe, M.; Head-Gordon, M. J Chem Phys 1997, 106, 9708.
35. Stone, A. J. The Theory of Intermolecular Forces; Oxford University Press: Oxford, 1996.
36. Draghicescu, C.; Draghicescu, M. J Comput Phys 1995, 116, 69.
37. Lindsay, K. Ph.D. Thesis; University of Michigan, 1997.
38. John, F. Partial Differential Equations; Springer: New York, 1971, p. 52, 3rd ed.
39. Andrews, G. E.; Askey, R.; Roy, R. Special Functions; Cambridge University Press: Cambridge, UK, 1999, p. 302.
40. Krasny, R. J Fluid Mech 1987, 184, 132.
41. Brady, M.; Leonard, A.; Pullin, D. I. J Comput Phys 1998, 146, 520.
42. Clarke, N. R.; Tutty, O. R. Comput Fluids 1994, 23, 751.
43. Hao, M.-H.; Olson, W. K. Macromolecules 1989, 22, 3292.
44. Shao, C.-S.; Byrd, R. H.; Eskow, E.; Schnabel, R. B. J Global Opt, to appear.