Large scale simulation of macromolecules in solution: Combining the periodic fast multipole method with multiple time step integrators

Francisco Figueirido and Ronald M. Levy[a)]
Department of Chemistry, Rutgers, The State University of New Jersey, Piscataway, New Jersey 08855-0939

Ruhong Zhou and B. J. Berne
Department of Chemistry and Center for Biomolecular Simulation, Columbia University, New York, New York 10027

(Received 31 December 1996; accepted 13 March 1997)

Large scale simulations of macromolecules in solution that do not suffer from artifacts arising from force truncations are becoming feasible. New force evaluation algorithms such as the fast multipole method (FMM) and multiple time scale integration methods such as the reversible reference system propagator algorithm (r-RESPA) have been combined and used to perform fast and stable simulations of large macromolecular systems. A consistent treatment of the long-range forces in simulations with periodic boundary conditions requires the use of a periodic form of the Coulomb potential. In this article, the FMM is extended to periodic systems, and combined with RESPA, yielding a new algorithm that is successfully applied to the simulation of large biomolecules in solution. If the interactions at different stages are separated smoothly, good energy conservation is obtained even for time steps as large as 12 fs on a system of over 40 000 atoms, and a CPU speedup of more than a factor of 20 is achieved compared to the standard Verlet integrator with Ewald sum for the Coulombic interaction. As compared with the recently developed particle-mesh Ewald (PME) method, the periodic r-RESPA/FMM has a break-even point at about 20 000 atoms; for larger systems, r-RESPA/FMM is expected to be more efficient. © *1997 American Institute of Physics.* [S0021-9606(97)51923-4]

I. INTRODUCTION

Performing large scale simulations of macromolecules in solution is still a challenge. For some problems, a realistic representation of the effects of the solvent environment necessitates an atomistic model of both solute and solvent, thus the need for large numbers of atoms. Since the number of pairwise interactions among N atoms grows like N^2, the direct computation of all of them is not practical, even with current supercomputers, for more than a few thousand atoms. This problem is often circumvented in biomolecular simulations by truncating the forces beyond a ''cutoff'' distance.[1,2] With clever pair-list generating algorithms, the computational cost scales as $O(N)$. However, the truncation of forces changes the underlying physical system. Recently, it has been shown that the truncation of long-range electrostatic interactions introduces unrealistic physical effects.[3–12] To avoid the truncation of long-range interactions, several groups have experimented with approximate schemes, of which the most widely used is the fast multipole method (FMM) of Greengard and Rokhlin[13–17] and its variants.[18–22] This algorithm, described in more detail in Sec. II, decreases the computational burden to $O(N)$ by cleverly exploiting a hierarchy of clusters and using multipolar expansions to approximate the potential produced by these clusters. Recently, another promising algorithm, particle-mesh Ewald (PME),[21,23–25] has been described in the literature.

Another common approach to speed up the simulation of solvated biomolecules is the use of holonomic constraints[26,27] to effectively freeze the rapidly varying degrees of freedom, thus allowing for larger time steps. Unfortunately, the use of a Verlet integrator plus these constraints limits the time step that can be safely taken to between 0.5 and 2 fs, depending on the complexity of the system. For a solvated protein of about 1000 atoms, one is typically limited to a time step of 1 fs. To go beyond these limitations, algorithms more sophisticated than the usual Verlet integrator are necessary, and several multiple time step methods have been proposed.[2,28–37] The reversible reference system propagator algorithm (r-RESPA) developed by Berne *et al.*[33] is a general technique which yields a family of multiple time step integration algorithms.

In this article, we devise a new algorithm which combines the periodic FMM for computing the long-range electrostatic forces with the RESPA multiple time scale integrator, and apply it to simulations of solvated proteins with periodic boundary conditions. The periodic FMM is necessary to handle, in a consistent manner, the effects of periodic boundary conditions, which are typically used in simulations that include explicit solvent. The combined r-RESPA/FMM algorithm for free boundary conditions was described in an article by Zhou and Berne;[34] this article extends their treatment to the periodic case. We also report comparisons with published performance figures for the PME and r-RESPA/PME methods.

The structure of the article is as follows. In Sec. II, the FMM is described, based largely on Greengard's

a)Electronic mail: ronlevy@lutece.rutgers.edu

 0021-9606/97/106(23)/9835/15/$10.00

dissertation.[13] Since most simulations use periodic boundary conditions, the consistent treatment of the electrostatic interactions in a periodic system using the FMM is discussed in Sec. III. Section IV provides a description of the *r*-RESPA technique, with the emphasis on the separation of forces based on different distance scales. In Sec. V, the separation of forces in the FMM used in the applications of the *r*-RESPA method is discussed. This separation in FMM can be done in a number of ways, including a sharp separation based on the boundaries between boxes, as done by Zhou and Berne[34] for isolated protein systems, or a smooth separation using a switching function. Switching functions have been widely used in nonbonded force separations in *r*-RESPA.[33,34] We have implemented both methods and compare their efficiency. Section VI presents an analysis of simulations on several different systems, ranging from a pair of ions in aqueous solution to a solvated macromolecule, with emphasis on the energy conservation as a function of the simulation parameters. From this analysis, it appears that the presence of solvent has a dramatic effect on the conservation of energy: For a given level of energy conservation, a more accurate simulation is needed when the solvent is included. It also emerges that using a smooth switching function sometimes allows the use of a much larger time step than when the sharp separation is implemented. In Sec. VII, we report the results of timing tests, as well as a comparison with the PME method. Finally, we offer some conclusions and discuss the implications of our findings for biomolecular simulations.

In Appendix A, we present some technical details concerning the extension of the FMM to periodic boundary conditions. Appendix B provides details of our implementation inside the molecular mechanics package IMPACT.[38]

II. THE FAST MULTIPOLE METHOD

The fast multipole method (FMM)[13,14] has already been described in several articles, at varying levels of detail. An overview of the method is given below.

The FMM starts from the observation that the electrostatic potential produced by a collection of charges can be approximated by a multipolar sum

$$\phi(\mathbf{x}) \approx \sum_{l=0}^{p} \sum_{m=-l}^{l} (-1)^l \frac{\mathbf{m}_l^m}{A_l^m \|\mathbf{x}-\mathbf{x}_0\|^{l+1}} Y_l^m(\widehat{\mathbf{x}-\mathbf{x}_0}), \quad (1)$$

where the multipolar coefficients $\mathbf{m}_l^m$ depend only on the sources and not on the observation point, $\mathbf{x}$ is the observation point, and $\mathbf{x}_0$ is the geometric center of the charges. The caret indicates the unit vector. Our definition of the multipolar coefficients, which is chosen to simplify the equations that follow,[17] differs from the usual one by the factor $(-1)^l/A_l^m$, where A_l^m is given by the formula[13]

$$A_l^m = \frac{(-1)^l}{\sqrt{(l-m)!(l+m)!}}. \quad (2)$$

In contrast to White and Head Gordon,[17] however, we use the same nonstandard normalization for the spherical harmonics as Greengard:[13]

$$Y_l^m(\theta,\phi) = \sqrt{\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi} \quad (3)$$

for $m \geqslant 0$ and

$$Y_l^{-m} = Y_l^{m*}. \quad (4)$$

Although the use of this approximation was proposed almost twenty years ago,[39,40] Appel[41] was the first to recognize that one can compute these coefficients efficiently using a hierarchical algorithm, where the particles are collected into ''clusters'' at different levels. The first level, $L=0$, encompasses all the particles in the system; each successive level $L+1$ is obtained by dividing the clusters at level L into octants. After this construction has proceeded up to some specified level L, where the clusters typically contain only a few particles, the multipole coefficients are computed for each cluster at that level, using the formula

$$\mathbf{m}_l^m = (-1)^l A_l^m \sum_\alpha q_\alpha \|\mathbf{x}_\alpha - \mathbf{x}_0\|^l Y_l^{-m}(\widehat{\mathbf{x}_\alpha - \mathbf{x}_0}), \quad (5)$$

where the sum is taken over all charged particles in the cluster, and $\mathbf{x}_0$ is the geometric center of the cluster. The algorithm continues then by evaluating the coefficients of a cluster at level L in terms of those of its octants at level $L+1$. For this purpose, a ''multipole translation'' formula is used

$$\mathbf{m}_l'^m = \sum_{j=0}^{l} \sum_{n=-j}^{j} J(m-n,n) \times \mathbf{m}_{l-j}^{m-n} (-1)^j \|\mathbf{x}-\mathbf{x}'\|^j A_j^n Y_j^{-n}(\widehat{\mathbf{x}-\mathbf{x}'}) \quad (6)$$

which gives the coefficients of the multipole expansion around $\mathbf{x}'$ in terms of the coefficients of the multipole expansion around $\mathbf{x}$, where $J(m,n)$ is given by

$$J(m,n) = \begin{cases} (-1)^{\min\{|m|,|n|\}} & \text{if } m \cdot n < 0 \\ 1 & \text{otherwise.} \end{cases} \quad (7)$$

Barnes and Hut[42] improved on Appel's method by introducing a faster algorithm for ''loading'' the particles onto the clusters at level L, that is, deciding to which of the 8^L clusters each particle belongs. This algorithm was shown by Salmon[43] to be asymptotically $O(N \log N)$; he also described a parallel version. Hernquist,[44] Makino,[45,46] and Barnes[47] later modified this algorithm to improve the performance on vector supercomputers. Saito[20] and Shimada *et al.*[21] described algorithms that used similar ideas. Greengard and Rokhlin[13,14] went a step further with the observation that the multipolar expansions of the potentials produced by several clusters could be lumped together into a local expansion of the form

$$\phi(\mathbf{x}) = \sum_{l=0}^{p} \sum_{m=-l}^{l} \mathbf{l}_l^m A_l^m \|\mathbf{x}-\mathbf{x}'\|^l Y_l^m(\widehat{\mathbf{x}-\mathbf{x}'}), \quad (8)$$

where the coefficients $\mathbf{l}_l^m$ again are independent of the observation point, $\mathbf{x}$, but depend on the center of expansion, $\mathbf{x}'$, and showed that their algorithm was asymptotically $O(N)$, albeit with a large constant factor. The corresponding field is given by the formula

$$\mathbf{E}(\mathbf{x})=\sum_{l=0}^{p}\sum_{m=-l}^{l}\mathbf{l}_l^m A_l^m \|\mathbf{x}-\mathbf{x}'\|^{l-1}[-lY_l^m(\widehat{\mathbf{x}-\mathbf{x}'}) + i(\widehat{\mathbf{x}-\mathbf{x}'})\times\mathbf{L}Y_l^m], \tag{9}$$

where the angular momentum operator $\mathbf{L}$ acts as follows:

$$i\mathbf{L}Y_l^m=\left(i\,\frac{\sqrt{l(l+1)-m(m+1)}}{2}\,Y_l^{m+1} + i\,\frac{\sqrt{l(l+1)-m(m-1)}}{2}\,Y_l^{m-1},\ \frac{\sqrt{l(l+1)-m(m+1)}}{2}\,Y_l^{m+1} - \frac{\sqrt{l(l+1)-m(m-1)}}{2}\,Y_l^{m-1},\quad imY_l^m\right). \tag{10}$$

The coefficients of these local expansions are computed in a hierarchical manner, described later, in which one of the steps involves a ''multipole to local'' translation formula analogous to Eq. (6)

$$\mathbf{l}_l'^m=\sum_{j=0}^{p}\sum_{n=-j}^{j}J'(n,m)\mathbf{m}_j^n\,\frac{1}{A_{l+j}^{n-m}\|\mathbf{x}-\mathbf{x}'\|^{l+j+1}}\times Y_{l+j}^{n-m}(\widehat{\mathbf{x}-\mathbf{x}'}), \tag{11}$$

where

$$J'(m,n)=\begin{cases}(-1)^{\min\{|m|,|n|\}} & \text{if } m\cdot n>0\\ 1 & \text{otherwise.}\end{cases} \tag{12}$$

The other main step in this calculation is a ''local to local'' translation formula,

$$\mathbf{l}_l'^m=\sum_{j=l}^{p}\sum_{n=-j}^{j}J''(n,n-m)\mathbf{l}_j^n(-1)^{j-l}\|\mathbf{x}-\mathbf{x}'\|^{j-l}\times A_{j-l}^{n-m}Y_{j-l}^{n-m}(\widehat{\mathbf{x}-\mathbf{x}'}), \tag{13}$$

where J'' is given by

$$J''(m,n)=\begin{cases}(-1)^{|n|} & \text{if } m\cdot n<0\\ (-1)^{|m-n|} & \text{if } m\cdot n>0 \quad \text{and} \quad |m|<|n|,\\ 1 & \text{otherwise,}\end{cases} \tag{14}$$

Board *et al.*[15] described one of the first implementations in the context of molecular dynamics. They also described a parallel version. White and Head–Gordon[17] studied the accuracy and performance of the FMM as a function of the number of multipoles retained in the sum, p, and the depth L of the tree. Ding *et al.*[18] described a slightly different algorithm where Cartesian multipoles are used instead of the expansions in spherical harmonics. Their method was limited to only a few moments due to the complexity of the Cartesian representation. Fenley *et al.*[48] implemented an adaptive version of the FMM for their calculations of the total electrostatic energy of strands of DNA. None of these implementations, however, treated the case of periodic boundary conditions, although Schmidt and Lee[49] had described an Ewald-like method to extend the FMM to this case.

The FMM involves the following stages.

(FMM.0) Create a tree structure to describe the clusters from levels $l=0$ to L. This must be done only once.

(FMM.1) Load the particles onto the clusters (nodes of the tree) at level L.

(FMM.2) Compute the multipolar coefficients of all clusters at level L, Eq. (5), and then use the recursive procedure, Eq. (6), to compute the coefficients at level l given those at level $l+1$, for $l<L$. At the end of this step, the multipolar coefficients of an expansion around the center of each cluster would have been computed.

(FMM.3) Using Greengard's recursive algorithm, compute the coefficients of the local expansion, around the center of each cluster c at level l ($l\leqslant L$). This involves two steps: (i) shifting the local expansion coefficients of c's parent at level $l-1$ to c at level l, using Eq. (13); (ii) adding the local expansions from multipoles of clusters that are children of c's parent's first and second neighbors, but are neither first nor second neighbors of c itself, using Eq. (11). After this step, the multipolar expansions of every cluster that is not a first nor a second neighbor of c (at the same level u) are lumped together. The first and second neighbors are excluded because they are calculated directly.

(FMM.4) For each particle in a cluster c at the lowest level L, compute the contribution to the energy and force from the local expansion of distant multipoles using Eq. (9); add to this the contribution from the nearby particles that have not contributed to the above local expansion coefficients, that is, particles in either c, or first or second neighbors of c.

In step (FMM.4), all the direct interactions between particles in c and those in c and its first and second neighbors are computed; this is just like the direct approach, except that it is applied at level L. Increasing L decreases the size of the clusters at that level, and therefore the time spent in step (FMM.4), however, it increases the total number of clusters, and thus the time spent in steps (FMM.2) and (FMM.3) [step (FMM.1) is usually extremely fast]. Of these two, steps (FMM.3) is by far the most expensive. To obtain an efficient algorithm, one must therefore find a balance between steps (FMM.3) and (FMM.4). One way to do this (see Sec. V) is to use a deep tree (large L, small clusters) but to do step (FMM.3) less often. This leads us naturally to consider multiple time step methods (see Sec. IV).

III. EXTENSION TO PERIODIC BOUNDARY CONDITIONS

The first implementation of the FMM for systems with periodic boundary conditions, to our knowledge, is that of Schmidt and Lee,[49] although Greengard's dissertation[13] contains a brief description of the main ideas. Recently, Esselink[50] compared the algorithmic complexity of the periodic FMM and the Ewald summation method. This section enlarges on the method of Schmidt and Lee and presents a

derivation of formulas which reduce to regular Ewald in the proper limits. We were unable to reduce the corresponding formulas in Ref. 49 to the correct limiting forms.

In the description below, we assume that the simulation box is cubic and its linear dimension is b. To introduce periodic boundary conditions, it is convenient to think of the simulation system as consisting of an (infinite) lattice of exact replicas of the unit cell. We focus on one particular cell among them and call it the ''central'' box; the others will be called the ''proper copies.'' The potential produced by all the proper copies at a point $\mathbf{x}$ inside the central box can be represented, neglecting for the moment issues of convergence, by the infinite sum

$$\phi(\mathbf{x})=\sum_{l=0}^{p}\sum_{m=-l}^{l}\sum_{\mathbf{n}\neq\mathbf{0}}(-1)^l\mathbf{m}_l^m\frac{Y_l^m(\widehat{\mathbf{x}-b\mathbf{n}})}{A_l^m\|\mathbf{x}-b\mathbf{n}\|^{l+1}}. \tag{15}$$

Schmidt and Lee regarded these proper copies as ''virtual'' clusters and applied the same FMM algorithm to them as to the central box. More precisely, the proper copies can be subdivided into first and second neighbors of the central box and all the others. The third neighbors and more distant copies, that is, those for which at least one component of $\mathbf{n}$ is larger than 2, will be called ''distant copies''. The local field from these distant copies at point $\mathbf{x}$ in the central box can then be expressed by

$$\phi(\mathbf{x})=\sum_{l=0}^{p}\sum_{m=-l}^{l}\mathbf{l}_m^l A_l^m\|\mathbf{x}\|^l Y_l^m(\hat{\mathbf{x}}), \tag{16}$$

where the expansion coefficients are given by the infinite sum

$$\mathbf{l}_l^m=\sum_{j=1}^{p}\sum_{n=-j}^{j}J'(n,m)\mathbf{m}_j^n \times\left(\frac{1}{b^{l+j+1}A_{l+j}^{n-m}}\sum_{\mathbf{n}\neq\mathbf{0}}{}'\frac{Y_{l+j}^{n-m}(\hat{\mathbf{n}})}{\|\mathbf{n}\|^{l+j+1}}\right). \tag{17}$$

(The $j=0$ term does not enter the summation since the box is neutral.) The first and second neighbors are treated in the usual way as if they were part of the simulation volume, except that forces on particles inside them are never computed. In what follows, it is more convenient to consider the sum in Eq. (17) as extending over all nonzero $\mathbf{n}$; to get back the constrained sum, we only need to subtract the explicit sum over first and second neighbors. This approach therefore differs from the usual FMM in that the simulation box starts with a nonzero local expansion, given by the sum in Eq. (17), and all clusters have the same number of first and second neighbors.

The only dependence on the instantaneous configuration of the charges in Eq. (17) lies in the multipole moments of the central box, $\mathbf{m}_l^m$. Thus the infinite sums in parenthesis need be computed only once, provided the shape of the central simulation box does not change. Following Schmidt and Lee, this computation is carried out using an extension of the Ewald summation method, which is described next; the derivation parallels that given by de Leeuw *et al.*[51] For fixed l and m, we need to compute

$$S_l^m\equiv\sum_{\mathbf{n}\neq\mathbf{0}}\frac{Y_l^m(\hat{\mathbf{n}})}{\|\mathbf{n}\|^{l+1}}, \tag{18}$$

which is conditionally convergent for $1\leqslant l\leqslant 2$ and absolutely convergent for $l>2$. To obtain a convergent sum that can be manipulated, we replace Eq. (18) by

$$S_l^m=\lim_{s\to 0^+}\sum_{\mathbf{n}\neq\mathbf{0}}\frac{\|\mathbf{n}\|^l Y_l^m(\hat{\mathbf{n}})}{\|\mathbf{n}\|^{2l+1}}e^{-s\|\mathbf{n}\|^2}. \tag{19}$$

With this regularization, the sum for odd l vanishes identically because of reflection symmetry. The sum over even l converges absolutely but slowly. To accelerate the convergence, we proceed, as for the Ewald summation, and use the identity

$$\frac{1}{x^{2r}}=\frac{1}{\Gamma(r)}\int_0^\infty t^{r-1}e^{-tx^2}dt, \tag{20}$$

obtained from the definition of the Γ function by a simple change of variables. Inserting this representation into Eq. (19) gives,

$$S_l^m=\lim_{s\to 0^+}\frac{1}{\Gamma\left(l+\frac{1}{2}\right)}\sum_{\mathbf{n}\neq\mathbf{0}}\|\mathbf{n}\|^l Y_l^m(\hat{\mathbf{n}}) \times\int_0^\infty t^{l+1/2-1}e^{-(s+t)\|\mathbf{n}\|^2}dt. \tag{21}$$

For $s>0$, the sum converges absolutely and the summation and integration can be interchanged, so that

$$S_l^m=\lim_{s\to 0^+}\frac{1}{\Gamma\left(l+\frac{1}{2}\right)}\int_0^\infty dt t^{l+1/2-1} \times\sum_{\mathbf{n}\neq\mathbf{0}}\|\mathbf{n}\|^l Y_l^m(\hat{\mathbf{n}})e^{-(s+t)\|\mathbf{n}\|^2}. \tag{22}$$

Separation of the integral at a midpoint, which de Leeuw *et al.* call α_2, requires evaluation of the two sums

$$A=\int_0^{\alpha^2}dt t^{l+1/2-1}\sum_{\mathbf{n}\neq\mathbf{0}}\|\mathbf{n}\|^l Y_l^m(\hat{\mathbf{n}})e^{-(s+t)\|\mathbf{n}\|^2}. \tag{23}$$

and

$$B=\int_{\alpha^2}^\infty dt t^{l+1/2-1}\sum_{\mathbf{n}\neq\mathbf{0}}\|\mathbf{n}\|^l Y_l^m(\hat{\mathbf{n}})e^{-(s+t)\|\mathbf{n}\|^2}. \tag{24}$$

In the second sum, the convergence is absolute for all $s>-\alpha^2$, so that we can take the limit $s\to 0^+$ with confidence. We can then rewrite the sum B as

$$B=\sum_{\mathbf{n}\neq\mathbf{0}}\frac{Y_l^m(\hat{\mathbf{n}})}{\|\mathbf{n}\|^{l+1}}\int_{\alpha^2\|\mathbf{n}\|^2}^\infty t^{l+1/2-1}e^{-t}dt. \tag{25}$$

We are thus led to consider the integrals

$$I_r(x)=\int_x^\infty t^{r-1}e^{-t}dt \tag{26}$$

which can be shown to satisfy the recurrence relation

$$I_r(x)=x^{r-1}e^{-x}+(r-1)I_{r-1}(x), \tag{27}$$

valid for any r. Since the values for half integral $r(r=l+1/2)$ are needed, the recurrence relations must be supplemented with the initial value

$$I_{1/2}(x)=\sqrt{\pi}[1-\mathrm{erf}(\sqrt{x})], \tag{28}$$

where $\mathrm{erf}(x)$ is the usual error function.

The first sum is rather more complicated and the details are given in Appendix A. After some algebra, one arrives at the final result for the sum over all proper copies:

$$\begin{aligned} S_l^m &= \sum_{\mathbf{n}\neq\mathbf{0}} \frac{Y_l^m(\hat{\mathbf{n}})}{\|\mathbf{n}\|^{l+1}} \\ &= \sum_{\mathbf{n}\neq\mathbf{0}} \frac{Y_l^m(\hat{\mathbf{n}})}{\|\mathbf{n}\|^{l+1}} I_{l+1/2}(\alpha^2\|\mathbf{n}\|^2) \\ &\quad + \sum_{\mathbf{k}\neq\mathbf{0}} \pi^{l-1/2} i^l Y_l^m(\hat{\mathbf{k}})\|\mathbf{k}\|^{l-2} e^{-\pi^2\|\mathbf{k}\|^2/\alpha^2}. \end{aligned} \tag{29}$$

As discussed above, the explicit sum over first and second neighbors must be subtracted from this result.

The results embodied in Eq. (29) resembles the Ewald summation formula[2,51] but differs from the latter in a key aspect: whereas the Ewald summation formula gives the Wigner potential at each point in the unit cell, Eq. (29) gives just the nontrivial part of the coefficients of the translation matrix that converts the unit cell's multipole moments into a local expansion around its center of the potential produced by all the proper copies (that is, not including the central cell). The full translation matrix is given by Eq. (17).

IV. REVERSIBLE RESPA (*r*-RESPA)

The reversible reference system propagator algorithms (r-RESPA)[33] form a family of multiple time step algorithms derived from a Trotter factorization of the Liouville propagator,

$$U(t)=e^{-itL}, \tag{30}$$

$$iL=\sum_j \left(\frac{1}{m_j}\mathbf{p}_j\frac{\partial}{\partial\mathbf{q}_j}+\mathbf{F}_j(\mathbf{q})\frac{\partial}{\partial\mathbf{p}_j}\right). \tag{31}$$

The basic idea is to decompose the Liouville operator into the sum $L=L_1+L_2$ and then use the Trotter formula to approximate the full propagator $U(\Delta t)$ for a finite but small time step Δt as

$$U(\Delta t)\approx U_2\left(\frac{\Delta t}{2}\right)U_1(\Delta t)U_2\left(\frac{\Delta t}{2}\right), \tag{32}$$

where U_j is the propagator associated with L_j. This expansion is accurate to order $O(\Delta t^3)$, but judicious choice of the decomposition can yield an algorithm for which a ''long'' Δt can be used; for a detailed discussion, the reader is referred to the original publication.[33] This decomposition guarantees that the integration is time reversible and confers a long-time stability on the integrator.[33,35–37] Different decompositions yield different integration algorithms;[34,52,53] for our purposes, two decompositions are of interest. The first decomposition defines

$$L=L_{\mathbf{q}}+L_{\mathbf{p}}, \tag{33}$$

$$iL_{\mathbf{q}}=\sum_j \frac{1}{m_j}\mathbf{p}_j\frac{\partial}{\partial\mathbf{q}_j}, \tag{34}$$

$$iL_{\mathbf{p}}=\sum_j \mathbf{F}_j(\mathbf{q})\frac{\partial}{\partial\mathbf{p}_j}. \tag{35}$$

In this case, since each Liouville operator ''propagates'' only one half of the conjugated variables, their finite-time propagators are easily computed:

$$U_{\mathbf{q}}(\Delta t)g(\mathbf{q},\mathbf{p})=e^{-i\Delta tL_{\mathbf{q}}}g(\mathbf{q},\mathbf{p})=g\left(\mathbf{q}-\frac{\Delta t}{m}\mathbf{p},\mathbf{p}\right) \tag{36}$$

$$U_{\mathbf{p}}(\Delta t)g(\mathbf{q},\mathbf{p})=e^{-i\Delta tL_{\mathbf{p}}}g(\mathbf{q},\mathbf{p})=g[\mathbf{q},\mathbf{p}-\Delta t\mathbf{F}(\mathbf{q})], \tag{37}$$

where $g(\mathbf{q},\mathbf{p})$ is an arbitrary (smooth) function defined on the phase space. From these identities and the approximation

$$U(\Delta t)\approx U_{\mathbf{p}}\left(\frac{\Delta t}{2}\right)U_{\mathbf{q}}(\Delta t)U_{\mathbf{p}}\left(\frac{\Delta t}{2}\right) \tag{38}$$

one immediately obtains the Verlet[1] integration algorithm, as shown in Ref. 33.

The second decomposition of interest separates the forces into ''fast'' and ''slow'' components, $\mathbf{F}_f$ and $\mathbf{F}_s$, and the corresponding Liouville operators are defined by

$$iL_f=\sum_j\sum_j\left(\frac{1}{m_j}\mathbf{p}_j\frac{\partial}{\partial\mathbf{q}_j}+\mathbf{F}_{f,j}(\mathbf{q})\frac{\partial}{\partial\mathbf{p}_j}\right), \tag{39}$$

$$iL_s=\sum_j \mathbf{F}_{s,j}(\mathbf{q})\frac{\partial}{\partial\mathbf{p}_j}. \tag{40}$$

The fast propagator (reference propagator) can be further decomposed as,

$$U_f(\Delta t)=\left[U_f\left(\frac{\Delta t}{n}\right)\right]^n \tag{41}$$

followed by a Verlet-type decomposition for the inner propagator. All the algorithms described in this article follow this pattern, with the innermost propagator handled by the simple Verlet integrator. The advantage and disadvantage of different factorization are discussed in a recent article.[54]

The fast and slow components are often identified by separating a distance-dependent force into a short- and a long- (or medium-) range pieces; this is conveniently done with a ''switching'' function, $s(r)$, such that $s(r)=0$ for $r<r_l$ and $s(r)=1$ for $r>r_u$, where $r_l<r_u$ are two cutoff distances chosen according to the problem at hand.[33,34,52] With such a switching function, one takes

$$\mathbf{F}_f(r)=[1-s(r)]\mathbf{F}(r) \tag{42}$$

$$\mathbf{F}_s(r)=s(r)\mathbf{F}(r). \tag{43}$$

The decomposition (39) can be applied recursively: the fast forces $\mathbf{F}_f$ can be further decomposed into modes with different time scales, and the same Trotter expansion used on this decomposition.

In macromolecular simulations, one typically subjects the system to holonomic constraints to keep the bond lengths fixed during the simulation. Moreover, many water models have been parameterized assuming rigid geometries. These constraints allow for larger time steps since the fast bond vibrations are frozen. The use of *r*-RESPA integrators, on the other hand, allows the use of a short time step for the rapidly varying bonding forces, and a large time step for the nonbonding forces, the most-expensive part of calculations, less frequently. However, since many force fields have been parameterized assuming rigid bonds, it is useful to allow for these constraints. In all the integration algorithms described in this article, with the exception of simple Verlet (where all the forces are computed at every step), we treat the bonding forces as the ''fastest'' forces, that is, they are the ones that get updated in the innermost propagator loop. Since it is only inside that loop that the coordinates are updated, to satisfy the holonomic constraints, we apply coordinate corrections using, for example, the SHAKE[26] algorithm, within this innermost loop. Since the time step is small, the updates are small and SHAKE converges rapidly. If one wants to apply also velocity corrections using RATTLE[27] they must, on the other hand, be applied after every update of the velocities, which occur at all levels. It should be noted that when SHAKE and RATTLE are used, the resulting RESPA integrator is no longer reversible. Since the CPU time spent doing these updates scales linearly with the number of atoms, the overhead involved is negligible for large systems where the nonbonded force calculation consumes well over 90% of the time.

In Sec. V, the actual force decompositions used in our code are described.

V. COMBINING THE FMM WITH *r*-RESPA

The bottlenecks in the FMM are steps (FMM.3), where the local expansion coefficients are generated, and (FMM.4), where the interactions with particles that lie in a cluster or one of its first or second neighbors are computed. Using a deeper tree (larger L) shifts the burden from step (FMM.4) to (FMM.3). The interactions between particles that are far from each other are computed through the local expansions in this step, and thus, this step involves the slowly varying long-range force which can be treated in the outer loop.

The bonding forces will always comprise the innermost loop. In our code, we have implemented several different decompositions of the nonbonding forces:

(i) *r*-RESPA2: all nonbonded forces are computed at once, that is, they are not decomposed; since the bonding forces are computed separately this is a two-stage *r*-RESPA.

(ii) *r*-RESPA3: the nonbonded forces are divided into two ranges: (a) short range: those arising from the direct pair interactions within the cluster itself and the first neighbor clusters; and (b) long range: the rest, that is, those computed as direct interactions with particles in the second shell of clusters plus those computed via the local expansions;

(iii) *r*-RESPA4: we divide the nonbonded interactions into three ranges: (a) short range: those arising from the cluster itself and first neighbors; (b) medium range: those arising from the second neighbors; and (c) long range: the contribution from the local expansions of distant multipoles;

For both *r*-RESPA3 and *r*-RESPA4, we have also implemented variants (*r*-RESPA3-sm and *r*-RESPA4-sm, respectively) which use a smooth, spherically symmetric switching function to separate the direct interactions into short and long range. The choice of decomposition is under complete control of the user through special keywords added to the IMPACT command language (see Appendix B).

As shown above, the direct calculation in the FMM is broken into two stages in two ways: (a) as in Zhou and Berne,[34] a sharp cutoff based on the cluster decomposition for electrostatic forces in isolated proteins; and (b) using a smooth, spherically symmetric switching function to define the short- and medium-range components (the long-range component is always identified with that computed from the local expansions), as was done previously.[33,34] The reason for this dual implementation was to test whether the sharpness of the Zhou–Berne cutoff introduces problems in situations other than those treated by them (proteins in vacuum).

The particular form of the switching function used in *r*-RESPA3-sm and *r*-RESPA4-sm is not very important, as long as it and its first derivative are continuous. We chose the same form as Zhou and Berne had chosen for their Lennard-Jones interactions, which were computed separately from the electrostatic ones, in contrast to our code. The switching function we use is defined by

$$s(r)=\begin{cases} 0 & \text{if } r<r_l, \\ S\left(\dfrac{r-r_l}{r_u-r_l}\right) & \text{if } r_l\leqslant r\leqslant r_u; \\ 1 & \text{if } r>r_u \end{cases} \tag{44}$$

where S is the polynomial

$$S(x)=x^2(3-2x) \tag{45}$$

which has the property that its first derivative vanishes at both $x=0$ and $x=1$. It should be noted that the sharp, cell-based cutoff used in *r*-RESPA3 and *r*-RESPA4 is more natural, and results in a faster method, from the point of view of the FMM. A spherically symmetric switching function is, on the other hand, more natural to separate the forces at different length scales. It is possible, however, to consider also a smooth switching function with cubic symmetry.

Note that for the two-stage method (*r*-RESPA2) the original FMM is used, since in this case we do not separate the nonbonding forces into short- and long-range components.

To combine the FMM with any of the *r*-RESPA integrators, a slight modification of the algorithm described in Sec. II is needed. Steps (FMM.0) through (FMM.3) remain essen-

tially the same, except that (FMM.2) and (FMM.3) are skipped in all but the outermost loop, where the forces that need to be computed contain the contribution from the local expansions. Step (FMM.4) is replaced by the following.

(FMM.4′) For each particle in a cluster c at the lowest level L, compute the contribution to the energy and force from: (a) the local expansion from distant multipoles around the center of the cluster c using Eq. (9), if we are in the outermost loop; (b) the medium-range interactions (from second neighbors), if this stage calls for it; or (c) the short-range interactions (from cluster itself and its first neighbors). Notice that the decomposition into short- and medium-range interactions is different, as discussed above, when a sharp cutoff or a switching function are used.

VI. ENERGY CONSERVATION

Energy conservation during the simulation is a common requirement for an integration algorithm.[33,34,38,55,56] The deviations from exact energy conservation arise from many sources: (a) the finite time step used in the numerical integration, which renders the method only approximately Hamiltonian; (b) the finite precision in the numerical evaluation of the forces; (c) the intrinsic deficiencies of the integration algorithm, such as not being symplectic. In this article, we are concerned mostly with (a) and (b). To study the effects of time step and errors in the forces on the energy conservation, we have run simulations of several different systems, ranging from pure SPC water to ions in water, to proteins in water. In addition, we have studied the effect of using periodic boundary conditions as opposed to free boundary conditions on the energy conservation.

Two energy conservation parameters are commonly used to describe the stability of a constant-energy molecular dynamics (MD) simulation.[33,34,38,55] One is the total energy fluctuation ΔE defined by

$$\Delta E \equiv \frac{1}{N_T} \sum_{i=1}^{N_T} \left| \frac{E_0 - E_i}{E_0} \right|, \tag{46}$$

where E_i is the total energy at step i, E_0 is the initial energy, and N_T is the total number of time steps. This quantity has been shown to be a reasonable measure of accuracy in previous simulations,[53,55] and a value of $\Delta E \leqslant 0.003$, i.e., $\log \Delta E < -2.5$, gives an acceptable numerical accuracy. Another common measure of accuracy is the ratio of the rms deviation of the total energy to the rms deviation of the kinetic energy,[38,56]

$$R \equiv \frac{\Delta E_{\mathrm{rms}}}{\Delta KE_{\mathrm{rms}}}. \tag{47}$$

A value of $R \leqslant 0.05$, has been used as an alternate criterion for stability in MD simulations.[53,55] In this article, we use $\log \Delta E$ as a measure for the energy conservation.

As a test of accuracy, we have examined the error in the force between two oppositely charged ions in a 32 Å box, as a function of their relative separation. The difference between the results of a very accurate Ewald calculation ($\eta = 11.0$, $r_c = 16.0$ Å, $k_{\max} = 30$) and a FMM calculation with $L = 3$, $p = 20$ was always less than 3×10^{-6}.

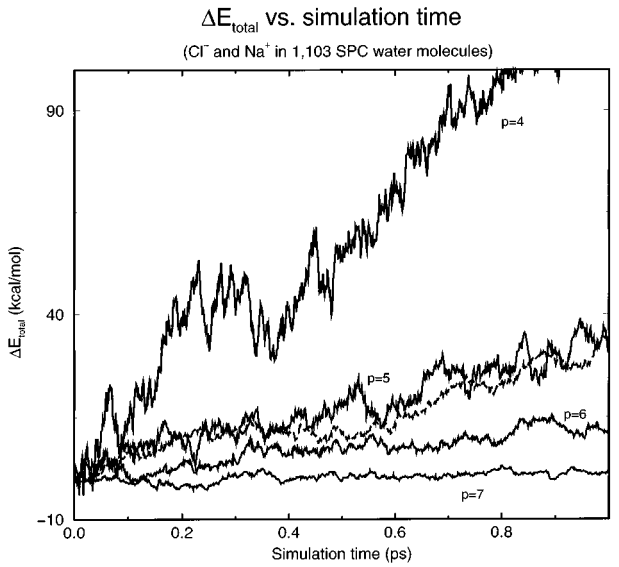

FIG. 1. Energy conservation for different force algorithms, using the Verlet integrator. The system studied contained one Cl^- and one Na^+ ion, plus 1103 SPC water molecules. The dashed line corresponds to a simulation using the Ewald summation method ($r_c = 9$ Å, $\eta = 5.5$, $k_{\max} = 5$); the others are for simulations using the FMM with different number of multipoles (p).

A. Ions in SPC water

We ran several 1 ps simulations using the simple Verlet integration algorithm, but using different force algorithms, on a system consisting of 1103 SPC water molecules plus one Cl^- and one Na^+ ion. The simulation box was a cube of side 32 Å. The SPC molecules were kept rigid using SHAKE/RATTLE[26,27] as described in Sec. V. For the Ewald method, the convergence parameter α was chosen equal to η / b_x, where b_x is the linear dimension of the (cubic) box and η was taken to be 5.5; the real-space cutoff distance was 9 Å. For the FMM, the depth was fixed at $L = 3$ and the number of multipoles was varied from a minimum of $p = 4$ to a maximum of $p = 7$. Figure 1 illustrates the drift with respect to the initial total energy as a function of the simulation time. It indicates that for this ion/water system, a multipole level of at least $p = 6$ should be used to generate a stable MD simulation, however, it is found that for isolated protein systems, using $p = 4$ is enough. This difference in required p for proteins in vacuum and NaCl solution shows that the level of multipoles needed to achieve a specified accuracy depends on the specific system. We will discuss this in Sec. VII.

Table I presents the results from several runs using the r-RESPA integrator. The entries in the first column give the combination of force separation stages. The notation we use is analogous to that used by Zhou and Berne;[34] each of the four components corresponds to a particular stage—except that a ''1'' means that the corresponding stage is absent, i.e.,

TABLE I. Energy conservation for different r-RESPA integrators. In all cases, the FMM with $L=3$, $p=7$ was employed. The system studied contained one Cl^- and one Na^+ ion, plus 1103 SPC water molecules. See the text for a discussion of the parameters used.

Method	Δt (fs)	$\log \Delta E$	CPU (10^3 s/ps)	$\Delta E_{\rm rms}$ (%)	R (%)
(2,1,1)	2	−4.11	4.9	0.009	5
(4,1,1)	4	−3.60	2.5	0.018	10
(2,2,1)	4	−1.81	2.4	0.89	180
(2,2,1)[a]	4	−4.01	3.4	0.009	5
(2,2,2)	4	−2.03	3.2	0.53	140
(2,2,2)[a]	4	−2.32	5.1	0.27	104
(8,1,1)	8	−2.92	1.3	0.036	19
(2,4,1)	8	−1.11	1.2	4.96	260
(2,4,1)[a]	8	−3.61	1.7	0.021	11
(4,2,1)	8	−1.05	1.4	5.68	260
(4,2,1)[a]	8	−4.00	2.2	0.010	5
(2,2,2)	8	−1.42	1.6	2.33	237
(2,2,2)[a]	8	−1.68	2.5	1.24	212
(3,3,2)	8	−1.47	2.2	1.94	202
(3,3,2)[a]	8	−1.62	2.2	1.4	206

[a]Using a smooth switching function.

has been coalesced with the one immediately above (to the right of) it. Thus, for example, the entry (2,2,1) means: a three stage r-RESPA with the two inner stages being repeated for two steps each—that is, it is a r-RESPA3 (or r-RESPA3-sm), according to the classification presented in Sec. V. The parameter Δt gives the time step, in femtoseconds, used for the outermost stage; those for the inner stages are obtained by dividing by the appropriate factor. For example, if the first entry says (2,2,2), and the time step for the outermost stage is 4 fs, then the time step for the innermost stage is given by

$$\Delta t_f = \frac{\Delta t}{2\times 2\times 2} = 0.5 \text{ fs}. \tag{48}$$

The third column reports $\log \Delta E$, the quantity defined in Eq. (46); the fourth column gives the number of CPU seconds used in the run. All the timings reported in this article have been normalized to a one node IBM RS6000 SP2. The fifth and sixth columns give some other common indicators of the quality of an integration algorithm:[34,38] the rms deviation of the total energy divided by, respectively, the average total energy and the rms deviation of the kinetic energy; both quantities are shown in percentages. All these simulations were run using FMM with $L=3$ and $p=7$. From these results, it appears that the optimal force decomposition is (2,4,1) with a smooth force separation, and a time step of 8 fs, since this combination yields good energy conservation ($\log \Delta E=-3.61$) at almost a third of the computational cost. It is noteworthy also that separating the second neighbor and local expansion contributions results in poor energy conservation, probably due to charge fluctuations near the boundaries between second and third neighbors.

B. Proteins in water

The effect of solvating a protein was studied on three systems: ribonuclease $H(2rn2)$, arabinose-binding protein

TABLE II. Energy conservation of protein ribonuclease H in water (8412 atoms). The data are collected from a 1 ps MD run of the system for different methods. In all cases, the FMM with $L=4$, $p=6$ was employed. See the text for a discussion of the parameters used.

Method	(n_1,n_2,n_3)	Δt (fs)	$\log \Delta E$	CPU (10^3 s/ps)
V-Ewald	(1,1,1)	1	−3.20	27.7
V-FMM	(1,1,1)	1	−3.68	35.0
R-FMM	(2,2,1)	2	−2.84	21.2
	(2,2,1)[a]	2	−4.16	32.3
	(2,2,2)	4	−2.52	19.5
	(2,2,2)[a]	4	−2.77	31.4
	(4,4,1)	8	−1.49	8.52
	(4,4,1)[a]	8	−3.70	13.0
	(4,4,1)	12	−0.89	6.80
	(4,4,1)[a]	12	−3.27	8.48

[a]Using a smooth switching function.

$(8abp)$, and lysozyme. In all cases, the protein is put in the center of a pre-equilibrated SPC water bath, which is chosen large enough to hold the protein. For the lysozyme case, a very large water box, 76 Å on each side, was used. The water molecules that overlap with the protein are deleted in all cases. After these steps, the number of remaining water molecules are: (a) 1982 for ribonuclease H; (b) 5990 for arabinose-binding protein; and (c) 14 093 for lysozyme. The solvated proteins are then minimized by using a steepest descent method, and equilibrated to about 300 °K by velocity rescaling. After full equilibration of the protein-water system has been achieved, which can take a CPU week for very large protein systems, a constant energy simulation can be run.

The results described below were obtained from 1 ps MD simulations of $2rn2$/water (8412 atoms), $8abp$/water (22 913 atoms), and lysozyme/water (44 259 atoms). Two algorithms for calculating electrostatic forces with periodic boundary conditions, Ewald sum and periodic FMM; and two integrators, Verlet and r-RESPA, are compared.

Table II lists $\log \Delta E$ and CPU timings for protein $2rn2$/water with different methods. For the Ewald method, a cutoff $r_c=15.0$ Å and $\eta=8$ are used in the real space, and $k_{\max}=10$ is used in the reciprocal space. The parameters used here should be close to the optimal values for $2rn2$/water system. The energy conservation of the Verlet/Ewald is found to be $\log \Delta E=-3.20$ in this case. The periodic FMM was combined with the r-RESPA method as discussed in Sec. V; the parameters used were $L=3$ and $p=7$. The results indicate that a smooth cutoff in RESPA force separations is necessary for protein/water systems. If a nonsmooth force separation (box separation) is used for both van der Waals and Coulombic forces in this protein water system, the $\log \Delta E$ is −2.84, −2.52, −1.49, and −0.89 for overall time steps of 2, 4, 8, and 12 fs in r-RESPA, respectively. This poor energy conservation might be due to our using an atom-based cutoff, rather than the usual molecule–or group-based cutoffs. If a smooth separation is used, much more, stable MD simulations can be generated with $\log \Delta E$ equal to −4.16, −2.77, −3.70, and −3.27 for time steps of 2, 4, 8, and 12 fs, respectively. This shows that a very large time

TABLE III. Energy conservation of arabinose-binding protein in water (23 912 atoms). The data are collected from 0.5 ps MD run of the system for different methods. In all cases, the FMM with $L=4$, $p=6$ was employed. See the text for a discussion of the parameters used.

Method	(n_1,n_2,n_3)	Δt (fs)	$\log \Delta E$	CPU (10^3 s/ps)
V-Ewald	(1,1,1)	1	−3.20	219
V-FMM	(1,1,1)	1	−3.75	104
R-FMM	(2,2,1)[a]	2	−3.91	67.9
	(2,2,2)[a]	4	−2.37	51.1
	(4,4,1)	8	−1.09	19.4
	(4,4,1)[a]	8	−3.76	24.3
	(4,4,1)	12	−0.64	13.2
	(4,4,1)[a]	12	−3.11	16.4

[a]Using a smooth switching function.

step of up to 12 fs can be used in the *r*-RESPA/FMM algorithm for the $2rn2$/water system provided the force separation is done with a smooth switching function. Compared to the Verlet/Ewald method, the *r*-RESPA/FMM with time step of 12 fs gives a factor of 3.3 in CPU time saving at comparable accuracy for this system (8412 atoms).

Table III lists analogous results for the $8abp$/water system (22 661 atoms). For the Ewald method, we varied the cutoffs between 15 and 22 Å and η between 8 and 15, and report the best results that were obtained in this range. The parameters $L=4$ and $p=7$ were used in the FMM method for $8abp$/water. Stable MD simulations are obtained by using smooth cutoffs, similar to the $2rn2$/water system. The CPU saving is a factor of 11.4, which is reasonable because the *r*-RESPA/FMM $[O(N)]$ becomes more favorable than the Verlet/Ewald method $[O(N^{3/2})]$ as the system size increases.

Lysozyme in water (44 259 atoms) is the largest system studied in this article. In Table IV, we also show the results using $L=4$, $p=7$, and different *r*-RESPA integrators. For the Ewald method, a cutoff of 18 Å and $\eta=12$ is used in the simulation. For the reciprocal space sum, we set $k_{\max}=15$. Again we observe that a smooth force separation is necessary to obtain good energy conservation for time steps of 4 fs or more. Similarly, we obtain a very stable simulation even using a time step up to 12 fs, which gives us a speedup of about 20 with respect to the Verlet/Ewald integrator.

TABLE IV. Energy conservation for different *r*-RESPA integrators. The entries marked ''Verlet'' were run with a 1 fs time step but only for 20 steps. The others were run for a full picosecond. In all cases, the time per picosecond is reported. Except when noted the FMM with $L=4$, $p=7$ was employed. The system studied contained one molecule of lysozyme with all titratable residues neutralized, plus 14 093 SPC water molecules. See the text for a discussion of the parameters used.

Method	(n_1,n_2,n_3)	Δt (fs)	$\log \Delta E$	CPU (10^3 s/ps)
V-Ewald	(1,1,1)	1	−2.18	562.2
V-FMM	(1,1,1)	1	−4.27	185.7
R-FMM	(2,2,2)	4	−2.30	41.7
	(2,2,2)[a]	4	−2.50	64.4
	(4,4,1)	8	−1.28	24.5
	(4,4,1)[a]	8	−4.00	35.4
	(4,4,1)	12	−0.84	16.8
	(4,4,1)[a]	12	−3.75	24.1

[a]Using a smooth switching function.

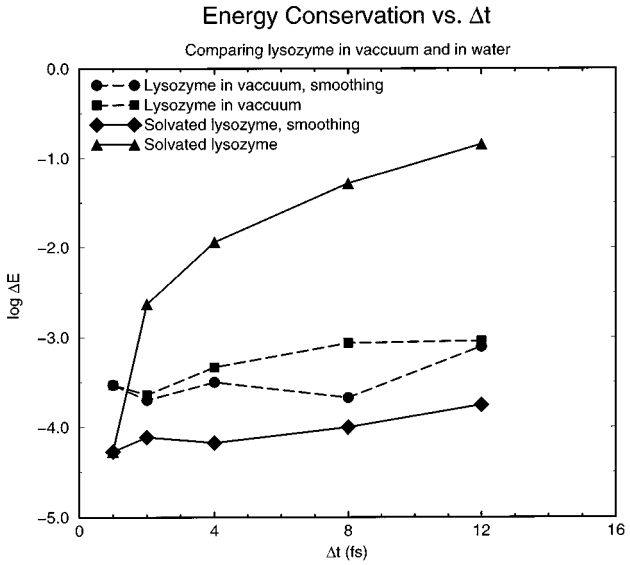

FIG. 2. Effect of the smooth cutoff on the energy conservation for protein lysozyme in vacuum (dashed lines) and in water (solid lines). It is shown that using smooth cutoff is necessary for the protein/water system, while it has less effect on the protein/vacuum system.

The above simulation results imply that for the protein/water systems it is necessary to use a smooth cutoff when performing the force separations in *r*-RESPA; however, as was reported in a previous article,[34] a very stable MD simulation can be obtained without a switching function for protein/vacuum systems. Here we use lysozyme/vacuum (1980 atoms) and lysozyme/water (44 259 atoms) systems as an example to exhibit the effect of a smooth cutoff on the energy conservation. Figure 2 shows the results for these two systems with and without smooth cutoffs. All energy conservation data are collected from 1 ps MD runs using different time steps. For the many possible combinations of (n_1,n_2,n_3) for a fixed overall time step, only the best result is reported. It is clear that the smooth cutoff greatly improves the energy conservation for the lysozyme/water system. When the overall time step increases from 1 to 12 fs, $\log \Delta E$ increases from -4.27 to -0.97 for sharp cutoffs, while it only increases from -4.27 to -3.75 for smooth cutoffs. On the other hand, the effect of the smooth cutoff is much smaller for the lysozyme/vacuum system, in agreement with previous results.[34] Apparently, the problem arises from the water molecules. For rigid water molecules (SPC), molecular cutoffs, not atomic cutoffs, are normally used to avoid splitting dipoles in electrostatic calculations. However, it is easier to use atomic cutoffs in FMM, because it is more natural to treat all atoms equally when assigning particles to cells on the tree. In our implementation, an atomic cutoff is used for the SPC water molecules. The fast translational and rotational motion of the small water molecules produces large charge fluctuations at the cell boundaries, which may induce large force fluctuations for those water molecules

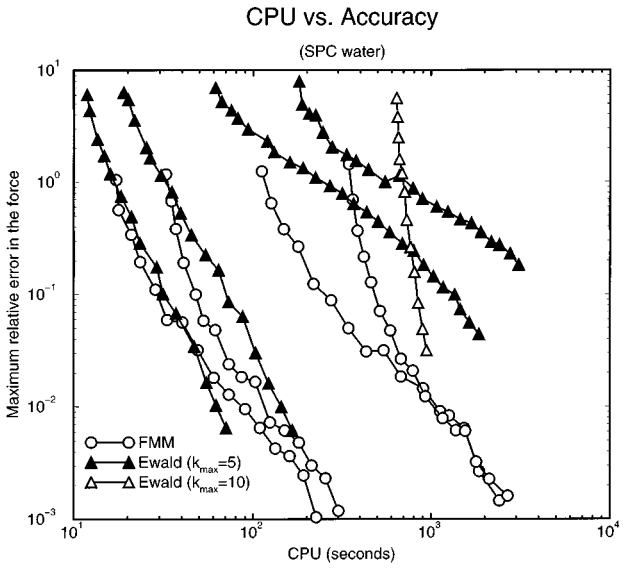

FIG. 3. Maximum relative error in the force vs CPU time (per step). The filled triangles show the results using the Ewald summation method with $k_{\max}=5$; the empty triangles were obtained with $k_{\max}=10$ (but only for the 14 817 molecule system); the empty circles correspond to the use of the FMM. From left to right, the data sets correspond to 2175, 3104, 14 817, and 28 886 SPC water molecules, respectively. From the top down, the Ewald results were obtained by varying the cutoff radius from $r_c=6$ to $r_c=30$ Å or less; the results for the FMM were obtained by varying the number of multipoles from $p=4$ to $p=19$. Note that the Ewald timings do not include the CPU time needed to generate the Verlet list of nonbonded interactions.

around the cell boundaries. Using a smooth cutoff reduces the abrupt force changes to some extent, thus significantly improving the energy conservation. The use of molecular cutoffs for water will improve the energy conservation, allowing the use of a smaller p than the $p=7$ now used for protein/water, thus saving CPU time.

VII. COMPARISON OF FMM WITH EWALD SUMMATION METHODS

A. Periodic FMM versus Ewald summation

Figure 3 shows a comparison of the CPU times per step for the Ewald method (triangles) and the FMM (circles). The calculations were performed on a Hewlett–Packard HPPA 9000/735 running *A*.09.01 at 100 MHz; the system studied consisted of pure SPC water with a varying number of molecules. From left to right, the lines correspond to systems with: 2175, 3104, 14 817, and 28 886 SPC molecules, respectively; the data for the Ewald method with $k_{\max}=10$ (white triangles) was obtained with the 14 817 molecule system. The various data points on each line were obtained as follows: for the FMM (circles) the number of levels was fixed ($L=3$ for the 2175 and 3104 molecule systems; $L=4$ for the others) and the maximum number of multipoles, p, was varied from 4 up to 19. For the Ewald method, the maximum number of k-space vectors ($k_{\max}$) was fixed (at 5 for the black triangles and 10 for the white triangles, with the convergence parameter α modified accordingly) and the real-space cutoff radius, r_c, was varied from 6 Å up in increments of 1 Å. For these runs, the time spent creating the Verlet list of nonbonded interactions was subtracted because the implementation we used was particularly slow; thus the Ewald timings represent a lower bound. In contrast, the timings for the FMM have not been corrected.

Several observations are worth making. The CPU time spent in the Ewald method (minus the generation of the Verlet list) is the sum of: (a) the CPU time spent in the real-space calculation, and (b) that spent in the reciprocal (k) space part. The former increases with increasing r_c; the latter with increasing $k_{\max}$.[57] For any given accuracy, there is always a compromise between these two terms, since one can always decrease r_c as long as both the convergence parameter α and $k_{\max}$ are increased accordingly. As a comparison between the two Ewald curves corresponding to the 14 817 molecule system shows, increasing $k_{\max}$ and α makes the curves steeper. For $k_{\max}$ large enough, the time (and accuracy) would be almost independent of r_c, since the real-space sums would decay very rapidly. However, for a given r_c, increasing $k_{\max}$ increases the CPU time, and that is reflected in the fact that the white triangles are shifted to the right.

For large p (number of multipoles), the CPU time for the FMM, for a fixed level L, is dominated by p and not by the number of particles in the system. This is clearly shown by the curves corresponding to 14 817 and 28 886 molecules: both have $L=4$ and they coalesce for p larger than about 13. At very high accuracy, the Ewald method with small cutoff r_c and large convergence parameter α is more efficient—at least for the 14 817 molecule case. For the larger system, the FMM will be faster.

As discussed in the preceding section, FMM with $p=7$ is accurate enough for simulations of solvated proteins, as measured by energy conservation. At this level (fourth circle from the top), the CPU time for the 14 817 molecule case is about 4.3 times faster than for the Ewald method with the same accuracy (note that the $k_{\max}=5$ and the $k_{\max}=10$ curves cross at about this point).

The combined r-RESPA/FMM algorithm is much faster than the standard Verlet/Ewald method for large solvated proteins with periodic boundary conditions. Comparative CPU timings of the solvated protein systems are listed in Tables II–IV. The CPU times for the Verlet/Ewald method vary between 27 and 562 ks/ps as the system size varies between 8000 and 44 000 atoms. In contrast, the CPU times for the r-RESPA/FMM method vary between 8 and 24 ks/ps, respectively, with similar levels of accuracy in energy conservation. This gives a factor of 3 to 23 in CPU time saving for systems with about 8000 to 44 000 atoms at the same accuracy. The CPU time savings will be even more promising for even larger systems, such as solvated nucleic acids. In summary, we observe that r-RESPA/FMM provides a computational advantage over standard Verlet/Ewald even for the smallest macromolecular system we have simulated, of about 8000 atoms.

TABLE V. CPU timing comparison of FMM vs SPME for a 40 Å water box (2038 molecules) system. The CPU time of simple cutoff at $r_c = 10$ Å is used as reference. Both Coulombic and van der Walls interactions are included. The data for SPME is taken from Essman *et al.*'s article [J. Chem. Phys. **103**, 8577, (1995)]. The results were obtained using the Verlet integrator.

Method	Δf	CPU (s/step)	ratio
CUT ($r_c = 10$ Å)	—	7.24	
SPME[a]	5×10^{-4}	7.36	1.01
CUT ($r_c = 10$ Å)	—	9.07	
FMM ($p=4$)[b]	1.68×10^{-3}	21.26	2.34
FMM ($p=6$)	4.32×10^{-4}	24.88	2.74
FMM ($p=8$)	1.25×10^{-4}	32.72	3.60

[a]Timing for SGI-R4400, a cutoff of 9 Å in direct sum (a cutoff larger than 9 Å is necessary for van der Waals forces) and 4th order interpolation with $36\times36\times36$ grid used.
[b]Timing for IBM R6000 SP2, level $L=3$, multipole terms p as specified.

B. Comment on FMM versus PME

Recently, the particle mesh Ewald method (PME), and a smooth variant (SPME), developed by Darden *et al.*, have been described in the literature.[21,23–25] They are based on Hockney and Eastwood's[58] idea of assigning charges to a mesh according to their real space positions; the CPU time savings come from applying the fast Fourier transform (FFT) to the particle mesh to accelerate the reciprocal-space calculations of the Ewald sum. The algorithms are found to be of order $O(N \log N)$.

There are three different algorithms for the calculation of the electrostatic forces in systems with periodic boundary conditions: (a) the (optimized) Ewald method, which scales like $O(N^{3/2})$, provided a fast pair list generation algorithm is used; (b) the PME, which scales like $O(N \log N)$; and (c) the periodic FMM, which scales like $O(N)$. For very large systems ($N > 10^5$), it is expected that the FMM will be the best choice, given its linear algorithmic complexity. However, what about systems of 10 000 to 100 000 atoms, which are currently feasible in computer simulation?

Table V gives CPU time comparisons between SPME[25] and FMM for a 40 Å system composed of 2038 SPC water molecules. All the results were obtained using the Verlet integrator. The CPU time of a simple spherical cutoff with $r_c = 10$ Å is used as reference for comparison. The data on SPME is taken from the study of Essman *et al.*[25] The timing studies show that the CPU times for SPME and the spherical 10 Å cutoff are comparable for an rms force accuracy of $\sim 5\times10^{-4}$. In contrast, FMM is about 3 times slower than the simple cutoff method at 10 Å, for a similar accuracy (using multipoles up to $p=6$). Thus, the FMM method is approximately 3 times slower than SPME for a system of $\sim$6000 atoms.

Table VI gives a comparison of CPU times for SPME and FMM for solvated proteins. The simple cutoff is again used as reference. The data for SPME is taken from Procacci *et al.*[59] Two similar size protein systems are compared in Table VI. For the FMM, levels $L=3$ and $L=4$ are used for the 8412 atoms and 22 661 atoms systems, respectively; multipole terms up to $p=6$ are used for both systems. The

TABLE VI. CPU timing comparison of FMM vs SPME for solvated protein system. The CPU time of simple cutoff at $r_c = 10$ Å is used as reference. The data for SPME is taken from Procacci *et al.*'s article (preprint). CPU time for SPME refers to DEC-Alpha 3000/800 s workstations; and CPU time for FMM refers to IBM R6000 SP2 workstations. A level $L=3$ for the 8412 atoms system and $L=4$ for the 22 661 atoms system, and multipole order $p=6$ for both are used in FMM.

Atoms	Method	Δt (fs)	r_c (Å)	$\log \Delta E$	$R=\frac{\Delta E}{\Delta KE}$	CPU (10^3 s/ps)	ratio
7070[a]	CUT	1	10.0		0.018	6.06	
	SPME	1	10.0		0.018	9.65	1.59
	R-SPME	12	10.0		0.036	2.59	0.43
8412[b]	CUT	1	10.0	−2.58		8.92	
	FMM	1	—	−3.68		35.04	3.92
	R-FMM	12	—	−3.27		8.48	0.95
20 627[c]	CUT	1	10.0			18.5	
	SPME	1	10.0			28.7	1.55
	R-SPME	12	10.0			7.7	0.42
22 661[d]	CUT	1	10.0	−2.60		31.50	
	FMM	1	—	−3.75		94.17	2.98
	R-FMM	12	—	−3.11		16.48	0.52

[a]*C*-Pycocyanin (3033 atoms) in 1335 water molecules.
[b]Ribonuclease H (2466 atoms) in 1982 water molecules.
[c]Rhodobacterial Sphaeroides (8321 atoms) in 4101 water molecules.
[d]Arabinose-binding protein (4691 atoms) in 5990 water molecules.

bonding and van der Waals forces are also included in both SPME and FMM when performing real MD simulations. The data for the FMM are collected from 1 ps MD simulations. The results show that the SPME is about 1.6 times slower than the simple cutoff method at $r_c = 10$ Å for systems with between 7070 and 20 627 atoms. In contrast, the FMM is between 3.9 and 2.6 times slower than the simple cutoff method for comparable system sizes (8412 and 22 661 atoms, respectively). Thus the FMM is still slightly slower than SPME for 22 000 atoms. Unfortunately, SPME data for larger systems is not available.

However, as shown in Table 6, after combining the *r*-RESPA method with SPME[59] and FMM (this article), the particle-mesh Ewald and fast multipole methods have comparable performance for systems with $\sim$20 000 atoms. They are both about twice as fast as a simple spherical cutoff ($r_c = 10$ Å). The SPME benefits most from the use of a very small real space part (as small as $r_c = 6$ Å) so that fast Fourier transforms, which speed up the *k*-space part, can be used to maximum advantage. In contrast, in order to gain the most from *r*-RESPA, it is desirable to use a large cutoff for the real space sum. Thus, there is a trade off between the SPME and *r*-RESPA methods. While the two methods appear to be competitive for $\sim$20 000 atoms, we expect that the RESPA/FMM will be faster than the RESPA/SPME method for even larger systems, because of their respective computational complexities.

VIII. CONCLUSIONS

We have extended the FMM method to periodic systems, with a full derivation of the local field transformation due to all distant multipoles in the periodic replicas. Our transformation, embodied in Eq. (29), can be easily reduced

to the normal Ewald sum under proper limits ($l=0$, $\mathbf{r}=0$, and no self-energy); however, we were unable to reduce the corresponding transformation in Ref. 49 to the proper Ewald limit. After combining the periodic FMM with r-RESPA, we devised a new molecular dynamics algorithm for charged systems, such as solvated proteins, with periodic boundary conditions. This combination of a reversible multiple time step integrator and an efficient algorithm for calculating long-range electrostatic interactions has been shown to be a powerful method for such simulations. The speedup of r-RESPA/FMM over the standard Verlet integrator with the Ewald sum is more than 20 for a protein water system with 44 000 atoms. As compared to the recently developed particle mesh Ewald (PME), the r-RESPA/FMM method has a break even point with the r-RESPA/PME method for systems with approximately 20 000 atoms. It is expected that the r-RESPA/FMM will be faster than the r-RESPA/PME for larger systems, since the asymptotic computational complexity is $O(N \log N)$ for PME and $O(N)$ for FMM.

In comparison with the results of a previous article,[34] the introduction of an aqueous solvent requires more accurate calculations than for isolated proteins in order to achieve the same level of energy conservation. This we attribute to the fast translational and rotational motions of the small water molecules which can produce large charge fluctuations when crossing cell boundaries, and the use of an atom-based cutoff instead of molecule-based cutoff for water molecules in FMM cell separations and r-RESPA force breakups.

It has also been shown that the use of a smooth switching function to effect the force separation for the direct interactions allows the use of very long time steps. The current implementation of smooth switching is suboptimal and results in a method that is about twice as slow as the cell-based separation for a given time step. Several avenues for improvement are under investigation, such as: the use of Verlet-like lists of interactions, and use of cubic rather than spherically symmetric switching functions.

The periodic r-RESPA/FMM described in this article has been implemented inside the molecular mechanics package IMPACT.[38] The resultant speed up has made possible for us to carry out nanosecond time scale simulations of pH-dependent effects on protein stability (manuscript in preparation).

Finally, it should be noted that the underlying electrostatic model is a periodic form of the Coulomb potential (the Wigner potential). There are artifacts associated with the periodic models, as there are with any computer models for macroscopic systems. The nature of the artifacts associated with the Wigner potential is a topic of current interest.[60–63]

ACKNOWLEDGMENTS

This work has been supported in part by Grant Nos. NIH GM30580 (R.M.L.), NIH GM43340 (B.J.B.), and by a grant to the Center for Biomolecular Simulation at Columbia University (NIH RR068R), and by an NSF MetaCenter Grant No. (MCA 93S021P) of computer time to R.M.L.

APPENDIX A: RECIPROCAL-SPACE SUM

In this appendix we derive a reciprocal-space representation for the sum

$$A=\int_0^{\alpha^2} dt t^{l+1/2-1} \sum_{\mathbf{n}\neq\mathbf{0}} \|\mathbf{n}\|^l Y_l^m(\hat{\mathbf{n}}) e^{-(s+t)\|\mathbf{n}\|^2}. \tag{49}$$

We first note that for any $l>0$ the integral converges when $\mathbf{n}=\mathbf{0}$ and vanishes identically. We then rewrite the sum using the Jacobi identity

$$\sum_{\mathbf{n}\neq\mathbf{0}} f(\mathbf{n}) = \sum_{\mathbf{k}} \hat{f}(2\pi\mathbf{k}) \tag{50}$$

where the Fourier transform $\hat{f}$ is defined by

$$\hat{f}(\mathbf{u}) = \int d\mathbf{x} e^{-i\langle \mathbf{u},\mathbf{x}\rangle} f(\mathbf{x}). \tag{51}$$

Using this identity, we obtain the expression

$$A=\sum_{\mathbf{k}} \int_0^{\alpha^2} dt t^{l+1/2-1} \times \int d\mathbf{y} e^{-2\pi i\langle \mathbf{k},\mathbf{y}\rangle} \|\mathbf{y}\|^l Y_l^m(\hat{\mathbf{y}}) e^{-(s+t)\|\mathbf{y}\|^2}. \tag{52}$$

For $\mathbf{k}\neq\mathbf{0}$, the integrals are finite in the limit $s\to 0$ as can be easily seen by a dimensional argument and the fact that the spatial integral will decay exponentially for $\|\mathbf{k}\|\to\infty$.

If $l>0$ and $\mathbf{k}=\mathbf{0}$, the integral

$$\int d\mathbf{y} \|\mathbf{y}\|^l Y_l^m(\hat{\mathbf{y}}) e^{-(s+t)\|\mathbf{y}\|^2} \tag{53}$$

vanishes identically due to the spherical symmetry of the regularization function. For $l=0$, this is not true and it is known[51] that in this case there is a divergence when $s\to 0$ that cancels only if the system is electrically neutral. In our treatment this cancelation is automatic since there is no $l=0$ (monopole) term to worry about. If the regularization function is not spherically symmetric, the above argument does not hold and we must proceed with more care. It is still true that the limit $s\to 0$ do not diverge for $l>0$, but there is a finite contribution for $l=1$. However, as discussed previously, we do not need the sums for odd l. To compute the Fourier coefficients for $\mathbf{k}\neq\mathbf{0}$, we make use of the expansion (Ref. 64)

$$e^{-2\pi i\langle \mathbf{k},\mathbf{x}\rangle} = \sum_{l=0}^{p} i^l (2l+1) j_l(2\pi\|\mathbf{k}\|\|\mathbf{x}\|) \times \sum_{m=-l}^{l} Y_l^m(\hat{\mathbf{k}}) Y_l^{-m}(\hat{\mathbf{x}}), \tag{54}$$

where $j_l(u)$ is the spherical Bessel function of order l, which can be written as

$$j_l(u) = \sqrt{\frac{\pi}{2u}} J_{l+1/2}(u) = (-1)^l u^l \left(\frac{1}{u}\frac{d}{du}\right)^l \frac{\sin u}{u}. \tag{55}$$

Using this expansion, we find the Fourier coefficients

$$4\pi i^l Y_l^m(\hat{\mathbf{k}}) \int_0^\infty r^{l+2} j_l(2\pi\|\mathbf{k}\|r) e^{-tr^2} dr. \tag{56}$$

Thus, we obtain for $l>1$ and even (since for odd l the sum vanishes identically)

$$A = \sum_{\mathbf{k}} 4\pi i^l Y_l^m(\hat{\mathbf{k}}) \|2\pi\mathbf{k}\|^{l-2} \int_0^{\alpha^2/\|2\pi\mathbf{k}\|^2} dt t^{l+1/2-1} \times \int_0^\infty dr r^{l+2} j_l(r) e^{-tr^2}. \tag{57}$$

To compute the first integral, we expand the Bessel function in a Taylor series and integrate term by term, obtaining

$$\begin{aligned}
&\int_0^\infty e^{-tr^2} r^{l+2} j_l(r) dr \\
&= \sqrt{\frac{\pi}{2}} \int_0^\infty e^{-tr^2} r^{l+1/2+1} J_{l+1/2}(r) dr \\
&= \sqrt{\frac{\pi}{2}} \int_0^\infty e^{-tr^2} r^{l+1/2+1} \sum_{k\geqslant 0} \frac{(-1)^k \left(\frac{r}{2}\right)^{l+1/2+2k}}{k!\Gamma\left(k+l+\frac{1}{2}+1\right)} \\
&= \sqrt{\frac{\pi}{2}} \sum_{k\geqslant 0} \frac{(-1)^k}{2^{l+1/2+2k} k!\Gamma\left(k+l+\frac{1}{2}+1\right)} \\
&\quad \times \int_0^\infty e^{-tr^2} r^{2l+2k+2} dr \\
&= \sqrt{\frac{\pi}{2}} \sum_{k\geqslant 0} \frac{(-1)^k \Gamma\left(l+\frac{1}{2}+k+1\right)}{2^{l+1/2+2k} k!\Gamma\left(k+l+\frac{1}{2}+1\right) 2t^{l+1/2+k+1}} \\
&= \sqrt{\frac{\pi}{2}} \frac{e^{-1/4t}}{(2t)^{l+1/2+1}}.
\end{aligned} \tag{58}$$

The integral over t becomes then

$$\begin{aligned}
&\int_0^{\alpha^2/\|2\pi\mathbf{k}\|^2} dt t^{l+1/2-1} \sqrt{\frac{\pi}{2}} \frac{e^{-1/4t}}{(2t)^{l+1/2+1}} \\
&= \sqrt{\frac{\pi}{2}} \frac{1}{2^{l+3/2}} \int_0^{\alpha^2/\|2\pi\mathbf{k}\|^2} dt \frac{e^{-1/4t}}{t^2} = \frac{\sqrt{\pi} e^{-\pi^2\|\mathbf{k}\|^2/\alpha^2}}{2^l}.
\end{aligned} \tag{59}$$

Combining all the above expressions we arrive at the result

$$A = \sum_{\mathbf{k}\neq\mathbf{0}} \pi^{l-1/2} i^l Y_l^m(\hat{\mathbf{k}}) \|\mathbf{k}\|^{l-2} e^{-\pi^2\|\mathbf{k}\|^2/\alpha^2}. \tag{60}$$

APPENDIX B: IMPLEMENTATION DETAILS

In this appendix, we will briefly describe our implementation of the FMM, with and without periodic boundary conditions, and its integration with r-RESPA.

Our implementation of the FMM consists of two sets of routines: (a) the ''core'' routines, which are designed to be independent of the way a particular simulation program represents the system; and (b) some ''interface'' routines that provide the necessary glue to connect the core routines with the rest of the MD program. We have written this interface for the program IMPACT,[38] an MD program developed at Rutgers that is very well suited to the simulation of biomolecules in solution. Moreover, this interface was written in such a way that the FMM can be run in ''serial mode'' on a single workstation or in ''parallel mode,'' on either a cluster of workstations or a massively parallel computer. For portability, the parallel mode is based on the parallel virtual machine (PVM)[65] message-passing library; a measure of the ease of portability is the fact that essentially the same code runs on clusters of workstations, the IBM RS6000 SP2 and the Cray T3D. Only the interface needs to know whether it is running in serial or parallel mode; the core routines are independent of the mode.

The core routines take care of such chores as: building the tree structure; loading the particles onto the tree (only the charges and positions need to be known); computing the multipole moments; computing the local expansion coefficients. We also provide routines to compute the potential and field at an arbitrary point, but those computations are best left to the interface routines since the core routines do not know about, for instance, the integration algorithm. There is another, perhaps more important, reason for making the interface code responsible for the force computation. In biomolecular simulations, one needs to compute not just the electrostatic but also shorter range (Lennard-Jones, hydrogen bonds) forces. Typically these forces are handled by the use of a so-called list of nonbonded interactions,[1] which needs to be updated every so often. In our code, however, we take advantage of the fact that the particles have already been spatially sorted at the time they were loaded on the tree and, since in any case we must compute the direct electrostatic interactions with particles in the first and second neighbor shells of each cluster (at the lowest level), we can at the same time compute the short-range forces. One important point is the following: not all pairs of atoms participate in nonbonded interactions; some pairs are ''excluded'' because of chemical constraints (they are connected by one, two, or three bonds). It is more efficient, however, to disregard this complication during the force computation and simply subtract the unwanted components afterwards (this same approach was used by Board *et al.*).[15]

To implement the Schmidt and Lee[49] method, we only need to add a function that computes the sums in Eq. (18), minus the contribution from first and second neighbors. In our current implementation, we are limited to constant volume simulations and therefore this computation needs to be done only once and so we can afford to do it very accurately (for each l,m). There is one further subtlety: when using periodic boundary conditions the contributions from the excluded atom pairs that must be subtracted are not purely Coulombic, but are given by the Wigner potential. Since these atoms are very close to each other we approximate this

potential by the first few terms in its expansion in powers of r:

$$\phi_W(\mathbf{x}) \approx \frac{1}{r} + \frac{2\pi}{3V} r^2 + \frac{c}{L} + O(r^4). \tag{61}$$

The self-energies of the charges (given by the constant term in the above equation times the square of the charge) are also subtracted. The constant c is approximately equal to -942.20 kcal Å/mol. Although this prescription might be considered nonstandard, whether to subtract the full Wigner interaction or just the Coulombic term is a matter of parametrization of the ''effective'' bonding potentials. The bonding potentials used in biomolecular simulations have traditionally been parametrized depending on cutoff-based simulations, so in principle either prescription needs to be reparametrized. In practice, the difference between the two models will be small.

As discussed in Sec. V, we have implemented several different force separations in the FMM. For efficiency, each case is handled by a separate function. These functions operate at the level of single clusters, and to avoid unnecessary repeated tests a pointer to the appropriate function is stored in a structure at each step before the force computations. This allows for a very general interface, where the main driver routines are independent of details selected at run time and makes it easy to add new force evaluation routines. However, the maximum number of force separations (''stages''), currently three not counting the topmost level, is hard coded into the program.

In our implementation, each cluster keeps a record of the total number of particles that it contains in all its subclusters. It is then possible to code a ''poor man's adaptive'' version of the FMM, in which clusters with no particles in them are skipped in some of the most computationally expensive steps.[34] Our code allows the user to select at run time, through the use of a keyword in the input script, whether to use this adaptive version. Since most periodic systems we simulate are rather dense and homogeneous, no savings are obtained by using the adaptive method in this case. However, for highly inhomogeneous systems, it can amount to a large speed up factor.

The approach we have taken to the parallelization of the FMM is simpler than that of Board *et al.*,[15] but it is not fully scalable. Partly because we use the code in our local network of workstations, it was decided to try to minimize the communication overhead. One way to do this is to broadcast to the processing nodes (nodes for short) the coordinates of all atoms in the system and let them compute all the multipole expansions. This certainly involves multiplication of work, but it is usually a very fast step and is not a bottleneck. However, if every node were to compute all local expansions that would result in almost no savings at all since this is typically one of the most expensive steps. Besides, although each node might need to know all the multipolar expansions, it only needs to know the local expansions for a subset of clusters, those which will contribute to the local expansion at the center of the cluster that has been assigned to the node. There is still some overlap since the same local expansions will be computed by several nodes. If internode communication is fast it might be preferable to avoid these multiplicities of work, but for loosely connected networks of workstations this is probably not the case.

[1] L. Verlet, Phys. Rev. **159**, 98 (1967).
[2] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford University Press, Oxford, 1991).
[3] H. E. Alper and R. M. Levy, J. Chem. Phys. **91**, 1242 (1989).
[4] M. Belhadj, H. E. Alper, and R. M. Levy, Chem. Phys. Lett. **179**, 13 (1991).
[5] H. Schreiber and O. Steinhauser, Biochemistry **31**, 5856 (1992).
[6] H. Schreiber and O. Steinhauser, Chem. Phys. **168**, 75 (1992).
[7] D. M. York, T. A. Darden, and L. G. Pedersen, J. Chem. Phys. **99**, 8345 (1993).
[8] G. S. Del Buono, F. E. Figueirido, and R. M. Levy, Proteins: Structure, Function and Genetics **20**, 85 (1994).
[9] F. Figueirido, G. S. Del Buono, and R. M. Levy, Biophy. Chem. **51**, 235 (1994).
[10] S. W. Rick and B. J. Berne, J. Am. Chem. Soc. **116**, 3949 (1994).
[11] J. E. Roberts and J. Schnitker, J. Chem. Phys. **101**, 5024 (1994).
[12] T. Simonson, Chem. Phys. Lett. **250**, 450 (1996).
[13] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems* (The MIT Press, Cambridge, Massachusetts, 1988).
[14] L. Greengard and V. Rokhlin, Physica A **29**, 139 (1989).
[15] J. A. Board *et al.*, Chem. Phys. Lett. **198**, 89 (1992).
[16] J. Shimadá, H. Kaneko, and T. Takada, J. Comput. Chem. **15**, 28 (1994).
[17] C. A. White and M. Head-Gordon, J. Chem. Phys. **101**, 6593 (1994).
[18] H.-Q. Ding, N. Karasawa, and W. A. Goddard III, J. Chem. Phys. **97**, 4309 (1992).
[19] F. S. Lee and A. Warshel, J. Chem. Phys. **97**, 3100 (1992).
[20] M. Saito, Mol. Simul. **8**, 321 (1992).
[21] J. Shimada, H. Kaneko, and T. Takada, J. Comput. Chem. **14**, 867 (1993).
[22] A. M. Mathiowetz, A. Jain, N. Karasawa, and W. A. Goddard III, Proteins: Structure, Function and Genetics **20**, 227 (1994).
[23] T. A. Darden, D. M. York, and L. G. Pedersen, J. Chem. Phys. **98**, 10 089 (1993).
[24] H. G. Petersen, J. Chem. Phys. **103**, 3668 (1995).
[25] U. Essman *et al.*, J. Chem. Phys. **103**, 8577 (1995).
[26] J.-P. Ryckaert, G. Ciccotti, and H. J. Berendsen, J. Comput. Phys. **23**, 327 (1977).
[27] H. C. Andersen, J. Comput. Phys. **52**, 24 (1983).
[28] O. Teleman and B. Jönsson, J. Comput. Chem. **7**, 58 (1986).
[29] R. D. Swindoll and J. M. Haile, J. Comput. Phys. **53**, 400 (1984).
[30] M. E. Tuckerman, G. J. Martyna, and B. J. Berne, J. Chem. Phys. **93**, 1287 (1990).
[31] M. E. Tuckerman, G. J. Martyna, and B. J. Berne, J. Chem. Phys. **94**, 1465 (1990).
[32] M. E. Tuckerman, G. J. Martyna, and B. J. Berne, J. Chem. Phys. **94**, 6811 (1991).
[33] M. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **97**, 1990 (1992).
[34] R. Zhou and B. J. Berne, J. Chem. Phys. **103**, 9444 (1995).
[35] J. J. Biesiadecki and R. D. Skeel, J. Comput. Phys. **109**, 318 (1993).
[36] D. I. Okunbor and R. D. Skeel, J. Comput. Chem. **15**, 72 (1994).
[37] D. Janežič and F. Merzel, J. Chem. Inf. Comput. Sci. **35**, 321 (1995).
[38] D. B. Kitchen *et al.*, J. Comput. Chem. **11**, 1169 (1990).
[39] A. J. C. Ladd, Mol. Phys. **33**, 1039 (1977).
[40] M. Neumann, Mol. Phys. **60**, 225 (1987).
[41] A. W. Appel, SIAM (Soc. Ind. Appl. Math.) J. Sci. Stat. Comput. **6**, 85 (1985).
[42] J. E. Barnes and P. Hut, Nature **324**, 446 (1986).
[43] J. K. Salmon, Ph.D. thesis, California Institute of Technology, 1991.
[44] L. Hernquist, J. Comput. Phys. **87**, 137 (1990).
[45] J. Makino, J. Comput. Phys. **87**, 148 (1990).
[46] J. Makino, J. Comput. Phys. **88**, 393 (1990).
[47] J. E. Barnes, J. Comput. Phys. **87**, 161 (1990).
[48] M. O. Fenley, W. K. Olson, K. Chua, and A. Boschitsch, J. Comput. Chem. **17**, 976 (1996).
[49] K. E. Schmidt and M. A. Lee, J. Stat. Phys. **63**, 1223 (1991).
[50] K. Esselink, Comp. Phys. Comm. **87**, 375 (1995).

[51] S. W. de Leeuw, J. W. Perram, and E. R. Smith, Proc. R. Soc. London, Ser. **A**, 27 (1980).
[52] P. Procacci, and B. J. Berne, J. Chem. Phys. **101**, 2421 (1994).
[53] D. D. Humphreys, R. A. Friesner, and B. J. Berne, J. Phys. Chem. **98**, 6885 (1994).
[54] S. J. Stuart, R. Zhou, and B. J. Berne, J. Chem. Phys. **105**, 1426 (1996).
[55] M. Watanabe and M. Karplus, J. Chem. Phys. **99**, 8063 (1993).
[56] W. F. van Gunsteren and H. J. C. Berendsen, Mol. Phys. **34**, 1311 (1977).
[57] G. Hummer, Chem. Phys. Lett. **235**, 297 (1995).
[58] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (Adam Hilger, Bristol, 1989).
[59] P. Procacci, T. Darden, and M. Marchi, J. Chem. Phys. (in press).
[60] G. Hummer, L. R. Pratt, and A. García, J. Phys. Chem. **99**, 14 188 (1995).
[61] F. E. Figueirido, G. S. Del Buono, and R. M. Levy, J. Chem. Phys. **103**, 6133 (1995).
[62] G. S. Del Buono, F. E. Figueirido, and R. M. Levy, Chem. Phys. Lett. **263**, 521 (1996).
[63] G. Hummer, L. R. Pratt, and A. E. García, J. Phys. Chem. **100**, 1206 (1996).
[64] J. D. Jackson, *Classical Electrodynamics*, 2nd ed. (Wiley, New York, 1975).
[65] A. Geist *et al.*, Technical Report No. ORNL/TM-12187, Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN (unpublished).