A new molecular dynamics method combining the reference system propagator algorithm with a fast multipole method for simulating proteins and other complex systems

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

(Received 9 May 1995; accepted 30 August 1995)

An efficient molecular dynamics (MD) algorithm is presented in this paper for biomolecular systems, which incorporates a novel variation on the fast multipole method (FMM) coupled to the reversible reference system propagator algorithm (r-RESPA). A top-down FMM is proposed which calculates multipoles recursively from the top of the box tree instead of from the bottom in Greengard's original FMM, in an effort to be more efficient for noncubic or nonuniform systems. In addition, the use of noncubic box subdivisions of biomolecular systems is used and discussed. Reversible RESPA based on a Trotter factorization of the Liouville propagator in generating numerical integration schemes is coupled to the top-down FMM and applied to a MD study of proteins *in vacuo*, and is shown to be able to use a much larger time-step than the standard velocity Verlet method for a comparable level of accuracy. Furthermore, by using the FMM it becomes possible to perform MD simulations for very large biomolecules, since memory and CPU time requirements are now nearly of order of $O(N)$ instead of $O(N^2)$. For a protein with 9513 atoms (the photosynthetic reaction center), the efficient MD algorithm leads to 20-fold reduction in CPU time for the Coulomb interaction and approximately 15-fold reduction in total CPU time over the standard velocity Verlet algorithm with a direct evaluation of Coulomb forces.

I. INTRODUCTION

Molecular dynamics simulations of biological macromolecules[1,2] are computationally demanding, owing to the large number of particles, as well as the complex nature of their associated interactions. There are generally two approaches to simplify this enormous computational problem: One is to use simpler physical models for the protein, the other is to develop more efficient theories or numerical methods without modification of the physical model. In this paper, we will focus on the second approach, that is to develop some numerically more efficient methods to reduce the computational burden.

The most time-consuming part of the simulation is the calculation of the long-range pairwise Coulombic forces. The computational complexity for these Coulombic interactions scales as $O(N^2)$, where N is the number of particles. Since the interaction decays with distance as $O(r^{-1})$, it is normally not appropriate to use short cutoffs for electrostatic interactions, especially for large proteins where the long-range effects of inhomogeneous charge distributions are thought to be important.[3,4]

The second most time-consuming part of the simulation arises from the small time steps required to accurately solve the equation of motion for the stiff bond stretch and bond bending vibrations, even though one may be primarily interested in events that occur over a much longer time scale. In standard numerical integration methods, such as those of the Störmer–Verlet variety,[5,6] one is generally required to use a time step smaller than one femtosecond in order to maintain an acceptable level of accuracy in the integration of the equations of motion.

There have been a number of efforts to solve the first problem, i.e., to reduce the computational complexity for the electrostatic interactions. Hockney *et al.*[7] proposed a particle/mesh method to address this problem by using a mesh over the computational domain. The source density is interpolated at the mesh points, then the Poisson equation is solved to obtain the potential values on the mesh, and the forces are computed from the potential. The computational complexity is thus reduced from $O(N^2)$ to $O(N \log N)$. The mesh provides only limited resolution and highly nonuniform source distributions cause significant errors. To improve accuracy, a particle–particle/particle–mesh method was developed,[8] in which the potential arising from short-range interactions is calculated directly. Appel[9] and Barnes *et al.*[10] developed a ''tree code'' method in place of the grid methods, in which the computational region is organized into a tree structure. The tree structure makes it possible to develop a rapid systematic procedure to determine which particles are ''distant'' from each other. By exploiting the fact that a particle interacts with a distant group of particles much as if it were interacting with a single particle at the center of mass (monopole) of the distant group, the complexity is also reduced to $O(N \log N)$, although there is again some loss of accuracy by using the monopole approximation.

Greengard and Rokhlin[11] then developed the fast multipole method (FMM) based on the ''tree code'' idea, but with higher-order multipoles in addition to the simple monopole approximation. The FMM method first organizes multipole representations of charge distributions in hierarchically structured boxes, then transforms those multipoles into local field expansions, so that each particle interacts with the local field

 0021-9606/95/103(21)/9444/16/$6.00

generated from distant particles. The multipoles are generated by direct calculation in the lowest level and successive shifting from lower levels to upper levels. In this paper, we propose a top-down recursive method in generating the multipoles in the hierarchical box tree, in which the multipoles are calculated recursively from the top of the tree instead of from the bottom as in Greengard's FMM method. At each level of the tree, the method first looks for charged particles in every box. If there is no charged particles in a particular box, then the multipoles and local field expansions for that box and all its subdivided boxes, are assigned automatically to be zero without any further calculations. This is more efficient for nonuniform or noncubic systems, such as proteins.

A variety of techniques have also been introduced to address the second problem, i.e., to increase the time step in MD simulations. One common approach is to constrain bond lengths using either the SHAKE or RATTLE algorithms.[12,13] Although application of these methods allow for a modest increase in time-step, time-dependent quantities may be affected.[14,15] Additionally, the constraint methods have been shown to work poorly for bond angle degrees of freedom when applied to macromolecules.[14]

Another approach to increase the time step in MD simulations is that of the multiple time-step methods.[16] These methods are based upon integration schemes that allow for different time steps according to how rapidly a given type of interaction is evolving in time. Teleman and Jönsson[17] introduced an algorithm whereby the slower degrees of freedom are held constant for a number of smaller time steps, which is usually called the long-range constant force approximation. This method has, however, been shown to lead to the accumulation of numerical error in calculated quantities.[18,19] Alternatively, Swindoll and Haile[20] introduced a procedure which uses a Taylor series approximation for the less rapidly evolving forces. Although this algorithm has been shown to give some improvement in CPU times for simple systems such as alkane chain liquids,[20] it is not yet evident whether it would be computationally advantageous in the case of macromolecules. In this paper, a multiple time-step algorithm is designed specifically for macromolecular simulation which uses a combination of time steps of different lengths to integrate interactions which evolve on different time scales. The algorithm is essentially a generalization of the previously introduced reversible reference system propagation algorithm (r-RESPA),[21] which employs a Trotter factorization[22] of the classical Liouville operator as a means to derive a numerical propagation scheme for the system. This r-RESPA scheme is a time reversible, symplectic (measure conserving in phase space), and highly stable integrator. This approach has been shown to be considerably more efficient than standard techniques when applied to simple systems or small proteins.[19,21,23,24]

In this paper, we will present a new efficient MD algorithm, which uses the top-down recursive FMM with noncubic box subdivisions and reversible RESPA. A brief description of the theories will be summarized in Sec. II, followed by the computational implementation in Sec. III. Section IV will give some results of the application of the algorithm to MD simulations for proteins *in vacuo*, and Sec. V contains the conclusion. It is shown that the new MD algorithm is much more efficient than the standard methods. For a protein with 9513 atoms, the CPU time is reduced by a factor of 20 for the electrostatic interactions, and about 15 for total MD simulations with a comparable level of accuracy compared to velocity Verlet method.

II. THEORY

A. The fast multipole method

The fast multipole method (FMM) was introduced by Greengard and Rokhlin.[11] It is an efficient method for evaluating Coulombic interactions between a large number of particles. The CPU time increases linearly $[O(N)]$ rather than as square of the number of particles $[O(N^2)]$. In this section, a top-down recursive method is proposed for multipole generation. The method is based on White and Head–Gordon's simplified derivation.[25]

The basic principle of FMM is rather elegant. It interpolates the potential and force on a particular charge due to distant charges not by direct calculation, but by using the local expansion of fields produced by the multipoles generated from those distant charges. If first organizes multipole representations of charge distributions in hierarchically structured boxes, then transforms these multipoles into local field expansions. Each particle then interacts with the local field to count the interaction from distant particles. In this method the short-range interactions are calculated directly. The potential (and force) consists of two parts:

$$\Phi(\mathbf{x}) = \Phi_{\text{multipole}}(\mathbf{x}) + \Phi_{\text{direct}}(\mathbf{x}) \tag{1}$$

where Φ_{direct} contains the short-range interparticle interactions, $\Phi_{\text{multipole}}$ contains the contribution from distant particles.

Before giving the detailed mathematical formulas for the FMM, it may be useful to define some parameters and "terminology." Three parameters (n,p,ws) are used in FMM: n represents the number of levels in the box tree, where the system is divided into 8^n lowest-level boxes; p is the number of terms used in the multipole expansions, (e.g., $p=2$ includes contributions up to quadrupole, and $p=4$ includes contributions up to hexadecapoles); ws is the parameter which defines well-separated boxes, where $\text{ws}=1$ indicates that Φ_{direct} includes the contribution from the box itself and its 26 nearest-neighbor boxes, and $\text{ws}=2$ indicates Φ_{direct} includes contributions up to the second nearest-neighbor boxes at the lowest level in the tree. Box level 0 specifies the largest box which holds the whole system, while box level $l+1$ is obtained from level l by subdivision of each box into eight smaller equal-sized boxes. A tree structure is then imposed on this box hierarchy. The eight boxes at level $l+1$ obtained by subdivision of a box will be referred to its *children*, and the level l box is called their *parent* box.

We now turn to the theory of FMM. The method is based on the the expansion of the Coulombic potentials in multipoles. For two unit charges at $\mathbf{r}(r,\theta,\phi)$, and $\mathbf{a}(a,\alpha,\beta)$ the potential may be written as

$$\frac{1}{|\mathbf{r}-\mathbf{a}|}=\sum_{l=0}^{\infty} P_l(\cos\ \gamma)\ \frac{a^l}{r^{l+1}}$$
$$=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\ \frac{(l-m)!}{(l+m)!}\ \frac{a^l}{r^{l+1}}\ P_{lm}(\cos\ \alpha)$$
$$\times P_{lm}(\cos\ \theta)e^{-im(\beta-\phi)}$$
$$=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\ \frac{(l-|m|)!}{(l+|m|)!}\ \frac{a^l}{r^{l+1}}\ P_{l|m|}(\cos\ \alpha)$$
$$\times P_{l|m|}(\cos\ \theta)e^{-im(\beta-\phi)}, \tag{2}$$

where γ is the subtended angle between $\mathbf{r}$ and $\mathbf{a}$ $(a<r)$. The expansion separates the coordinates of the two particles in terms of associated Legendre polynomials. After redefining the associated Legendre polynomials,[25]

$$\tilde{P}_{lm}\equiv(l-m)!P_{lm} \tag{3a}$$

$$\tilde{\tilde{P}}_{lm}\equiv\frac{1}{(l+m)!}\ P_{lm}, \tag{3b}$$

the multipole moments and electric potentials can be expressed more compactly.[25] We can now define the moments of a multipole expansion about the origin (the box center) of a charge (or charge distribution) at $\mathbf{a}$ as

$$M_{lm}(\mathbf{a})=a^l\tilde{P}_{lm}(\cos\ \alpha)e^{-im\beta}, \tag{4a}$$

$$M_{lm}(q;\mathbf{a})=qM_{lm}(\mathbf{a}), \tag{4b}$$

$$M_{lm}(Q;\mathbf{A})=\sum\ M_{lm}(q;\mathbf{a}), \tag{4c}$$

where $M_{lm}(\mathbf{a})$ are the multipole moments about an origin (such as a box center) for a unit charge located at $\mathbf{a}$ with respect to that origin, $M_{lm}(q;\mathbf{a})$ are the multipole moments for a charge q at $\mathbf{a}$, and $M_{lm}(Q;\mathbf{A})$ are the multipole moments for many charges in a box expanded around a common origin (box center). We use capital letters to denote collective sets of charges and positions in a box. Since these multipole moments $M_{lm}(Q;\mathbf{A})$ are all expanded with respect to the same origin, they can of course be summed directly.[26] The corresponding Taylor coefficients for a local field expansion of the potential due to a charge (or charge distribution) at $\mathbf{r}$ are then

$$L_{lm}(\mathbf{r})=\frac{1}{r^{l+1}}\ \tilde{\tilde{P}}_{lm}(\cos\ \theta)e^{im\phi}, \tag{5a}$$

$$L_{lm}(q;\mathbf{r})=qL_{lm}(\mathbf{r}), \tag{5b}$$

$$L_{lm}(Q;\mathbf{R})=\sum\ L_{lm}(q;\mathbf{r}), \tag{5c}$$

where the indices have the same meaning as those in the multipole moments, except the charge is now located at $\mathbf{r}$ respect to the origin. Then the potential at $\mathbf{r}$ due a charge at $\mathbf{a}$, or vice versa, could be expressed as

$$\frac{q}{|\mathbf{r}-\mathbf{a}|}=\sum_{l=0}^{\infty}\sum_{m=-l}^{m=l} M_{lm}(q;\mathbf{a})L_{lm}(\mathbf{r}) \tag{6a}$$

$$=\sum_{l=0}^{\infty}\sum_{m=-l}^{m=l} M_{lm}(\mathbf{a})L_{lm}(q;\mathbf{r}). \tag{6b}$$

The expansion of above equation is exact in the limit of an infinite sum. However, it is normally not necessary to add up very high-order multipoles or local expansions, and a truncation of $p=3$ or $p=4$ is usually sufficient for simulations which do not require extraordinarily high accuracies.[11,27] Also, it has been shown that the same level of truncation is appropriate for both the multipole and the local expansions.

The essential idea of FMM is to build a hierarchical structure of multipoles, a tree structure, with each multipole containing the contribution of a subset of charges of limited spatial extent. The smallest box on the lowest level of the tree will then contain only a small number of particles (3–20). The boxes on successively higher levels (parent boxes) are the union of eight lower level children boxes, until one single box on the highest level contains all other boxes, and therefore all charges.

The multipoles associated with lowest level boxes are calculated according to Eqs. (4a)–(4c) directly. Multipoles of higher level boxes are calculated not from charges, but from the shifting of lower level multipoles. This is usually called an ''upward pass,''

$$M_{lm}(Q;\mathbf{A}+\mathbf{b})=\sum_{j=0}^{l}\sum_{k=-j}^{j} T_{lm,jk}^{MM}(\mathbf{b})M_{jk}(Q;\mathbf{A}), \tag{7}$$

where $\mathbf{b}$ is the displacement of the multipole expansion center and $T_{lm,jk}^{MM}$ is the shift translation operator for multipole to multipole

$$T_{lm,jk}^{MM}(\mathbf{b})=M_{l-j,m-k}(\mathbf{b}). \tag{8}$$

The new version of FMM proposed here, the top-down recursive FMM, however, calculates multipoles recursively from the top of the tree instead of from the bottom as in ''upward pass'' in FMM. It is a top-down recursive generation of the multipoles specially designed for biosystems, such as proteins, since these biomolecules are usually nonuniform and noncubic systems. By using the top-down generation scheme, it is easier to cut some ''branch'' of the tree if there are no charges in it. At each level in the recursive calculation, it first looks for charged particles in each box. If the number of charges in a box is zero then it skips all multipole calculations for the box, its children, grandchildren..., as well as all the corresponding local field translations described below. The top-down recursive calculation allows this pruning operation systematically, which is more efficient for noncubic or nonuniform systems.

After the multipoles M_{lm} of all boxes are obtained, a local expansion L_{lm} is constructed for each box, describing the potential inside the box caused by all distant charges

$$L_{lm}(Q;\mathbf{A}-\mathbf{b})=\sum_{j=0}^{\infty}\sum_{k=-l}^{l} T_{lm,jk}^{LM}(\mathbf{b})M_{jk}(Q;\mathbf{A}), \tag{9}$$

and the transformation operator is

$$T^{LM}_{lm,jk}(\mathbf{b}) = L_{l+j,m+k}(\mathbf{b}). \tag{10}$$

Another essential idea in the FMM is the efficient use of higher-level local expansions to reduce the effort in subsequent transformations, since these transformations are the most CPU time consuming calculations in the FMM. The local expansions for a given level n are actually calculated using

$$L^{(n)}_{lm} = L'^{(n)}_{lm} + L^{(n-1)}_{lm}, \tag{11}$$

where $L'^{(n)}_{lm}$ is the local expansions from the present level's (level n) multipoles which are not already contained in $L^{(n-1)}_{lm}$. For example, if there is a box B_1 at a given level, only the multipoles in boxes at this level which are well separated from B_1, but are not well separated from B_1's parent are transformed into local expansions in B_1. The number of these multipoles is never larger than $6^3-3^3=189$ for ws=1, and $10^3-5^3=875$ for ws=2.

The shift of a higher level expansion to a lower level is done much like the multipole shifts from a lower level to a higher level. This is usually called a ''downward pass,''

$$L_{lm}(Q;\mathbf{R}-\mathbf{b}) = \sum_{j=l}^{\infty} \sum_{k=-j}^{j} T^{LL}_{lm,jk}(\mathbf{b}) L_{jk}(Q;\mathbf{R}), \tag{12}$$

and the shift translation operator from local expansion to local expansion is

$$T^{LL}_{lm,jk}(\mathbf{b}) = M_{j-l,k-m}(\mathbf{b}). \tag{13}$$

Once the local expansions at the lowest level are known, the electrostatic potential and the forces are easily calculated by using Eqs. (1) and (6).

As mentioned above, in the new version of the FMM, we have used a top-down recursive method to generate the multipole expansion, in an effort to be more efficient for noncubic or nonuniform systems, such as biosystems. Another way to save CPU time in the FMM is to use noncubic boxes to subdivide noncubic systems. Using noncubic boxes, such as rectangular boxes, may help to reduce the number of vacant boxes in the tree, and also reduce the number of pairs in neighbor boxes which need to be calculated directly. Thus, the CPU time required can be further reduced for noncubic systems. Care must be taken however when using noncubic boxes because the accuracy of the potential goes as

$$\left|\Phi(r) - \frac{q}{|\mathbf{r}-\mathbf{a}|}\right| = |q| \sum_{l=p+1}^{\infty} P_l(\cos\ \gamma)\left(\frac{a}{r}\right)^{p+1} \leqslant \frac{|q|}{r-a}\left(\frac{a}{r}\right)^{p+1}, \tag{14}$$

where **a** refers to the positions for particles inside a lowest-level box, and **r** refers to the positions for particles in well-separated boxes. It is clear that the largest error will come from maximizing the ratio of a/r. For a lowest-level cubic box with side $2d$, this maximum ratio occurs for $a=\sqrt{3}d$, and $r=3d$ (for ws=2) or $r=5d$ (for ws=2). Then, the error bound can be expressed as

$$|\Phi_{\text{exact}} - \Phi_{\text{ws}=1}| \leqslant \frac{|q|}{(3-\sqrt{3})d}\left(\frac{1}{\sqrt{3}}\right)^{p+1}, \tag{15a}$$

$$|\Phi_{\text{exact}} - \Phi_{\text{ws}=2}| \leqslant \frac{|q|}{(5-\sqrt{3})d}\left(\frac{\sqrt{3}}{5}\right)^{p+1}. \tag{15b}$$

However, if we use rectangular boxes with sides $d_1 \leqslant d_2 \leqslant d_3$, $a=\sqrt{3}d$ should now be replaced by $a=(d_1^2+d_2^2+d_3^2)^{1/2}$, and $r=3d$ ($r=5d$) should be replaced by $r=3d_1$ ($r=5d_1$) (d_1 is the smallest dimension in d_1, d_2, and d_3 for a rectangular box). If d_1 is much less than other two dimensions, for example in linear molecules like some polymers, the accuracy may go down when we use higher p, because now a may be larger than r in some particular cases. So for these linear molecules, it may not be appropriate to use noncubic boxes. One way to fix this problem may be to use several adjoining level 0 boxes to hold the whole molecule, instead of using only one level 0 box. The possibility of using several boxes to hold one polymer molecule is under investigation. Of course, the top-down recursive method is also useful for these linear molecules.

Since the three dimensions for most proteins are normally comparable (the difference is less than a factor of three for all proteins studied here), and also since we do not require very high p (as described in Sec. IV), noncubic boxes could be used to divide protein molecules, with a little loss of accuracy. Thus both the top-down recursive method and noncubic boxes are used in the following simulations (for simplicity, we call this new version of the fast multipole method ''TFMM'' in following sections).

B. The reversible reference system propagator algorithm

1. The Trotter expansion of the Liouville propagator

The reference system propagator algorithm (RESPA) for molecular dynamics was first introduced by Tuckerman, Martyna, and Berne.[18,21] It has been shown to be much more efficient than standard techniques, such as the velocity Verlet method. The early applications were applied to simple systems with stiff and soft forces, short- and long-range interactions, and disparate masses.[21,28] Recently, this algorithm has been successfully applied to diatomic molecules in solution,[29] small organic molecules by Watanabe and Karplus,[23] the fullerene crystal by Procacci and Berne,[24] and a small protein by Humphreys and Berne.[19] In this paper, we will couple the algorithm to the fast multipole method and apply it to very large proteins.

The reversible RESPA[28] is based upon the Trotter expansion of the classical Liouville propagator. The Liouville operator L for a system of N degrees of freedom in Cartesian coordinates is defined as

$$iL = [\ldots, H] = \sum_{i=1}^{N} \left[\dot{x}_i \frac{\partial}{\partial x_i} + F_i(x) \frac{\partial}{\partial p_i}\right], \tag{16}$$

where [...,...] is the Poisson bracket, H is the Hamiltonian, (x_i, p_i) is the position and conjugate momentum for the co-

ordinate i. The state of the system at a time t, $\Gamma(t)$, is defined as the collective set of positions and conjugate momenta $\{x(t),p(t)\}$. It evolves with time as

$$\Gamma(t)=U(t)\Gamma(0), \tag{17}$$

where $U(t)$ is the classical time evolution propagator

$$U(t)=e^{itL}. \tag{18}$$

Since the classical Liouville operator is self-adjoint, $U(t)$ is a unitary operator and the time evolution in above equation is reversible.

We may then choose to decompose the Liouville operator into two parts, such that

$$L=L_1+L_2. \tag{19}$$

This allows us to apply the Trotter theorem,[22] giving

$$\begin{aligned} e^{it(L_1+L_2)}&=[e^{i(t/N)(L_1+L_2)}]^N \\ &=[e^{i(\Delta t/2)L_2}e^{i\Delta tL_1}e^{i(\Delta t/2)L_2}]^N+O(\Delta t^3), \end{aligned} \tag{20}$$

where $\Delta t\equiv t/N$. In practice, Δt is chosen small enough, i.e., N large enough, to generate an accurate and stable MD simulation. One may then define a discrete time propagator as

$$\begin{aligned} G(\Delta t)&=U_2(\Delta t/2)U_1(\Delta t)U_2(\Delta t/2) \\ &=e^{i(\Delta t/2)L_2}e^{i\Delta tL_1}e^{i(\Delta t/2)L_2}. \end{aligned} \tag{21}$$

The inner propagator in Eq. (21) can be further decomposed as

$$\begin{aligned} e^{i\Delta tL_1}&=[e^{i\delta\tau L_1}]^n \\ &=[e^{(\delta\tau_1/2)F_1(x)(\partial/\partial p)}e^{\delta\tau_1\dot{x}(\partial/\partial x)}e^{(\delta\tau_1/2)F_1(x)(\partial/\partial p)}]^n \\ &\quad+O(\delta\tau_1^3), \end{aligned} \tag{22}$$

if $L_1=\dot{x}(\partial/\partial x)+F_1(x)(\partial/\partial p)$. This provides us with a means for determining the time evolution of a system whose interactions evolve according to two different time scales. That is, the inner propagator $e^{i\Delta tL_1}$ may be taken to be associated with the rapidly varying interactions with a smaller time-step, which we call as a ''reference'' system propagator. The outer propagators ($e^{i\Delta t/2L_2}$) are used to evolve the slowly varying interactions with a larger time step, which is named as a ''correction'' propagator. The formal solution for the discretized equations of motion is then given by

$$\begin{aligned} \Gamma(\Delta t)&=U_2(\Delta t/2)U_1(\Delta t)U_2(\Delta t/2)\Gamma(0)+O(\Delta t^3) \\ &=e^{i(n\delta\tau/2)L_2}[e^{i\delta\tau L_1}]^n e^{i(n\delta\tau/2)L_2}\Gamma(0)+O(\Delta t^3). \end{aligned} \tag{23}$$

This approach can be easily expanded to a general case with more than two effective time scales. Suppose we have m different time scales for a particular system, so we choose to break the Liouville operator into a sum of m terms,

$$L=L_1+L_2+\cdots+L_m. \tag{24}$$

It follows that the entire discretized propagator for a system with m time scales can be written as

$$\begin{aligned} G(\Delta t)\equiv& e^{i(n_1n_2\ldots n_{m-1}\delta\tau_1/2)L_m}\ldots \\ &\times[e^{i(n_1\delta\tau_1/2)L_2}[e^{i\delta\tau_1L_1}]^{n_1}e^{i(n_1\delta\tau_1/2)L_2}]^{n_2}\ldots \\ &\times e^{i(n_1n_2\ldots n_{m-1}\delta\tau_1/2)L_m}. \end{aligned} \tag{25}$$

The m different timescales are given by $\delta\tau_1$, $n_1\delta\tau_1$, $n_1n_2\delta\tau_1,\ldots,$ $n_1n_2\ldots n_{m-1}\delta\tau_1$. This corresponds to a situation where the innermost reference propagator is evaluated every small time step $\delta\tau_1$, the second innermost correction propagator is evaluated every n_1 small steps, and so on, with the mth correction propagator evaluated only every $n_1n_2\ldots n_{m-1}$ small time-steps.

2. Reversible reference system propagation algorithm (r-RESPA) for biomolecules

In order to apply the techniques discussed above to the MD simulation of biomolecules, we take the Liouville operator for a macromolecule *in vacuo* containing N atoms to be

$$iL=\sum_i^{3N}\left[\dot{x}_i\frac{\partial}{\partial x_i}+F_i(x)\frac{\partial}{\partial p_i}\right], \tag{26}$$

where

$$\begin{aligned} F(x)&=F_{\text{stret}}(x)+F_{\text{bend}}(x)+F_{\text{tors}}(x)+F_{\text{Hbond}}(x) \\ &\quad+F_{\text{vdW}}(x)+F_{\text{elec}}(x), \end{aligned} \tag{27}$$

F_{stret}, F_{bend}, F_{tors}, F_{Hbond}, F_{vdW}, and F_{elec} represent the forces for stretch, bending, torsion (including improper torsion), hydrogen bonding, van der Waals, and electrostatic interactions, respectively. Their functional forms can be found elsewhere.[30,31] The databases of parameters for these functional forms are generally called force fields. There are several force fields available for biomolecular simulations, such as CHARMM,[3] AMBER,[31] OPLS,[32] etc.

In an atomic level simulation using force fields, the stretch vibrations are usually the fastest motions in the molecular dynamics of biomolecules, so we use the evolution of the stretch vibration as a ''reference'' propagator with the smallest time scale. The nonbonded interaction, including van der Waals and electrostatic forces, are the slowest varying interactions, and a much larger time step may be used. The bending, torsion and hydrogen-bonding forces are treated as intermediate time-scale interactions.

In addition, the nonbonded forces can be divided into several regions in pair distance according to their importance and varying speeds. The near region is normally more important than the distant region because the nonbonded forces decay with distance. Since most of the CPU time in a MD simulation is spent in the calculation of these nonbonded interactions, the separation in pair distance results in valuable speedups.

Using a three-fold distance split, the nonbonded forces are separated in three regions: near, medium, and far distance zones. Thus, the Liouville operator can be express as a sum of five terms

$$L=L_1+L_2+L_3+L_4+L_5, \tag{28}$$

where

$$iL_1 \equiv \dot{x}\,\frac{\partial}{\partial x} + F_1(x)\,\frac{\partial}{\partial p}, \tag{29a}$$

$$iL_i \equiv F_i(x)\,\frac{\partial}{\partial p}, \quad i=2,3,4,5, \tag{29b}$$

and

$$F_1(x) \equiv F_{\text{stret}}(x), \tag{30a}$$

$$F_2(x) \equiv F_{\text{bend}}(x) + F_{\text{tors}}(x) + F_{\text{Hbond}}(x), \tag{30b}$$

$$F_3(x) \equiv F_{\text{vdW}}^{\text{near}}(x) + F_{\text{elec}}^{\text{near}}(x), \tag{30c}$$

$$F_4(x) \equiv F_{\text{vdW}}^{\text{med}}(x) + F_{\text{elec}}^{\text{med}}, \tag{30d}$$

$$F_5(x) \equiv F_{\text{vdW}}^{\text{far}}(x) + F_{\text{elec}}^{\text{far}}. \tag{30e}$$

To separate the nonbonded forces into near, medium, and far zones, pair distance separations are used for the van der Waals forces, and box separations are used for the electrostatic forces since the box separation is a more convenient breakup in the TFMM method.

The distance separation for van der Waals forces is most easily implemented by using switching functions.[21,28] For example, if we want to separate the pairwise interaction F_{vdW} into m subsegments in the pair distance regions, i.e., $[0-r_1]$, $[r_1-r_2]$,..., and $[r_{m-1}-r_m]$, we may write the van der Waals force as

$$F_{\text{vdW}}(x) = \sum_{k=1}^{m} F_{\text{vdW}}^{k}(x) \equiv \sum_{k=1}^{m} [S_k(r) - S_{k-1}(r)] F_{\text{vdW}}(x), \tag{31}$$

where $S_k(r)$ are switching functions which are defined as

$$S_0(r) \equiv 0; \quad S_m(r) \equiv 1.0$$

and

$$S_k(r) = \begin{cases} 1, & 0 \leqslant r < r_k - \Delta r_k, \\ 1 - R_k^2(3-2R_k), & r_k - \Delta r_k \leqslant r \leqslant r_k, \\ 0, & r_k \leqslant r. \end{cases} \tag{32}$$

Here $R_k \equiv [r-(r_k-\Delta r_k)]/\Delta r_k$, r is the interatomic distance, r_k is the kth distance cutoff, and Δr_k is the kth healing length. The analytical form for the switching function is arbitrary, and the only requirement is that it and its first derivative are to be continuous, at r_k and $r_k-\Delta r_k$. The reason to use switching functions is to avoid the sudden changes in forces when crossing between different distance regions. This ensures that the force in the first region $(0<r\leqslant r_1)$ decreases smoothly to zero at r_1, and in other regions, such as in the kth region, it increases smoothly from zero at r_{k-1} and decreases smoothly to zero at r_k. Using this method, highly stable molecular dynamics simulations are possible.

The box separation used for the electrostatic interactions is taken to be consistent with the box division in the TFMM. For a particular box in the lowest-level, the box and its 26 nearest boxes are regarded as the near zone, its 98 (5^3-3^3) second-nearest neighbor boxes as the medium zone, and all the other boxes as the far zone. As we will see in Sec. IV, using ws=2 in the TFMM is even faster than ws=1 if a high accuracy level is required. Thus, for the box separation of the electrostatic forces, using ws=2 may be a better choice than using ws=1, since it is more convenient to calculate the forces from near and medium zones directly, and forces from the far zone by local field expansions from distant multipoles, which is exactly what is done in the TFMM with ws=2.

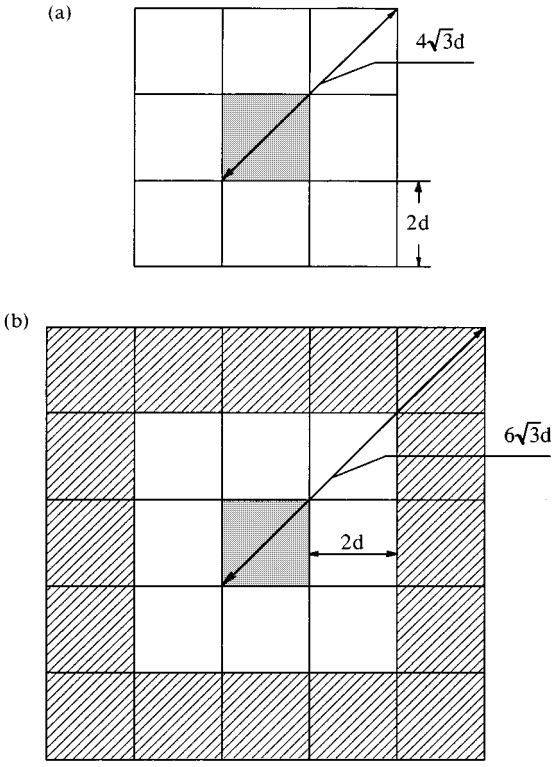

FIG. 1. A diagram showing the box separation for the electrostatic forces. (a) The near zone: a lowest-level box (center box) and its 26 nearest-neighbor boxes. (b) The medium zone: its 98 second-nearest-neighbor boxes (dashed boxes).

A diagram for the near and medium zones for the electrostatic forces is given in Fig. 1. Cubic boxes were used for simplicity. Assuming that the side-length for the smallest box in the tree is $2d$, the pair distance in the near zone, as shown in Fig. 1(a), ranges from 0 to $4\sqrt{3}d$; the medium zone, as shown in Fig. 1(b), ranges from $2d$ to $6\sqrt{3}d$; and the far zone ranges from $4d$ to infinity. Since these zones are constructed from cubic boxes, they are not spherical shells, therefore these zones overlap in distance. The overlap region will be wider if rectangular boxes are used. The pair numbers in the overlap region decrease to zero slowly for the nearer zone since there are few pairs of atoms which are in opposing corners, and increase slowly from a small number for the next zone for a similar reason. For example, in the overlap region $(2d-4\sqrt{3}d)$ of the near and medium zones, the pair numbers for the near zone decreases slowly to zero when the pair distance goes to $4\sqrt{3}d$, and the pair numbers for the medium zone increases slowly from a small number when the distance increases from $2d$. This means that the pair numbers in the overlap regions behave as a sort of switching function, so that no explicit switching functions are used for the electrostatic forces. The following MD simulations presented in Sec. IV show that we can generate very stable MD

simulations using the box separation for electrostatic forces without an explicit switching function.

After separating the nonbonded forces in the three distance regions, we may write the discretized propagator as

$$\begin{aligned} G(\Delta t) \equiv & e^{i(n_1 n_2 n_3 n_4 \delta\tau_1/2)L_5}\left[e^{i(n_1 n_2 n_3 \delta\tau_1/2)L_4}\right. \\ & \times\left[e^{i(n_1 n_2 \delta\tau_1/2)L_3}\left[e^{i(n_1 \delta\tau_1/2)L_2}\left[e^{i\delta\tau_1 L_1}\right]^{n_1}\right.\right. \\ & \times e^{i(n_1\delta\tau_1/2)L_2}\left.\left.\right]^{n_2} e^{i(n_1 n_2 \delta\tau_1/2)L_3}\right]^{n_3} \\ & \left.\times e^{i(n_1 n_2 n_3 \delta\tau_1/2)L_4}\right]^{n_4} e^{i(n_1 n_2 n_3 n_4 \delta\tau_1/2)L_5}, \end{aligned} \tag{33}$$

where L_1, L_2, L_3, L_4, and L_5 are given by Eqs. (29a) and (29b). The inner ''reference'' propagator in Eq. (33), which contains the bond stretching vibrations evolution, is given by

$$e^{i\delta\tau_1 L_1} = \exp\left[\delta\tau_1\left(\dot{x}\,\frac{\partial}{\partial x} + F_1(x)\,\frac{\partial}{\partial p}\right)\right]. \tag{34}$$

This can be further expanded by using the following Trotter factorization

$$\begin{aligned} &\exp\left[\delta\tau_1\left(\dot{x}\,\frac{\partial}{\partial x} + F_1(x)\,\frac{\partial}{\partial p}\right)\right] \\ &\quad = \exp\left[(\delta\tau_1/2)F_1(x)\,\frac{\partial}{\partial p}\right]\exp\left(\delta\tau_1\dot{x}\,\frac{\partial}{\partial x}\right) \\ &\qquad \times \exp\left[(\delta\tau_1/2)F_1(x)\,\frac{\partial}{\partial p}\right] + O(\delta\tau_1^3). \end{aligned} \tag{35}$$

As has been shown in Refs. 21 and 28 this factorization is equivalent to the velocity Verlet algorithm. The outer ''correction'' propagators $e^{(i\delta\tau_m/2)L_m}$, for $m=2$, 3, 4, 5, are of the form

$$e^{(\delta\tau_m/2)F_m(x)(\partial/\partial p)}. \tag{36}$$

After acting to the right on an arbitrary state $\{x,p\}$, one find that

$$\exp\left[\left(\frac{\delta\tau_m}{2}\right)F_m(x)\,\frac{\partial}{\partial p}\right]\{x,p\} = \{x, p + (\delta\tau_m/2)F_m(x)\}. \tag{37}$$

Thus, the evolution of the system is determined numerically by acting with the propagator in Eq. (33) to the right on the initial state $\{x(0),p(0)\}$, using Eqs. (37) and (35).

In the following sections we apply the new MD algorithm based on the combination of TFMM and r-RESPA to simulations of proteins *in vacuo*. We use the TFMM to calculate electrostatic interactions for large protein molecules and we use the propagator in Eq. (33) to generate the integration scheme, with a reference propagator of the form in Eq. (35). The results are discussed and compared with that of the standard velocity Verlet integrator.

III. COMPUTATIONAL IMPLEMENTATION

The potential parameters used in this paper are adopted from the AMBER[31] force field, although our program is designed to be capable of using various force fields. The full long-range Coulomb potentials ($O(r^{-1})$) were used in the simulation by taking advantage of the fast multipole algorithm. For the short-range van der Waals potentials ($O(r^{-6})$), a cutoff distance of $r_c = 12$ Å was used. The cutoff was made such that the forces and their first derivatives go smoothly to zero at r_c by using a switching function as in Eq. (32). The calculation results show that $r_c = 12$ Å is large enough for the van der Waals potentials.

Normally, molecular simulations using force fields exclude nonbonding forces between atoms that are considered to be chemically bonded. But, as we have seen above, it is more convenient to calculate potentials and forces for all point charges in the FMM, so we need to subtract the chemically bonded pair contributions from the FMM results. Usually, (1,2) stretching and (1,3) bending interactions are excluded totally in all force fields, but (1,4) torsional interactions are treated slightly differently in various force fields due to different choices of parameters. In the AMBER force field, only $\frac{1}{2}$ of the (1,4) interactions are subtracted.

The FMM algorithm was implemented as a portable module using the C programming language. It uses a tree structure of boxes to handle the multipoles and Taylor expansions. For example, a specific box in the tree can be described as ''box$[n][i][j][k]$'' in C language syntax, where n is the level in the tree, and i,j,k are indices of the box in x, y, and z directions respectively, which can range from 0 to 2^n. Then the eight nearest neighbors can be easily accessed by $i' = i + \{1,0,-1\}$; $j' = j + \{1,0,-1\}$; and $k' = k + \{1,0,-1\}$. Furthermore, its parent is now just box$[n-1][i/2][j/2][k/2]$, and its eight children are given by box $[n+1][2i + \{0,1\}][2j + \{0,1\}][2k + \{0,1\}]$. All the multipoles and Taylor expansions can be well organized through this structure for use in the ''upward'' and ''downward'' passes. In our top-down FMM, the charges in each box $[n][i][j][k]$ are first checked during the recursive multipole calculation from the top of tree. If the charge is zero, then the multipole and corresponding Taylor expansion coefficients for this box and all its children and grandchildren are assigned to be zero without any further calculations.

The FMM module is then called by a MD module which is written in FORTRAN. The implementation of the r-RESPA algorithm in the MD module is quite straightforward, and a schematic FORTRAN code can be found elsewhere.[19] All simulations were performed on IBM RISC 6000/model 580 and 590 computers.

IV. RESULTS AND DISCUSSIONS

Six protein systems *in vacuo* are used in this paper to test the new MD algorithm: a 292-atom fraction of insulin (4insb), crambin (655 atoms, 1crn), interleukin 8 (1144 atoms, 3il8), ribonuclease-H (2470 atoms, 2rn2), L-* arabinose-Binding Protein (4674 atoms, 8abp), and the Photosynthetic Reaction Center (including the active branch only, i.e., subunit C and L, 9513 atoms, 1prc).

Before performing a production MD simulation, we need to apply some primary ''treatments'' to the initial structure—x-ray structure from Brookhaven PDB file, with addition of explicit H atoms. First, we minimize the x-ray structure using the conjugate gradient method to obtain a minimum energy structure. This is necessary because AMBER and other available force fields, while reasonable, are not sufficiently accurate to give exact structures, and the explicit H atoms

TABLE I. Tree level n and the average particle numbers in the finnest-level box N_0 used in FMM for various proteins with total atom number N.

Protein	4insb	1crn	3il8	2rn2	8abp	1prc
N	292	655	1144	2470	4674	9513
n	2	2	3	3	3	4
N_0	4.56	10.23	2.24	4.82	9.13	2.32

added by the program may not in correct positions. Typically, it takes several thousand iterations to minimize a 1000-atom protein, and tens of thousands of iterations to minimize a 5000-atom protein. The rms deviation for the minimized structure compared to the initial x-ray structure is usually very small, only 0.5–1.0 Å. The initial velocities are then sampled from a Maxwell–Boltzmann distribution at a given initial temperature, such as 100 K. In order to avoid having the structure blow up, the minimized structure is slowly heated up to 300 K from 100 K over a 10 ps MD run. This is then followed by a 20 ps MD run at 300 K (canonical ensemble) for equilibration. During the equilibration the velocities are resampled from a Maxwell–Boltzmann distribution periodically if the average temperature over the previous 0.5 ps deviated from 300 K by more than ± 5 K.

A. Optimum parameters for TFMM

As we discussed in Sec. II A, three parameters (n, p,ws) are used in the FMM. In this section, we will determine the optimum values for these parameters in our MD simulation.

The total number of tree levels n is determined by

$$n = \mathrm{int}(\log_8^{N/N_0}), \tag{38}$$

where the int function returns the integer part, N is the number of total atoms, and N_0 is the desired average particle number in the finest-level box. Generally, using a larger N_0 requires more multipole terms to obtain a given level of accuracy, and thus more CPU time; on the other hand, using a smaller N_0 may result in more tree levels, and thus more CPU overheads in FMM as well. Various simulations have shown that setting N_0 to 2–16 is an optimum choice.[11,25,27,33] Similar results are found by our calculation. The optimum tree level n and the average particle number in the finest-level box N_0 for various proteins are listed in Table I.

For the well-separated parameter ws, it is obvious that ws=1 is faster but less accurate than ws=2 for the same p level. Thus, we may ask which is more efficient for a given level of accuracy: to use ws=1 with a higher p or use ws=2 with a lower p. In order to address this question, we define two measures of accuracy. One is the relative error in the potential, the other is the relative error in the forces:

$$\Delta\Phi = \frac{|\Phi_{\mathrm{direct}} - \Phi_{\mathrm{TFMM}}|}{|\Phi_{\mathrm{direct}}|} \tag{39a}$$

$$\Delta F = \frac{\Sigma |\mathbf{F}^{(i)}_{\mathrm{direct}} - \mathbf{F}^{(i)}_{\mathrm{TFMM}}|^2}{\Sigma |\mathbf{F}^{(i)}_{\mathrm{direct}}|^2}. \tag{39b}$$

Some numerical results for both ws=1 and ws=2 are listed in Table II. For ws=1, ΔF is $10^{-2}-10^{-3}$ for $p=4$; while for ws=2, ΔF is about $10^{-3}-10^{-4}$ for $p=4$. Thus, ws=2 is approximately ten times as accurate as ws=1. To reach the same accuracy as that in ws=2 with $p=4$, multipoles up to

TABLE II. Accuracy and speed up of the new version of the fast multipole method for various proteins in both ws=1 and ws=2 cases. The CPU time of the direct method is also included for comparison. For each column, the three numbers are CPU time, $\Delta\Phi$ and ΔF_{rms} respectively. All CPU times (seconds) are obtained from IBM RISC6000/MODEL 590 machines.

		ws=1			ws=2		
Protein	Direct	$p=2$	$p=4$	$p=8$	$p=2$	$p=4$	$p=8$
4insb	0.06	0.08	0.14	0.42	0.10	0.14	0.26
(292)		8.740E-3	3.045E-3	4.266E-4	3.928E-4	6.222E-4	3.169E-5
		2.132E-2	1.108E-2	6.284E-3	2.547E-4	7.046E-4	7.393E-5
1crn	0.30	0.24	0.31	0.74	0.38	0.44	0.65
(655)		5.318E-4	4.696E-5	5.755E-6	3.488E-4	4.021E-6	1.209E-7
		1.067E-2	3.035E-3	4.700E-4	1.055E-3	1.521E-4	4.575E-6
3il8	1.02	0.64	0.72	1.13	1.01	1.09	1.46
(1144)		3.520E-4	7.194E-5	2.648E-5	6.290E-5	7.497E-6	1.624E-7
		9.914E-3	3.566E-3	1.371E-3	9.698E-4	1.789E-4	9.968E-6
2rn2	4.98	1.42	2.55	8.32	2.50	3.48	8.24
(2470)		1.052E-4	6.798E-5	4.934E-5	2.039E-4	1.707E-5	1.487E-6
		2.507E-2	1.020E-2	3.271E-3	7.898E-3	2.327E-3	3.742E-5
8abp	17.48	2.81	3.99	10.29	7.05	8.04	13.13
(4674)		1.095E-4	9.061E-5	2.843E-5	1.398E-4	2.831E-5	2.229E-6
		2.155E-2	8.345E-3	2.367E-3	6.933E-3	1.823E-3	2.536E-4
1prc	76.13	7.32	14.37	52.52	13.06	20.26	60.03
(9513)		1.720E-4	1.167E-4	1.606E-5	2.076E-4	8.478E-6	8.170E-6
		3.491E-2	1.780E-2	9.655E-3	1.328E-2	4.313E-3	1.125E-3

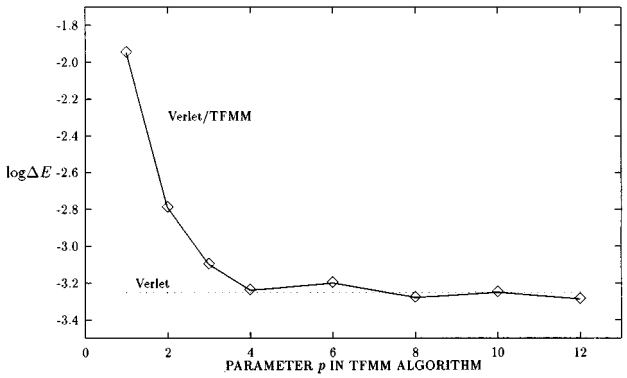

FIG. 2. Dependence of energy conservation on the parameter p used in the FMM algorithm which is now incorporated in MD module, ($p=2$, includes contributions up to quadrupole, $p=4$, to hexadecapoles).

$p=4$, multipoles up to $p=8$ should be used in ws=1, which requires more CPU time. This is consistent with Greengard's[11] and Head-Gordon's[25] results. Both of them claimed that ws=2 was the optimum choice. Also, we note from Table II that the crossover point for the TFMM vs direct evaluation is about 1000 atoms for ws=2 at an accuracy level of $\Delta\Phi\sim10^{-4}-10^{-5}$ and $\Delta F\sim10^{-3}-10^{-4}$, which is comparable or even better than previously reported results.[11,25,27,33]

In addition, since we may use distance separations in the r-RESPA algorithm, ws=2 is the better choice for our MD simulations because we can easily use the nearest-neighbor boxes as the near zone, the second-nearest shell of box as the medium zone and all the other boxes as the far zone.

To address the parameter p, we need to determine what p level is sufficient to generate a stable MD simulation, i.e., to avoid possible accumulation of errors in MD runs. Two energy conservation parameters are commonly used to describe the stability of a constant-energy MD simulation.[19,23] One is the total energy fluctuation ΔE defined by

$$\Delta E\equiv\frac{1}{N_T}\sum_{i=1}^{N_T}\left|\frac{E_{\text{initial}}-E_i}{E_{\text{initial}}}\right|, \tag{40}$$

where E_i is the total energy at step i, E_{initial} 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,[19,23] and a value of $\Delta E\leqslant0.003$, i.e., $\log(\Delta E)<-2.5$, gives an acceptable numerical accuracy. Another common measure of the accuracy is the ratio of the rms deviation of the total energy to the rms deviation of the kinetic energy,[34]

$$R\equiv\frac{\Delta E_{\text{rms}}}{\Delta KE_{\text{rms}}}. \tag{41}$$

A value of $R\leqslant0.05$, can be used as an alternate criterion for stability in MD simulations.[19,23]

The protein ribonuclease-H (2rn2) is used as an example to perform constant energy (microcanonical ensemble) MD simulations to determine the necessary p level. Figure 2 shows the dependence of $\log(\Delta E)$ with p for 1 ps of MD simulation using the standard velocity Verlet integration scheme with the new version of FMM implemented for Coulomb forces. The result for the direct electrostatic calculation is also included for comparison. It is found that $p=4$, which includes contributions up to hexadecapoles, is sufficient for a stable constant-energy MD simulation. So in the following simulations, $p=4$ is used.

At this point, the three parameters used in TFMM have been optimized for our MD simulation. The tree levels n for various proteins are listed in Table I (N_0 equals to 2–10). The other two parameters are set to $p=4$ and ws=2 for all proteins studied.

B. Energy conservation comparison

In this section, we will compare the energy conservation for three different methods, r-RESPA, velocity Verlet, and constant long-range force approximation (CLFA).

In r-RESPA, we separate forces according to their intrinsic different time scales to increase the overall time step. We use the notation (n_1,n_2,n_3,n_4) to indicate different combinations of time-scale separation. That is, if the time step is δt for stretching forces, then time step is

$n_1\delta t$ for bending, torsion and H-bond forces,

$n_1n_2\delta t$ for near zone van der Waals and electrostatic forces,

$n_1n_2n_3\delta t$ for medium zone van der Waals and electrostatic forces,

$n_1n_2n_3n_4\delta t$ for far zone van der Waals and electrostatic forces.

In dividing the near, medium, and far zones, we use pair distance separations for van der Waals forces and box separations for electrostatic forces, as described in Sec. II. The results of the calculation show that $r_1=7-9$ Å (with healing length 1.5–2.0 Å) is a good choice for the near zone in dividing the van der Waals forces using a switching function. A value of $r_1=8.0$ Å is used in the following simulations. The pair distance region (8–12 Å) is defined as the medium zone, and no far zone of van der Waals forces is actually included here because of the cutoff at 12.0 Å, which is sufficiently large for van der Waals forces.

Box separation for the electrostatic forces is more convenient within the FMM (ws=2 is used). For simplicity, we consider a cubic-box subdivision. The side length ($2d$) for the smallest box in the tree, which contains 2 to 10 atoms on average, is from 4 to 6 Å for all proteins studied here, i.e., $d=2.0$ to 3.0 Å. Using an average value of $d=2.5$ Å for estimation, the pair distance for the electrostatic near zone, which includes the smallest box and its 26 nearest-neighbor boxes, ranges from 0 to $4\sqrt{3}d$ (0 to 17.3 Å). The medium zone encompasses the second nearest-neighbor boxes, ranging from $2d$ to $6\sqrt{3}d$ (5.0 to 26.0 Å), and the far zone from $4d$ to infinity ($\geqslant$10.0 Å). The electrostatic forces in the near and medium zones are evaluated directly, while the contributions from the far zone are evaluated by local field expansions from distant multipoles. No explicit switching functions are used for electrostatic forces, as has been mentioned before, since these zones overlap in distance and the pair numbers in the overlap region behave as a sort of switching function.

Physically, the separation of nonbonded forces on the basis of the pair distance is a kind of ''short/long'' range

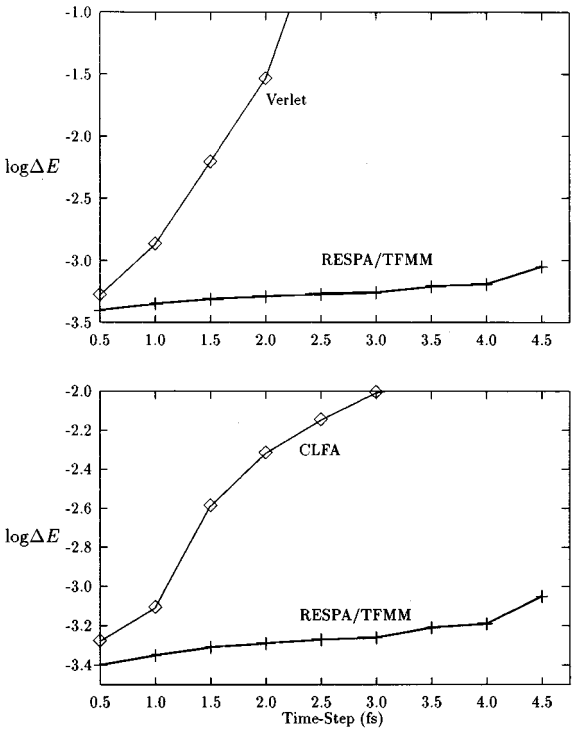

FIG. 3. Comparison of the energy conservation for various methods in a 1 ps constant-energy MD runs for the protein ribonuclease-H: (a) r-RESPA/TFMM vs the velocity Verlet. (b) r-RESPA/TFMM vs the constant long-range force approximation (CLFA).

TABLE III. Comparison of energy conservation and associated CPU times for velocity Verlet, constant long-range force approximation (CLFA), and r-RESPA/TFMM. Here, Δt (fs) is the overall time step (the smallest time step is 0.5 fs for CLFA and 0.25 fs for r-RESPA/TFMM, $\{n\}$ represents the combinations of separations in CLFA and r-RESPA). T_{total} is the total CPU time spent (in seconds) in all force routines. All data are collected from 1 ps MD runs for protein ribonuclease-H in IBM RISC6000/MODEL 590 machines. The asterisk indicates that the structure blows up when $\Delta t=3.0$ fs in Verlet method.

Method	$\{n\}$	Δt	log(ΔE)	R	T_{total}
Verlet	⋯	0.50	−3.2861	0.0374	16776.7
	⋯	1.00	−2.8746	0.0715	8363.9
	⋯	1.50	−2.2127	0.0934	6401.7
	⋯	2.00	−1.5436	0.2966	4218.6
	⋯	3.00	⋯*	⋯	
CLFA	1	0.50	−3.2861	0.0374	16776.7
	2	1.00	−3.0181	0.0574	9386.8
	3	1.50	−2.5984	0.1708	6753.3
	4	2.00	−2.3232	0.1885	5514.8
	6	3.00	−2.0180	0.4415	4495.4
	8	4.00	−1.8654	0.8730	3651.5
RESPA/	(1,1,1,2)	0.50	−3.4171	0.0247	14575.2
TFMM	(1,1,2,2)	1.00	−3.3592	0.0371	9281.6
	(1,1,2,3)	1.50	−3.3100	0.0343	8285.4
	(1,2,2,2)	2.00	−3.2962	0.0330	4895.0
	(1,2,2,3)	3.00	−3.2690	0.0329	3977.9
	(2,2,2,2)	4.00	−3.1980	0.0378	2520.7

breakup, the separation of bonded forces from nonbonded forces is an ''internal/external'' force breakup, and the separation of stretching vibrations from other bonded interactions is a ''stiff/soft'' force breakup. These three separations have been shown to be ideal for the application of the r-RESPA method.[28,29] It is the difference in intrinsic time scales in these breakups that make the r-RESPA algorithm valuable and powerful.

Figure 3(a) shows the energy conservation performances of velocity Verlet and r-RESPA/TFMM. Here, again, we studied protein ribonuclease-H as an example, with 1 ps MD runs for both methods. The curve for r-RESPA/TFMM is obtained from various combinations of (n_1, n_2, n_3, n_4) with smallest time step $\delta t=0.25$ fs. The overall time step is then given by $\Delta t=n_1n_2n_3n_4\delta t$. For example, (1,1,2,2) gives a time step of 1.0 fs, and (2,2,2,2) gives a time step of 4.0 fs. The results indicate that for a similar accuracy level, the r-RESPA/TFMM method is able to use a time step nearly 8–9 times larger than that of velocity Verlet. For velocity Verlet, E_{total} starts drifting to higher energies with time when the time step exceeds 1.0 fs, whereas r-RESPA/TFMM is quite stable even for an overall time step as large as 4.0 fs.

It is also interesting to compare r-RESPA/TFMM with a frequently used approximation whereby one treats the long range force as effectively constant over a number of time steps n while a standard integrator such as velocity Verlet is used. We denote this method as the constant long-range force approximation (CLFA). In order to make the comparison, we choose to employ the identical ''short/long''-range force breakup for nonbonded forces in two methods. That is, we choose the near zone as ''short'' range ($r\leqslant 8.0$ Å for vdW, and nearest-neighbor boxes for electrostatic force), and medium and far zones as ''long'' range. In both region there are contributions from vdW and electrostatic forces. The overall time step for CLFA is then $n\delta t$. The energy conservation parameter log (ΔE) is plotted as a function of overall time step in Fig. 3(b) for both CLFA and r-RESPA/TFMM. The results indicate that the constant long-range approximation leads to very poor energy conservations in this case, whereas r-RESPA/TFMM remains quite stable to significantly larger time steps.

Furthermore, our r-RESPA/TFMM is even faster than CLFA for time steps larger than 2.0 fs. It should be mentioned that log (ΔE) will decrease to some extent for CLFA if we use a small time-step of $\delta t=0.25$ fs in CLFA, or use both near and medium zones as ''short'' range, but log (ΔE) still increases much faster with the overall time step than r-RESPA/TFMM, i.e., it is not as stable as r-RESPA/TFMM. And, of course, the CPU time required in CLFA increases at the same time, which is absolutely not desired.

Table III summarizes some results from the three different methods velocity Verlet, CLFA, and r-RESPA/TFMM for comparison.

C. Spectral density comparison

To explore the question of whether r-RESPA/TFMM does indeed generate the correct dynamics for the system, we compare a spectral density $I(\tilde{\nu})$ as a function of the frequency $\tilde{\nu}$ in wave numbers, obtained from the two methods, velocity Verlet and r-RESPA/TFMM, where

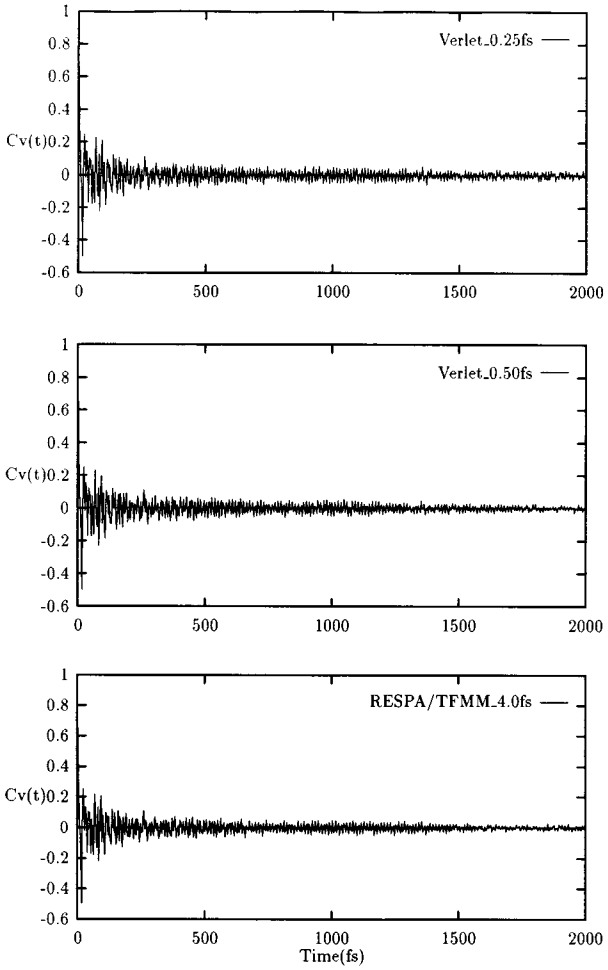

FIG. 4. The velocity autocorrelation function $C_v(t)$ as a function of time for three cases: (a) velocity Verlet with $\Delta t=0.25$ fs, (b) velocity Verlet with $\Delta t=0.50$ fs, and (c) r-RESPA/TFMM with time step 4.0 fs [combination of (2,2,2,2) with $\delta t_1=0.25$ fs].

$$I(\tilde{\nu})=\int_0^\infty C_{\mathbf{v}}(t)\cos(2\pi c\tilde{\nu}t)dt \tag{42}$$

and $C_{\mathbf{v}}(t)$ is the normalized velocity autocorrelation function of the system,

$$C_{\mathbf{v}}(t)=\frac{\langle\Sigma_{i=1}^{N}\mathbf{v}_i(t)\cdot\mathbf{v}_i(0)\rangle}{\langle\Sigma_{i=1}^{N}\mathbf{v}_i(0)\cdot\mathbf{v}_i(0)\rangle}. \tag{43}$$

Here c is the speed of light, $\mathbf{v}_i(t)=(v_x(t),\ v_y(t),\ v_z(t))$ is the velocity of atom i at time t, and N is the total number of atoms. As an example, we use protein interleukin 8 (3il8) for the spectrum simulation. The velocity autocorrelation function and its infrared spectrum are obtained from 5 ps MD runs for three different cases:

(1) velocity Verlet with time step 0.25 fs (Verlet_0.25 fs)
(2) velocity Verlet with time step 0.50 fs (Verlet_0.5 fs)
(3) r-RESPA/TFMM with time step 4.0 fs, (2,2,2,2) combination (RESPA/TFMM_4.0 fs).

We assume the Verlet_0.25 fs case represents the ''exact'' result. Figure 4 shows the autocorrelation functions for the three cases (only 2 ps is plotted). They look similar in Fig. 4, but the details shown in Fig. 5 indicate that the autocorrelation function of Verlet_0.50 fs actually differs from the Verlet_0.25 fs after 0.5 ps. On the other hand, RESPA/TFMM_4.0 fs stays very close to Verlet_0.25 fs for 1.2 ps. These differences in autocorrelation functions result in the differences between the three corresponding spectral profiles given in Fig. 6. The sharp peak at 2955–2985 cm^{-1} is due to the C–H stretch vibrations, and the small peak near 3320 cm^{-1} are due to the hydrogen-bonded O–H stretches, while the small shoulders around 600–1200 cm^{-1} and small peaks around 1500 cm^{-1} belong to various bending modes, as well as C–C and C═O stretch vibrations.

In order to establish a quantitative estimate of the accuracy of the resulting spectral densities, we consider[24,35]

$$D\equiv\arccos\left(\frac{\mathbf{S}_1\cdot\mathbf{S}_2}{|\mathbf{S}_1||\mathbf{S}_2|}\right), \tag{44}$$

where

$$\mathbf{S}=(s_1,\ldots,s_n), \tag{45}$$

and the s_i are the spectral components at frequency i. The quantity D in the above equation can be viewed as the angle between the vectors $\mathbf{S}_1$ and $\mathbf{S}_2$. If the two spectra are identical, then $D=0$, whereas, if they are uncorrelated, $D=\pi/2$. We take as a reference, the ''exact'' spectral density to be defined as that obtained from a MD simulation using the velocity Verlet integrator with a time step of 0.25 fs. We then calculate D with respect to this reference spectral density for the other two cases. We obtain $D=0.549$ for that of velocity Verlet using a time-step of 0.5 fs, and $D=0.109$, for r-RESPA/TFMM with an overall time step 4.0 fs. Thus, r-RESPA/TFMM not only reduces the CPU time, but also gets better spectra compared to the velocity Verlet integrator using a time step of 0.5 fs.

The poor D value for the velocity Verlet with $\Delta t=0.5$ fs can be attributed to a numerically induced ''blue shift'' evident at the higher frequencies of the spectral density.[19,24] To illustrate this, Fig. 7 shows the detailed spectra for the three cases in four different frequency regions. The differences between the three cases are small in the frequency region 600–1600 cm^{-1}, but the Verlet_0.50 fs spectrum starts to differ from Verlet_0.25 fs at higher frequencies. The sharp C–H stretch peak at 2954 cm^{-1} shifts to 2961 cm^{-1}, the hydrogen-bonded O–H stretch shifts from 3321 to 3332 cm^{-1}, and the free O–H stretch shifts from 3728 to 3745 cm^{-1}. The RESPA/TFMM_4.0 fs spectrum agrees with Verlet_0.25 fs very well, and no evident shifts are found for these peaks. This indicates that the smallest time step for stretching is critical in generating the correct infrared spectra. Also, the fact that Verlet_0.50 fs agrees well with Verlet_0.25 fs at low frequencies, but differs at higher frequencies (the higher the frequency, the larger the blue shift) indicates that for high-frequency vibrations, such as C–H, O–H stretch, a time step of less than 0.50 fs is necessary.

D. CPU timing comparison

To our knowledge, in all the FMM work previously reported[11,25,27,33,36] the space was subdivided into cubic

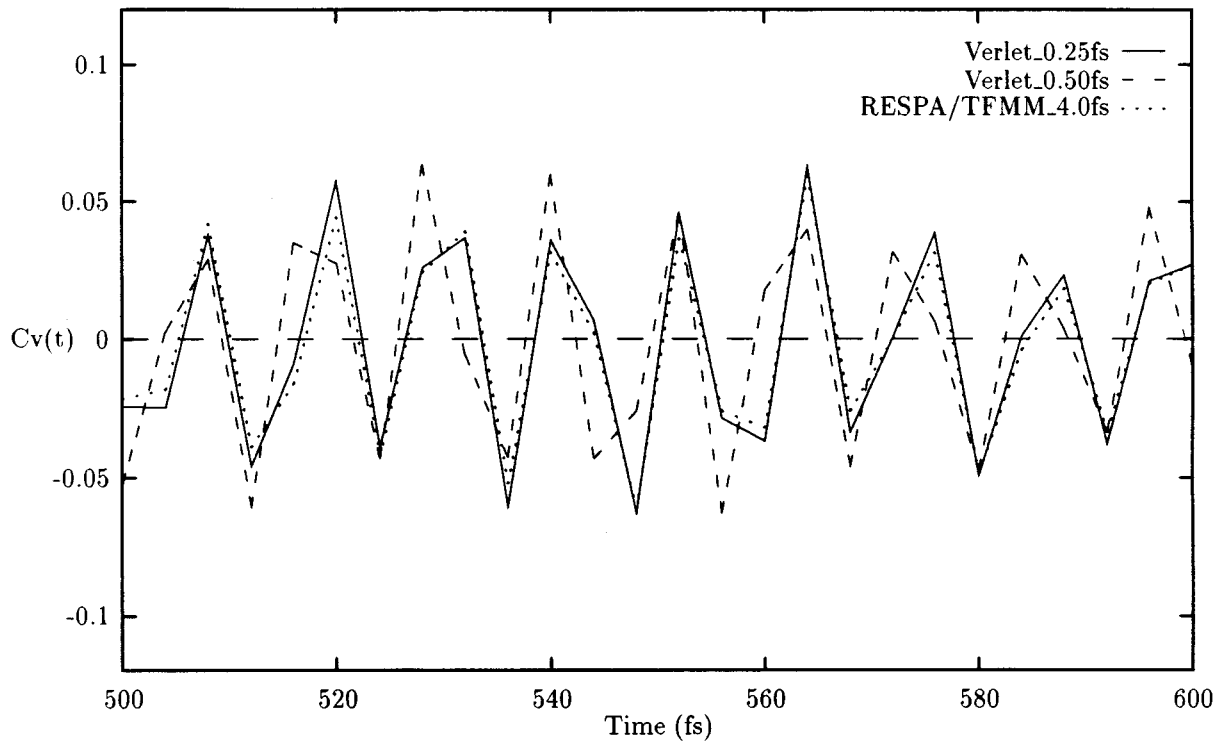

FIG. 5. Comparison of the velocity autocorrelation functions in detail: velocity Verlet with $\Delta t=0.25$ fs (solid line), velocity Verlet with $\Delta t=0.50$ fs (dash line), and r-RESPA/TFMM with overall time step 4.0 fs [combination of (2,2,2,2) with $\delta t=0.25$ fs, dotted line].

boxes. This subdivision of space makes it simpler to construct the FMM algorithm and easier to determine the error bounds (Sec. II A). The use of rectangular boxes instead of cubic boxes will further reduce the number of vacant boxes, and will also reduce the set of pair numbers in the near region which must be evaluated directly. Although using rectangular boxes may lower the accuracy for systems with very high aspect ratios, for most proteins the accuracies are perfectly acceptable (see discussions in Sec. II A).

Figure 8 shows the CPU times for the calculation of the electrostatic potential and the forces in one MD time-step for FMM (cubic box), the top-down FMM with cubic box (TFMM-C) and the top-down FMM with rectangular boxes (TFMM). The optimum parameters (n,p,ws) obtained in Sec. IV A are used here. The top-down FMM with cubic boxes is found to be up to 15% faster than the conventional FMM depending on the inhomogeneity of the proteins. The percentages of the vacant boxes for these proteins are from 14% (1crn) to 37% (1prc) when using cubic boxes, and this percentage may be a normal range for most proteins. However, it must be pointed out that the percentage of the vacant boxes depends on the tree levels used; the higher the tree level, the larger the percentage. Here, we have used optimum tree levels listed in Table I; that is, the average atom number in the smallest box is roughly 2–10. The TFMM, using both top-down recursion and rectangular boxes, however, is found to be 20%–40% faster than the normal FMM, indicating that it may be useful for noncubic inhomogeneous systems, such as proteins.

The CPU saving of r-RESPA vs velocity Verlet springs from the fact that in r-RESPA one can use a time step ~8 times larger than that of velocity Verlet for the far region nonbonded forces (which are usually the most CPU consuming interactions). Figure 9 shows plots of the total CPU times for 1 ps MD runs using velocity Verlet with a time step 0.5 fs and r-RESPA with an overall time step 4.0 fs [combination of (2,2,2,2) in force separation]. It is clear that r-RESPA is about 4–6 times faster than velocity Verlet for various proteins with the same level of accuracy.

Implementation of the TFMM for electrostatic interactions in r-RESPA will further reduce the CPU time for Coulomb interactions compared to the direct evaluation. The total CPU times for r-RESPA/TFMM are also shown in Fig. 9 for comparison. It is found that for a protein larger than ~1000 atoms the TFMM is faster than the direct evaluation, indicating that the crossover point of is about 1000 atoms. For the protein 1 prc, the TFMM further reduces the total CPU time by a factor of 3, i.e., the r-RESPA/TFMM is ~3 times faster than r-RESPA for a protein with 9513 atoms (in fact, the CPU saving for electrostatic forces is about 4–5, the factor 3 is for total CPU saving). This CPU saving is from the elegant use of the multipoles and local field expansions in FMM as discussed in Sec. II A, as well as the top-down recursion and noncubic box separation.

It is of interest to replot the CPU time in Fig. 9 on a log scale, $\log(T_{\mathrm{CPU}})$ vs $\log(N)$, and fit it with a straight line,

$$\log(T_{\mathrm{CPU}})=c_0+c_1\log(N), \tag{46}$$

then we find that the CPU time scales with N as:

Velocity Verlet: $T_{\mathrm{CPU}}\sim N^{1.90}$

r-RESPA: $T_{\mathrm{CPU}}\sim N^{1.79}$

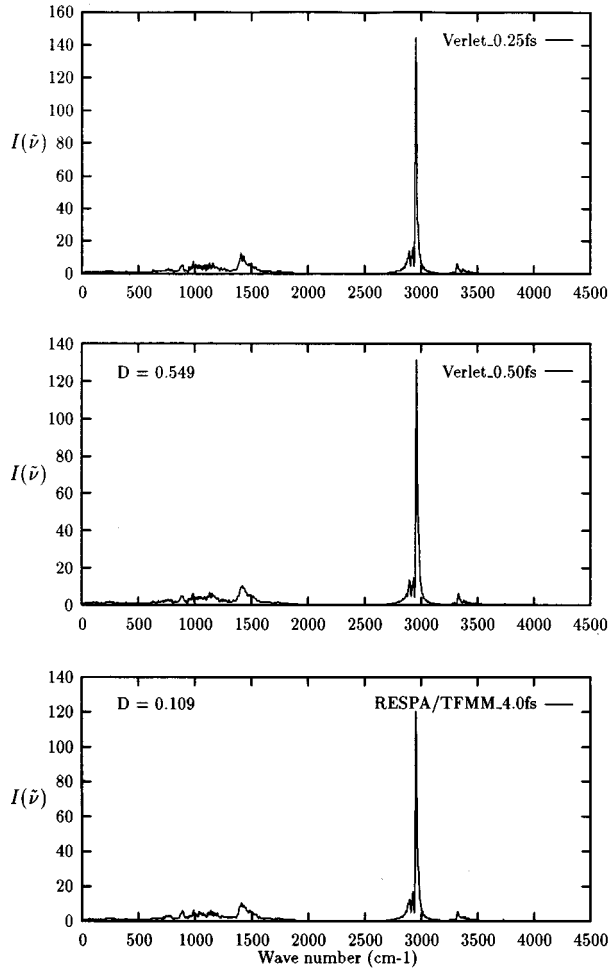

FIG. 6. Spectral intensity $I(\tilde{\nu})$ as a function of wave number for three cases: (a) velocity Verlet with $\Delta t=0.25$ fs, (b) velocity Verlet with $\Delta t=0.50$ fs, and (c) r-RESPA/TFMM with time step 4.0 fs [combination of (2,2,2,2) with $\delta t_1=0.25$ fs]. Intensities are in arbitrary units.

r-RESPA/TFMM: $T_{\mathrm{CPU}}\sim N^{1.32}$.

It is clear that the CPU time scales nearly as N^2 in the velocity Verlet integration method. The r-RESPA method reduces CPU time by a factor of 4–6 for various proteins, and reduces the order from 1.90 to 1.79. After applying TFMM in r-RESPA, the CPU time scales almost linearly with the number of atoms, which is also clear in Fig. 9. Thus, we expect an even larger speedup for larger biosystems. Also, it is no longer necessary to set up pairlists for these long-range pairwise Coulombic interactions, so the memory requirement is also of order of $O(N)$ rather than $O(N^2)$.

To gain a deeper insight into the CPU timings and savings for different forces using different methods, we list detailed CPU timings in Table IV for the largest protein studied in this paper, the photosynthetic reaction center. In order to reduce computational effort, these data are collected over only 0.1 ps MD runs. It is clear that 99% of CPU time is spent on the calculation of the nonbonded forces (vdW and Coloumb forces) when we use standard methods, which indicates that more efficient methods for computing nonbonded forces are highly desirable. The r-RESPA and FMM are exactly designed for this purpose. Compared to the ve-

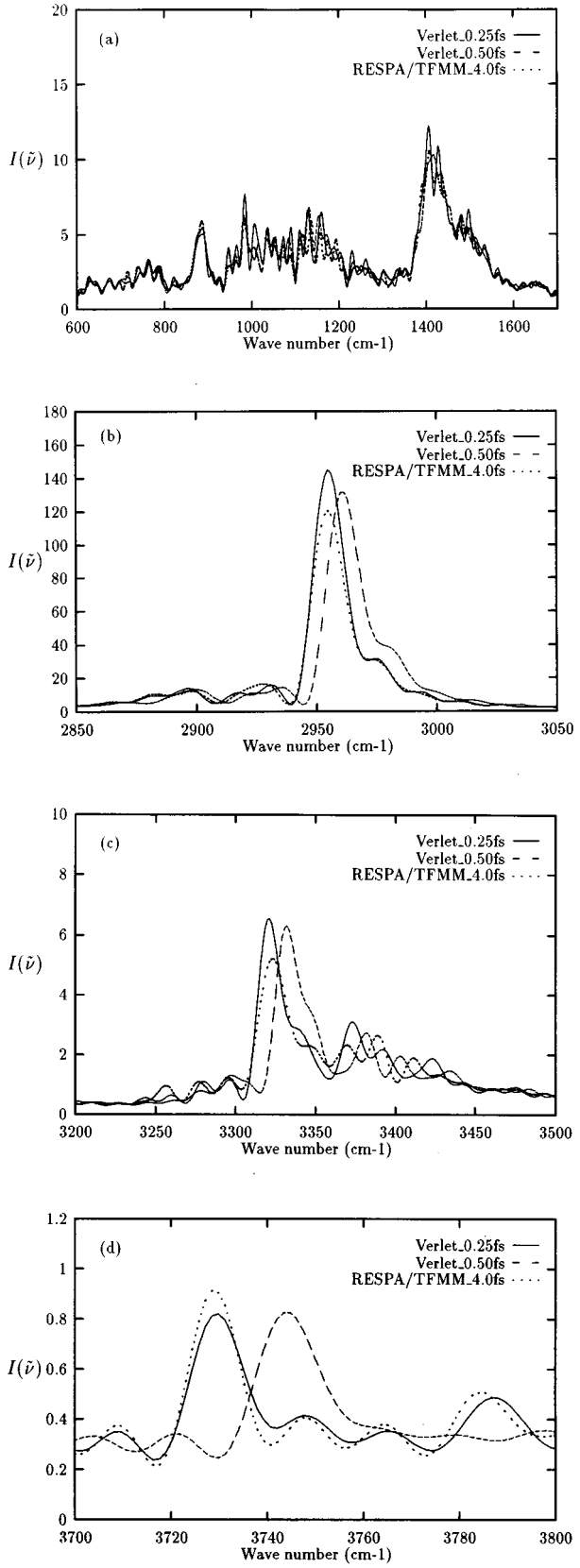

FIG. 7. Comparison of the details of the spectral intensities $I(\tilde{\nu})$ in Fig. 6 in four frequency regions, velocity Verlet with $\Delta t=0.25$ fs (solid line), velocity Verlet with $\Delta t=0.50$ fs (dash line), and r-RESPA/TFMM with overall time-step 4.0 fs [combination of (2,2,2,2) with $\delta t=0.25$ fs, dotted line]. Intensities are in arbitrary units.

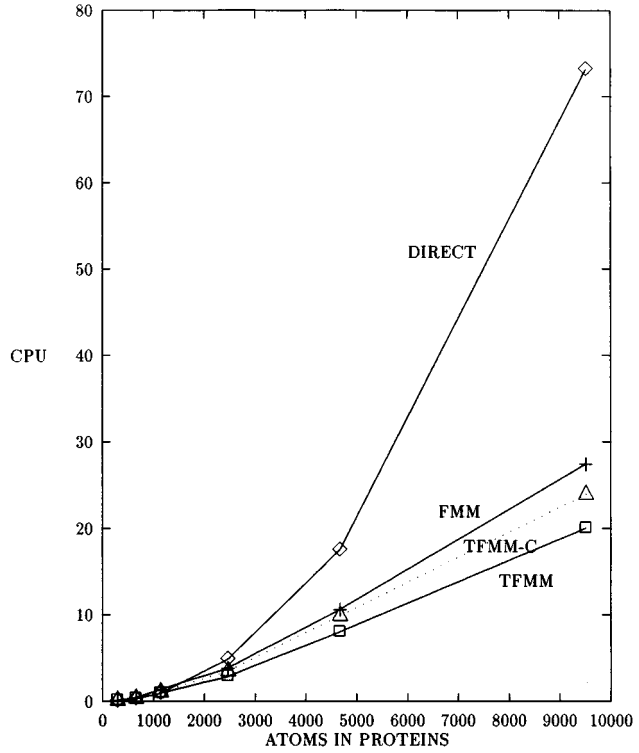

FIG. 8. Comparison of the performances of the convensional FMM (cubic box), top-down FMM with cubic-box subdivision (TFMM-C), and top-down FMM with rectangular-box subdivision (TFMM) in calculating the long range electrostatic interactions for proteins. Parameters $p=4$ and ws$=2$ are used in fast multipole algorithm, and parameter n (total levels in the box tree) is listed in Table I. CPU times for direct evaluation of the Coulomb interactions are also included for comparison. The CPU time (in seconds) is for RISC6000/MODEL 590 computers.

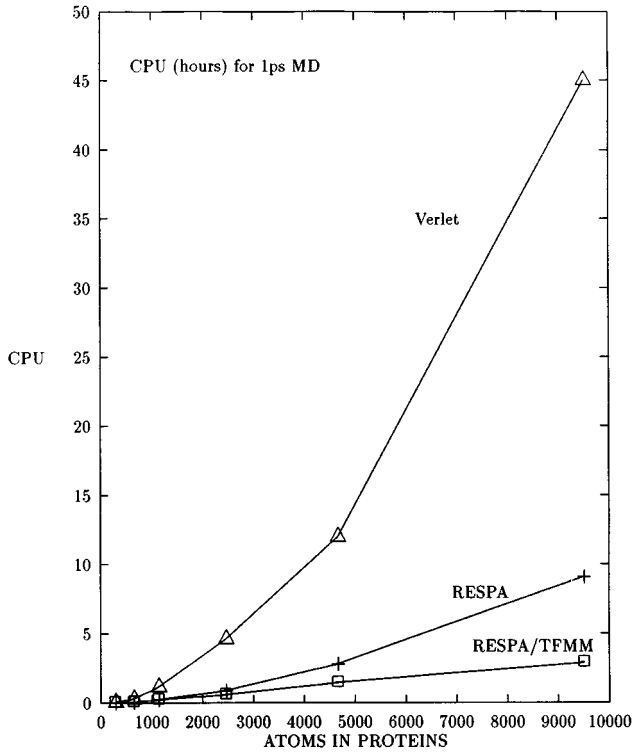

FIG. 9. CPU times (in hours) for 1 ps MD runs for various proteins using three different methods, direct velocity Verlet with a time step 0.5 fs, r-RESPA with direct evaluation of electrostatic forces and an overall time step of 4.0 fs, and r-RESPA/TFMM with an overall time step 4.0 fs [combination of (2,2,2,2) in force breakup]. The energy conservation parameter $\log \Delta E$ for the three methods are comparable. The CPU time (hours) is for RISC6000/MODEL 590 computers.

locity Verlet with time step 0.5 fs, the r-RESPA, with a time step 4.0 fs [combination (2,2,2,2)] and a comparable level of accuracy, lowers the total CPU time from 15566 to 3489 s, by a factor of 4.46. The r-RESPA/TFMM further reduces the total CPU time from 3489 to 1154 s, by a factor of 3.02, and reduces the CPU time for the calculation of electrostatic forces from 2993 to 680 s, by a factor of 4.40. Overall, the r-RESPA/TFMM method lowers the total CPU time by a factor 15 and lowers the CPU time for electrostatic forces by a factor of 20. Since this is for the high accuracy $\log(\Delta E) < -3.0$ const-energy MD simulations, the CPU time saving is quite impressive. If some loss of accuracy can be tolerated, such as in constant-temperature MD simulations where velocities are rescaled artificially, we may obtain an even larger speedup by using $p=3$ (octapole) or $p=2$ (quadrupole). In addition, the best combination of the force separation in r-RESPA is found to be (2,2,2,2) rather than (1,1,1,16), which could be physically reasonable, since the intrinsic separation in time scales for the different forces increases gradually.

As illustrated in Sec. IV C, it is necessary to use very small time steps, such as 0.25 fs, to obtain the reasonable spectral densities for vibrational stretches, such as C–H, O–H stretching. So, if compared to the velocity Verlet with a time step 0.25 fs and direct evaluation of Coulomb forces, a CPU time speed up of ~30 would be expected for r-RESPA/TFMM for a protein with 9513 atoms. Since 0.5 fs is generally used for MD simulations of proteins, our comparison is made to velocity Verlet with time step 0.5 fs. Larger time steps, such as 0.8 fs, 1.0 fs, are also reported for MD simulations of proteins by using SHAKE.[37,38] However, these simulations using SHAKE will affect spectral densities, for example, there will be no C–H peak in spectra if C–H bond length is constrained. Furthermore, SHAKE will affect some other properties, such as time dependent quantities[14,15] and spectral densities associated with the main chain and side chain torsional motion.[38]

Finally, since the CPU saving in r-RESPA/TFMM comes from algorithm improvements, we expect to find comparable improvements in performance on other platforms, such as parallel machines. Parallelization of this efficient MD algorithm is currently under development.

V. CONCLUSION

The new MD algorithm presented here, which uses a new version of the fast multipole method (FMM) and the reversible reference system propagator algorithm (r-RESPA), is a significant improvement over other algorithms in dealing with the two main bottlenecks in simulating biosystems: (a) calculating the full long-range Coulombic interaction and (b) treating the intrinsic differences in timescals for various interactions. The improvements can be summarized as follows:

TABLE IV. Results from 0.1 ps MD runs for the protein Photosynthetic Reaction Center (9513 atoms). Detailed CPU times in various force routines are listed for the three different simulation methods, velocity Verlet, r-RESPA, r-RESPA/TFMM, for comparison. Δt is the overall time step, and $\{n\}$ is the representation of combinations in force separation in r-RESPA, with the smallest time-step $\delta t = 0.25$ fs. The asterisk indicates that the structure blows up when $\Delta t = 3.0$ fs in Verlet method.

Method	$\{n\}$	Δt	$\log(\Delta E)$	R	T_{stret}	T_{bend}	T_{tors}	T_{vdW}	T_{elec}	T_{total}
Verlet	⋯	0.25	−3.6078	0.0192	53.2	34.1	79.6	2916.5	26787.8	29932.4
	⋯	0.50	−3.3346	0.0304	38.5	17.3	40.7	1485.9	13980.2	15566.3
	⋯	1.00	−2.7868	0.0523	19.4	9.1	22.2	736.3	6905.3	7692.5
	⋯	1.50	−2.0046	0.1756	15.1	6.7	17.5	583.0	4553.2	5175.5
	⋯	2.00	−1.4190	0.3656	10.1	4.4	11.9	378.1	3424.6	3828.7
	⋯	3.00	⋯*	⋯						
RESPA	(1,1,1,2)	0.50	−3.5960	0.0179	53.3	34.3	79.5	2920.9	13935.9	17025.3
	(2,2,2,2)	4.00	−3.3734	0.0327	53.0	17.3	40.8	380.0	2993.6	3489.1
RESPA/	(1,1,1,2)	0.50	−3.5170	0.0249	53.2	34.1	79.5	2907.8	4996.9	8069.4
TFMM	(1,1,2,2)	1.00	−3.4027	0.0277	53.2	34.2	79.6	1462.0	2841.3	4474.4
	(1,1,2,3)	1.50	−3.4291	0.0248	54.1	34.8	80.6	1453.5	2330.4	3951.4
	(1,2,2,2)	2.00	−3.3174	0.0303	53.1	34.3	79.6	726.1	1420.8	2314.3
	(1,2,2,3)	3.00	−3.3812	0.0357	52.6	33.8	78.9	719.3	1199.4	2085.0
	(2,2,2,2)	4.00	−3.3334	0.0380	53.3	17.1	39.9	360.8	680.9	1154.8
	(1,2,2,5)	5.00	−2.9038	0.0593	53.1	34.2	79.6	703.9	882.4	1756.6
	(2,2,1,5)	5.00	−2.6762	0.0729	53.0	17.2	39.8	658.5	858.5	1607.2
	(2,2,5,1)	5.00	−2.3520	0.1686	53.3	17.2	39.7	189.6	557.6	856.3
	(2,2,2,3)	6.00	−2.6791	0.0711	53.2	17.3	39.8	371.2	604.1	1085.5

(1) A new version of the fast multipole method (FMM) with top-down recursive generation of the multipoles and rectangular box subdivisions is proposed, which is 20%–40% faster than the standard FMM method for simulations of proteins *in vacuo*. By using the new version of FMM, the requirement for CPU time and memory scales as $O(N)$ instead of $O(N^2)$, and the total CPU time is reduced by a factor of ~ 3 for a protein with 9513 atoms at an efficient accuracy level.
(2) The r-RESPA method can use a time step 8–9 times larger than that of the standard velocity Verlet method (with time step 0.5 fs) with even better infrared spectra for biomolecules, which results a 4–6 times speed up in total CPU time. It is also found that r-RESPA can generate a more stable MD simulation than the frequently used constant long-range force approximation.
(3) By using both the modified FMM and r-RESPA, the computational task is now nearly $O(N)$, which makes possible efficient MD simulations of very large biomolecular systems.
(4) For the photosynthetic reaction center (9513 atoms), the new MD algorithm leads to a 20-fold CPU time speedup for electrostatic interactions, and a 15-fold speedup of the total MD simulation compared to standard methods at the same level of energy conservation, with even better spectra properties. Of course, greater speedups are expected when the algorithm is applied to larger biosystems.

The new method presented in this paper should be very useful for simulations of large proteins surrounded by water molecules, and such applications are under the way.

ACKNOWLEDGMENTS

This work was supported by a grant from the National Institutes of Health (GM43340) and from the NIH Division of Research Resources (SP41RR06892). We wish to thank Dr. Steve Stuart for many useful discussions and careful reading of the manuscript. Some of the calculations were performed on the IBM SP2 at the Maui High Performance Computer Center (MHPCC).

[1] J. A. McCammon and S. C. Harvey, *Dynamics of Proteins and Nucleic Acids* (Cambridge University, Cambridge, 1987).
[2] K. M. Merz, Jr. and S. M. Le Grand, *The Protein Folding Problem and Tertiary Structure Prediction* (Birkhäuser, Boston, 1994).
[3] B. R. Brooks, R. E. Bruccoeri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, J. Comp. Chem. **4**, 187 (1983).
[4] A. Windemuth and K. Schulten, Mol. Simul. **6**, 121 (1991).
[5] L. Verlet, Phys. Rev. **159**, 98 (1967).
[6] W. C. Swope, H. C. Anderson, P. H. Berens, and K. R. Wilson, J. Chem. Phys. **76**, 637 (1982).
[7] R. W. Hockney and J. W. Eastwood, *Comutational Simulation Using Particles* (McGraw-Hill, New York, 1981).
[8] R. W. Hockney, Methods Comput. Phys. **9**, 136 (1970).
[9] A. Appel, SIAM J. Sci. Stat. Comp. **6**, 85, (1985).
[10] J. E. Barnes and P. Hut, Nature **324**, 446, (1986).
[11] L. Greengard, and V. Rokhlin, J. Comp. Phys. **60**, 187 (1985).
[12] J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen, J. Comp. Phys. **23**, 327 (1977).
[13] H. C. Anderson, J. Comp. Phys. **52**, 24 (1983).
[14] W. F. van Gunsteren and M. Karplus, Macromolecules, **15**, 1528 (1982).
[15] S. Toxaerd, J. Chem. Phys., **87**, 6140 (1987).
[16] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford University, Oxford, 1987).

[17] O. Teleman and B. Jönsson, J. Comp. Chem. **7**, 58 (1986).
[18] M. E. Tuckerman and B. J. Berne, J. Chem. Phys. **95**, 8362 (1991).
[19] D. D. Humphreys, R. A. Friesner, and B. J. Berne, J. Phys. Chem. **98**, 6885 (1994).
[20] R. D. Swindoll, and J. M. Haile, J. Comp. Phys. **53**, 289 (1984).
[21] M. E. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **97**, 1990 (1992).
[22] H. F. Trotter, Proc. Am. Math Soc. **10**, 545 (1959).
[23] M. Watanabe, and M. Karplus, J. Chem. Phys. **99**, 8063 (1993).
[24] P. Procacci, and B. J. Berne, J. Chem. Phys. **101**, 2421 (1994).
[25] C. A. White and M. Head-Gorden, J. Chem. Phys. **101**, 6593 (1994).
[26] Here, we used M_{lm} for multipole moments and L_{lm} for local field expansions (Taylor coefficients) for clarity, which are different from White and Head-Gordon's notation. They used O_{lm} and ω_{lm} for multipole moments, and M_{lm} and μ_{lm} for Taylor coefficients.
[27] H.-Q. Ding, N. Karasawa, and W. A. Goddard III, J. Chem. Phys. **97**, 4309 (1992).
[28] M. E. Tuckerman, G. J. Martyna, and B. J. Berne, J. Chem. Phys. **94**, 6811 (1991).
[29] M. E. Tuckerman, and B. J. Berne, J. Chem. Phys. **98**, 7301 (1993).
[30] F. Mohamadi, N. G. J. Richards, W. C. Guida, R. Liskamp, M. Lipton, C. Caufield, G. Chang, T. Hendrickson, and W. C. Still, J. Comp. Chem. **11**, 440 (1990).
[31] S. J. Weiner, P. A. Kollman, D. T. NGuyen, and D. A. Case, J. Comp. Chem. **7**, 230 (1986).
[32] W. L. Jorgensen and J. Tirado-Rives, J. Am. Chem. Soc. **110**, 1657 (1988).
[33] J. A. Board, Jr., J. W. Causey, J. F. Leathrum Jr., A. Windermuth, and K. Schulten, Chem. Phys. Lett. **189**, 89 (1992).
[34] W. F. van Gunsteren and H. J. C. Berendsen, Mol. Phys. **34**, 1311 (1977).
[35] J. Skilling and R. K. Bryan, Mon. Not. R. Astron. Soc. **211**, 111 (1984).
[36] J. Shimada, H. Kaneko, and T. Takada, J. Comp. Chem. **15**, 28 (1994).
[37] M. Marchi, J. N. Gehlen, D. Chandler, and M. Newton, J. Am. Chem. Soc. **115**, 4178 (1993).
[38] M. Watanabe and M. Karplus, J. Phys. Chem. **99**, 5680 (1995).