A Multiple-Time-Step Molecular Dynamics Algorithm for Macromolecules

Darryl D. Humphreys,† Richard A. Friesner, and Bruce J. Berne*

Department of Chemistry and the Center for Biomolecular Simulation, Columbia University, New York, New York 10027

Received: February 22, 1994®

We present a computationally efficient molecular dynamics algorithm designed to take advantage of the inherent separation of time scales in biomolecular systems. The algorithm is essentially a generalization of the previously introduced reversible reference system propagation algorithm (r-RESPA) which employs a Trotter factorization of the Liouville propagator to generate numerical integration schemes for molecular dynamics applications. The method is compared with the velocity Verlet integration algorithm in a molecular dynamics (MD) simulation of the protein crambin *in vacuo*. The multiple-time-step algorithm is shown to be able to take a much larger time step for a comparable level of accuracy than that of the standard method, leading to a 4–5-fold reduction in the CPU time required to calculate the nonbonded forces.

Introduction

Computer simulations have become a valuable tool used in the theoretical investigation of biomolecular systems. Unfortunately, these simulations are often computationally demanding tasks owing to the large number of particles, as well as the complex nature of their associated interactions. In the method of molecular dynamics (MD), this problem is particularly acute due to the fact that limitations in accuracy are imposed by the fastest time scale considered, such as the vibrational periods of bond stretch and bend degrees of freedom, even though one may be primarily interested in events that occur over a much longer time scale. In standard numerical integration methods, such as of the Störmer–Verlet variety,[1,2] one is generally required to use a time step on the order of a femtosecond in order to maintain an acceptable level of accuracy in the integration of the equations of motion. Since the CPU time to calculate the interactions scales as N^2, where N is the number of particles, and typical simulations are often for hundreds of picoseconds, the required computational effort can indeed be enormous. This often results in the need to make approximations, such as truncating the nonbonded interactions at some cutoff distance, which may in turn affect the quality of the calculation.[3]

A variety of techniques have been introduced in order to increase the time step in molecular dynamics simulations in an attempt to surmount these difficulties. One common approach is to constrain bond lengths using either the SHAKE or RATTLE algorithms.[4,5] Although application of these methods allows for a modest increase in the time step, time-dependent quantities may be affected.[6,7] Additionally, the constraint methods have not been shown to work well for bond angle degrees of freedom when applied to the case of macromolecules.[6]

Another approach to increase the time step in MD simulations is that of the multiple-time-step (MTS) methods.[8] These methods are based upon integration schemes that allow for time steps of differing lengths according to how rapidly a given type of interaction is evolving in time. Teleman and Jönsson[9] introduced an algorithm whereby the slower degrees of freedom are held constant for a number of smaller time steps which are used to integrate the faster degrees of freedom. This method has, however, been shown to lead to the accumulation of numerical error in calculated quantities.[10] Alternatively, Swindoll and Haile[11] 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,[11] it is not yet evident as to whether it would be computationally advantageous in the case of macromolecules.

In this paper we present a molecular dynamics algorithm 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),[12] which employs a Trotter factorization[13] 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, such as those containing disparate masses, long- and short-range interactions, and stiff and soft forces.[12] Additionally, this method has recently been applied to the case of small organic molecules by Watanabe and Karplus,[14] as well as to that of a fullerene crystal.[15]

In the case of macromolecules, although the potential is substantially more complex than that of a simple Lennard–Jones liquid, our approach is very much in the same spirit. Here, also, we divide the forces of the system into effective time scales according to how rapidly a given force varies with time. By factorizing the Liouville propagator, we are then able to generate numerical integration algorithms whereby different time steps may be chosen for each effective time scale. This avoids having to take an unnecessarily small time step for interactions that are slowly varying while concomitantly maintaining accuracy for the rapidly varying degrees of freedom.

In this paper we first present a brief description of the formalism, followed by an application of the method to a molecular dynamics simulation of the protein crambin *in vacuo*. This was achieved by incorporating the algorithm into a modified version of the MACROMODEL[16] molecular modeling package. It is shown that the r-RESPA method is significantly more efficient than that of the standard velocity Verlet integrator, leading to an approximately 4–5-fold reduction in the (*nonbonded*) CPU time for a system containing 655 particles for a comparable level of accuracy.

Method

Trotter Expansion of the Liouville Propagator. Our starting point for the numerical integration of Hamilton's equations is

† In partial fulfillment of the Ph.D. in the Department of Chemistry, Columbia University.

® Abstract published in *Advance ACS Abstracts*, June 1, 1994.

0022-3654/94/2098-6885$04.50/0

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

$$iL = [..., H] = \sum_i^N \left[\dot{q}_i \frac{\partial}{\partial q_i} + F_i(q) \frac{\partial}{\partial p_i} \right] \quad (1)$$

where [...,...] is the Poisson bracket. The state of the system at a time t is then given by

$$\Gamma(t) = U(t)\, \Gamma(0) \quad (2)$$

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

$$U(t) = e^{itL} \quad (3)$$

and where

$$\Gamma(t) = \{q(t),p(t)\} = (q_1(t),...,q_N(t),p_1(t),...,p_N(t)) \quad (4)$$

$\{q,p\}$ are the coordinates and conjugate momenta, and H is the Hamiltonian of the system. We may then choose to decompose the Liouville operator into two parts, such that

$$L = L_1 + L_2 \quad (5)$$

This allows us to apply the Trotter theorem,[13] giving

$$e^{it(L_1+L_2)} = [e^{i(t/P)(L_1+L_2)}]^P$$

$$= [e^{i(\Delta t/2)L_2} e^{i\Delta t L_1} e^{i(\Delta t/2)L_2}]^P + O(\Delta t^3) \quad (6)$$

where $\Delta t \equiv t/P$. One may then define a discrete time propagator as

$$G(\Delta t) = U_2(\Delta t/2)\, U_1(\Delta t)\, U_2(\Delta t/2)$$

$$= e^{i(\Delta t/2)L_2} e^{i\Delta t L_1} e^{i(\Delta t/2)L_2} \quad (7)$$

Having done this, we are now able to take advantage of the form of the discrete time propagator, using it to develop multiple-time-step integration algorithms. Although there exist numerous ways of breaking up the true system into a "reference system" propagator and "correction" propagators, a convenient choice can be made by noting that the inner propagator in eq 7 can be further decomposed as follows

$$e^{i\Delta t L_1} = (e^{i\delta\tau L_1})^n \quad (8)$$

where $n\delta\tau = \Delta t$. 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, or reference, propagator may be taken to be associated with the rapidly varying interactions and is propagated according to a smaller time step than that of the outer correction propagators which can be used to evolve the less rapidly varying degrees of freedom. The formal solution for the discretized equations of motion is then given by

$$\Gamma(\Delta t) = U_2(\Delta t/2)\, U_1(\Delta t)\, U_2(\Delta t/2)\, \Gamma(0) + O(\Delta t^3) \quad (9)$$

$$= e^{i(n\delta\tau/2)L_2} [e^{i\delta\tau L_1}]^n e^{i(n\delta\tau/2)L_2}\, \Gamma(0) + O(\Delta t^3) \quad (10)$$

This approach may also be generalized to cases with more than two effective time scales. Suppose for example we choose to break the Liouville operator into a sum of three terms, that is

$$L = L_1 + L_2 + L_3 \quad (11)$$

Let us then define

$$L_A \equiv L_1 + L_2 \quad (12)$$

We may apply the following Trotter decomposition to give a discretized propagator of the form

$$G(\Delta t) \equiv e^{i(\Delta t/2)L_3} e^{i\Delta t L_A} e^{i(\Delta t/2)L_3} \quad (13)$$

The middle propagator may then be factorized as

$$e^{i\Delta t L_A} = [e^{i\delta\tau_2 L_A}]^{n_2} \quad (14)$$

where $\delta\tau_2 \equiv \Delta t/n_2$. Another Trotter decomposition gives

$$e^{i\delta\tau_2 L_A} = e^{i\delta\tau_2 (L_1+L_2)} \quad (15)$$

$$= e^{i(\delta\tau_2/2)L_2} e^{i\delta\tau_2 L_1} e^{i(\delta\tau_2/2)L_2} + O(\tau_2^3) \quad (16)$$

Similarly,

$$e^{i\delta\tau_2 L_1} = [e^{i\delta\tau_1 L_1}]^{n_1} \quad (17)$$

where $\delta\tau_1 \equiv \delta\tau_2/n_1$. The entire discretized propagator for a system with three time scales is then given by

$$G(\Delta t) \equiv e^{i(n_1 n_2 \delta\tau_1/2)L_3} \times [e^{i(n_1\delta\tau_1/2)L_2} [e^{i\delta\tau_1 L_1}]^{n_1} e^{i(n_1\delta\tau_1/2)L_2}]^{n_2} e^{i(n_1 n_2 \delta\tau_1/2)L_3} \quad (18)$$

This would correspond to a situation where the inner reference propagator is evaluated every small time step $\delta\tau_1$, whereas the middle and outer correction propagators are evaluated every n_1 and $n_1 n_2$ small time steps, respectively. The generalization to N time scales follows in an analogous manner.

Multiple-Time-Step Molecular Dynamics Algorithm for Macromolecules. As a first attempt to apply the techniques discussed above to the simulation of a macromolecule, we take the Liouville operator for a macromolecule *in vacuo* containing N atoms to be

$$iL = \sum_i^{3N} \left[\dot{x}_i \frac{\partial}{\partial x_i} + F_i(x) \frac{\partial}{\partial p_i} \right] \quad (19)$$

such that

$$F(x) = F_{\text{stretch}}(x) + F_{\text{bend}}(x) + F_{\text{dihedral}}(x) + F_{\text{hydrogen bonds}}(x) + F_{\text{van der Waals}}(x) + F_{\text{electrostatic}}(x) \quad (20)$$

where the functional form of the associated potential is given in refs 16 and 17. In order to simplify the notation, let us now define the bonded/torsion and nonbonded contributions of $F(x)$ to be

$$F_b(x) \equiv F_{\text{stretch}}(x) + F_{\text{bend}}(x) + F_{\text{dihedral}}(x) + F_{\text{hydrogen bonds}}(x) \quad (21)$$

$$F_{nb}(x) \equiv F_{\text{van der Waals}}(x) + F_{\text{electrostatic}}(x) \quad (22)$$

Since the majority of the CPU time in a molecular dynamics simulation of a macromolecule is spent calculating the nonbonded forces, we choose to divide the nonbonded forces into both a long- and short-range contribution, that is

$$F_{nb}(x) = F_{nb}^{\text{short}}(x) + F_{nb}^{\text{long}}(x) \quad (23)$$

This may be done through the use of a switching function S such that

$$F_{nb}^{\text{short}}(x) \equiv S(r) \cdot F_{nb}(x) \quad (24)$$

$$F_{nb}^{long}(x) \equiv [1 - S(r)]\cdot F_{nb}(x) \quad (25)$$

The switching function itself is given by[12,18]

$$S(r) = \begin{cases} 1 & r < r_c - \Delta r \\ 1 + R^2(2R - 3) & r_c - \Delta r \le r \le r_c \\ 0 & r_c < r \end{cases} \quad (26)$$

where $R \equiv [r - (r_c - \Delta r)]/\Delta r$, r is the interatomic distance, r_c is the short-range cutoff, and Δr is the healing length. Having done this, we may express the Liouville operator as a sum of four terms

$$L = L_1 + L_2 + L_3 + L_4 \quad (27)$$

where

$$iL_1 \equiv \dot{x}\frac{\partial}{\partial x} + F_1(x)\frac{\partial}{\partial p} \quad (28)$$

$$iL_2 \equiv F_2(x)\frac{\partial}{\partial p} \quad (29)$$

$$iL_3 \equiv F_3(x)\frac{\partial}{\partial p} \quad (30)$$

$$iL_4 \equiv F_4(x)\frac{\partial}{\partial p} \quad (31)$$

and

$$F_1(x) \equiv F_{stretch}(x) \quad (32)$$

$$F_2(x) \equiv F_{bend}(x) + F_{dihedral}(x) + F_{hydrogen\ bonds}(x) \quad (33)$$

$$F_3(x) \equiv F_{nb}^{short}(x) \quad (34)$$

$$F_4(x) \equiv F_{nb}^{long}(x) \quad (35)$$

In molecular dynamics one typically seeks a numerical solution $\{x(t_n),p(t_n)\}$ for the atomic positions and momenta in Cartesian coordinates as a function of the discretized time $t_n = n\Delta t$, given that the system was in an initial state $\{x(0),p(0)\}$. Formally, the solution is given by

$$\{x(t_n),p(t_n)\} = U(n\Delta t)\{x(0),p(0)\} = e^{in\Delta tL}\{x(0),p(0)\} \quad (36)$$

For the present case, we consider the approximation

$$\{x(t_n),p(t_n)\} \approx G^n(\Delta t)\{x(0),p(0)\} \quad (37)$$

where $G(\Delta t)$ is given by

$$G(\Delta t) \equiv e^{i(n_1n_2n_3\delta\tau_1/2)L_4}[e^{i(n_1n_2\delta\tau_1/2)L_3}[e^{i(n_1\delta\tau_1/2)L_2} \times [e^{i\delta\tau_1L_1}]^{n_1}e^{i(n_1\delta\tau_1/2)L_2}]^{n_2}e^{i(n_1n_2\delta\tau_1/2)L_3}]^{n_3}e^{i(n_1n_2n_3\delta\tau_1/2)L_4} \quad (38)$$

and L_1, L_2, L_3, and L_4 are given by eqs 28–31. In order to implement the algorithm numerically, we make use of the property

$$e^{y\partial/\partial x}f(x) = f(x + y) \quad (39)$$

where y is taken to be independent of x.[12] Given that the outer correction propagators $e^{i(\delta\tau_\alpha/2)L_\alpha}$, α = 2–4, are of the form $e^{(\delta\tau_\alpha/2)F_\alpha(x)\partial/\partial p}$, where

$$\delta\tau_2 \equiv n_1\delta\tau_1 \quad (40)$$

$$\delta\tau_3 \equiv n_1n_2\delta\tau_1$$

$$\delta\tau_4 \equiv n_1n_2n_3\delta\tau_1$$

and $\Delta t = \delta\tau_4$, we have, upon acting to the right on an arbitrary state $\{x,p\}$,

$$e^{(\delta\tau_\alpha/2)F_\alpha(x)\partial/\partial p}\{x,p\} = \{x,p + (\delta\tau_\alpha/2)F_\alpha(x)\} \quad (41)$$

Similarly, considering now the inner reference propagator in eq 38, that is

$$e^{i\delta\tau_1L_1} = e^{\delta\tau_1(\dot{x}\partial/\partial x + F_1(x)\partial/\partial p)} \quad (42)$$

it has been shown that the following Trotter factorization

$$e^{\delta\tau_1(\dot{x}\partial/\partial x + F_1(x)\partial/\partial p)} = e^{(\delta\tau_1/2)F_1(x)\partial/\partial p}\, e^{\delta\tau_1\dot{x}\partial/\partial x}\, e^{(\delta\tau_1/2)F_1(x)\partial/\partial p} + O(\delta\tau_1^3) \quad (43)$$

is equivalent to the velocity Verlet algorithm.[12] This can be demonstrated by operating with the reference propagator upon an initial state $\{x(0),p(0)\}$,

$$e^{(\delta\tau_1/2)F_1(x)\partial/\partial p}\, e^{\delta\tau_1\dot{x}\partial/\partial x}\, e^{(\delta\tau_1/2)F_1(x)\partial/\partial p}\{x(0),p(0)\} \quad (44)$$

$$= e^{(\delta\tau_1/2)F_1(x)\partial/\partial p}\, e^{\delta\tau_1\dot{x}\partial/\partial x}\{x(0),p(\delta\tau_1/2)\}$$

$$= e^{(\delta\tau_1/2)F_1(x)\partial/\partial p}\{x(\delta\tau_1),p(\delta\tau_1/2)\}$$

$$= \{x(\delta\tau_1),p(\delta\tau_1)\}$$

where

$$p(\delta\tau_1/2) = p(0) + (\delta\tau_1/2)F(x(0)) \quad (45)$$

$$x(\delta\tau_1) = x(0) + (\delta\tau_1/m)[p(0) + (\delta\tau_1/2)F(x(0))]$$

$$= x(0) + \delta\tau_1\dot{x}(0) + (\delta\tau_1^2/2m)F(x(0)) \quad (46)$$

m is the atomic mass, and

$$p(\delta\tau_1) = p(\delta\tau_1/2) + (\delta\tau_1/2)F(x(\delta\tau_1))$$

$$= p(0) + (\delta\tau_1/2)[F(x(0)) + F(x(\delta\tau_1))] \quad (47)$$

Thus, the evolution of the system is determined numerically by acting with the propagator in eq 38 to the right on the initial state $\{x(0),p(0)\}$, using eq 41 and eqs 43–47. A schematic FORTRAN implementation of the resulting algorithm can be found in the Appendix.

In the following section we apply the method discussed above, using the approximate propagator in eq 38 and the reference propagator of eq 43, in the MD simulation of the small protein crambin *in vacuo*. The results are compared with that of the standard velocity Verlet integrator.

Molecular Dynamics of Crambin *in Vacuo*

As a preliminary test of the method, we compare the multiple-time-step (r-RESPA) algorithm described in the previous section to the standard velocity Verlet algorithm for the simulation of crambin (see Figure 1) *in vacuo*. This was done by implementing the algorithm in a modified version of the MACROMODEL[16] molecular modeling package. We choose a completely flexible all-atom model (modified version of the AMBER all-atom parameters)[17] using infinite van der Waals and electrostatic cutoffs with a constant dielectric, ϵ = 1. The initial state of the system

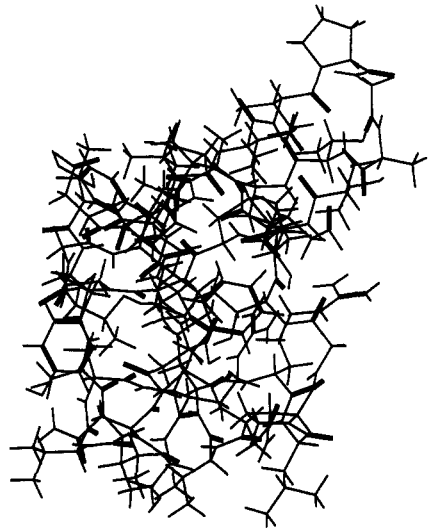

Figure 1. X-ray structure of crambin (PDB code 1CRN) with addition of explicit hydrogens.

was obtained by first performing 4000 iterations of conjugate gradient minimization upon the X-ray structure. The system was initially heated to 300 K by sampling velocities from a Maxwell–Boltzman distribution. This was then followed by 20 ps of molecular dynamics using the velocity Verlet integration algorithm with a time step of 0.5 fs in order to equilibrate. During the equilibration the velocities were resampled from a Maxwell–Boltzman distribution every 100 steps if the average temperature over the previous 0.5 ps deviated from 300 K by more than ±5 K. All simulations were performed on an IBM RISC system 6000/Model 580 computer. Each production run consisted of a total of 5 ps of simulation time.

In order to compare the efficiency of the two methods, it is important to compare their respective performances in achieving the same level of accuracy. In order to do this, we define an energy conservation parameter ΔE, such that

$$\Delta E \equiv \frac{1}{N}\sum_{i}^{N}\left|\frac{E_{\text{initial}} - E_i}{E_{\text{initial}}}\right| \tag{48}$$

where E_i is the total energy at step i, E_{initial} is the initial energy, and N is the total number of time steps. This quantity has been shown to be a reasonable measure of accuracy in previous simulations.[12,14] Watanabe et al.[14] found that a value of $\Delta E \leq 0.001$–0.003 is a sufficient condition for the numerical accuracy of the integration. Alternatively, 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,[19]

$$R \equiv \frac{\Delta E_{\text{rms}}}{\Delta KE_{\text{rms}}} \tag{49}$$

A value of $R \leq 0.01$ has been used as a criterion for stability in previous simulations.[14]

As a first test of the algorithm, we compare the energy conservation of the multiple-time-step method, using the reference propagator given in eq 43 with that of the velocity Verlet integrator, by using an "internal/external" force decomposition. Physically this corresponds to separating the interactions into two time scales on the basis of whether the force is of an internal (taken to include hydrogen bonds) or of a nonbonded nature. Additionally, we consider the case here where the bond stretch forces are separated from the other "internal" degrees of freedom. We shall refer to this as the "stiff/soft" force separation. The results are shown in Table 1. Here it is shown that r-RESPA can take a time step approximately 3 times as large as that of velocity Verlet $\Delta t = 0.5$ fs for comparable energy conservation, i.e. $\delta\tau_1 = 0.25$ fs, $n_1 = 2$, $n_2 = 3$, $n_3 = 1$.

TABLE 1: Stiff/Soft and Internal/External Force Decomposition: Comparison of Energy Conservation and Associated CPU Times for Velocity Verlet ($n_1 = n_2 = n_3 = 1$) and r-RESPA[a]

Δt	$\delta\tau_1$	n_1	n_2	n_3	$\log(\Delta E)$	R	T_{NB}	T_{F}
0.25	0.25	1	1	1	−3.7073	0.0022	10085.9	10703.1
0.50	0.50	1	1	1	−3.0123	0.0087	5073.4	5383.6
1.00	1.00	1	1	1	−2.2388	0.0398	2579.6	2738.1
2.00	2.00	1	1	1	−1.0475	0.1981	1326.7	1407.1
0.50	0.25	1	2	1	−3.6289	0.0026	5041.7	5670.0
1.00	0.25	1	4	1	−3.3783	0.0046	2463.9	3076.9
1.50	0.25	1	6	1	−3.0820	0.0092	1662.6	2283.7
2.00	0.25	1	8	1	−2.7850	0.0163	1252.2	1874.3
2.50	0.25	1	10	1	−2.4186	0.0315	990.2	1601.5
1.00	0.25	2	2	1	−3.3313	0.0052	2582.8	2915.2
1.50	0.25	2	3	1	−3.0732	0.0088	1726.8	2059.4
2.00	0.25	2	4	1	−2.7804	0.0175	1294.6	1626.7
3.00	0.25	2	6	1	−2.1184	0.0496	862.1	1194.8
4.00	0.25	2	8	1	−0.3355	1.8764	647.3	978.6
1.00	0.50	1	2	1	−2.9855	0.0095	2656.5	2981.0
1.50	0.50	1	3	1	−2.8881	0.0130	1759.9	2080.3
2.00	0.50	1	4	1	−2.7135	0.0180	1323.1	1644.5
2.50	0.50	1	5	1	−2.3696	0.0314	1021.1	1332.2
3.00	0.50	1	6	1	−1.5932	0.1491	843.3	1152.3
3.50	0.50	1	7	1	−1.3592	0.2553	723.7	1031.6

[a] Here $\log(\Delta E)$ and R are given by eq 48 and eq 49. T_{NB} is the CPU time spent in the nonbonded force routine, and T_{F} is the total CPU time (s) spent in all force routines. Δt and $\delta\tau_1$ are in femtoseconds.

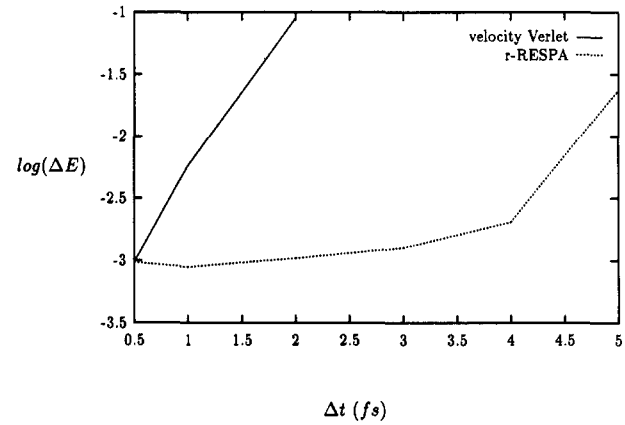

Figure 2. Short/long-range nonbonded force decomposition: dependence of energy conservation as defined by eq 48 vs time step for velocity Verlet (solid line) and r-RESPA using the propagator in eq 51 (dotted line).

Next we investigate the short- and long-range nonbonded force separations. Since decomposing the nonbonded forces into both long- and short-range contributions introduces some additional computational effort through the calculation of the switching function, the nonbonded pairs were separated into three "regions". This was done according to whether the value of the switching function of a given nonbonded pair in eq 26 was 0, 1, or of some intermediate value. Additionally, the intermediate pair list, i.e. the list of pairs whose switching function value is between 0 and 1, was modified to include a "buffer region" so as to take into account those nonbonded pairs which may change regions between updatings of the pair list. That is, the intermediate pair list is defined to be all nonbonded pairs at a given time such that

$$r_c - \Delta r - \Delta r_{\text{buffer}} \leq r_{ij} \leq r_c + \Delta r_{\text{buffer}} \tag{50}$$

where r_c is the short-range force cutoff, Δr is the switching function healing length, r_{ij} is the pair separation, and Δr_{buffer} is the buffer healing length, taken to be 0.5 Å. The pair list itself was updated between every 10 to 25 fs.

An example of the results for the long/short force decomposition is shown in Figure 2. Here $\delta\tau_1 = 0.5$ fs, $n_1 = 1$, $n_2 = 1$, and the

TABLE 2: Short/Long-Range Nonbonded Force Decomposition: Comparison of Energy Conservation and Associated CPU Times for Velocity Verlet ($n_1 = n_2 = n_3 = 1$) and r-RESPA[a]

Δt	$\delta\tau_1$	n_1	n_2	n_3	r_c	Δr	$\log(\Delta E)$	R	T_{PL}	T_{NB}	T_F
0.25	0.25	1	1	1			-3.7073	0.0022		10085.9	10703.1
0.50	0.50	1	1	1			-3.0123	0.0087		5073.4	5383.6
1.00	1.00	1	1	1			-2.2388	0.0398		2579.6	2738.1
2.00	2.00	1	1	1			-1.0475	0.1981		1326.7	1407.1
1.00	0.50	1	1	2	4.0	2.0	-3.0645	0.0086	46.9	2670.1	2980.5
2.00	0.50	1	1	4	4.0	2.0	-3.0158	0.0115	46.6	1460.4	1768.6
3.00	0.50	1	1	6	4.0	2.0	-2.8114	0.0177	46.6	1065.5	1373.6
4.00	0.50	1	1	8	4.0	2.0	-2.5564	0.0326	46.7	867.1	1175.0
5.00	0.50	1	1	10	4.0	2.0	-0.5789	1.1001	44.9	747.4	1055.1
1.00	0.50	1	1	2	6.0	2.0	-3.0605	0.0086	46.7	3023.8	3334.9
2.00	0.50	1	1	4	6.0	2.0	-2.9756	0.0102	46.0	1871.1	2178.0
3.00	0.50	1	1	6	6.0	2.0	-2.8706	0.0107	46.0	1321.6	1811.9
4.00	0.50	1	1	8	6.0	2.0	-2.6622	0.01389	45.8	1321.6	1628.8
5.00	0.50	1	1	10	6.0	2.0	-1.6791	0.1067	44.1	1211.8	1518.4
1.00	0.50	1	1	2	8.0	2.0	-3.0278	0.0094	43.6	3338.2	3645.3
2.00	0.50	1	1	4	8.0	2.0	-3.0155	0.0090	43.8	2373.1	2681.3
3.00	0.50	1	1	6	8.0	2.0	-2.9851	0.0095	46.0	2150.5	2473.6
4.00	0.50	1	1	8	8.0	2.0	-2.7526	0.0121	43.7	1883.8	2191.1
5.00	0.50	1	1	10	8.0	2.0	-1.8296	0.0668	42.9	1812.3	2123.1
1.00	0.50	1	1	2	10.0	2.0	-3.0272	0.0089	41.4	3743.7	4051.8
2.00	0.50	1	1	4	10.0	2.0	-3.0075	0.0096	40.9	2937.5	3245.6
3.00	0.50	1	1	6	10.0	2.0	-2.9984	0.0096	43.0	2781.0	3103.4
4.00	0.50	1	1	8	10.0	2.0	-3.0006	0.0107	41.3	2544.6	2853.6
5.00	0.50	1	1	10	10.0	2.0	-2.1594	0.0412	39.3	2448.1	2755.2
1.00	0.50	1	1	2	12.0	2.0	-3.0397	0.0092	37.7	4111.4	4418.3
2.00	0.50	1	1	4	12.0	2.0	-3.0542	0.0089	37.7	3487.4	3794.0
3.00	0.50	1	1	6	12.0	2.0	-3.0138	0.0094	37.8	3291.2	3598.8
4.00	0.50	1	1	8	12.0	2.0	-2.9935	0.0091	37.9	3170.5	3477.9
5.00	0.50	1	1	10	12.0	2.0	-2.5333	0.0326	36.4	3089.5	3397.0

[a] Here $\log(\Delta E)$ and R are given by eq 48 and eq 49. T_{NB} is the CPU time spent in the nonbonded force routine, T_{PL} is the CPU time spent calculating the r-RESPA neighbor lists, and T_F is the total CPU time (s) spent in all force routines. r_c and Δr are the short-range cutoff and switching function healing length, respectively, in angstroms. Δt and $\delta\tau_1$ are in femtoseconds.

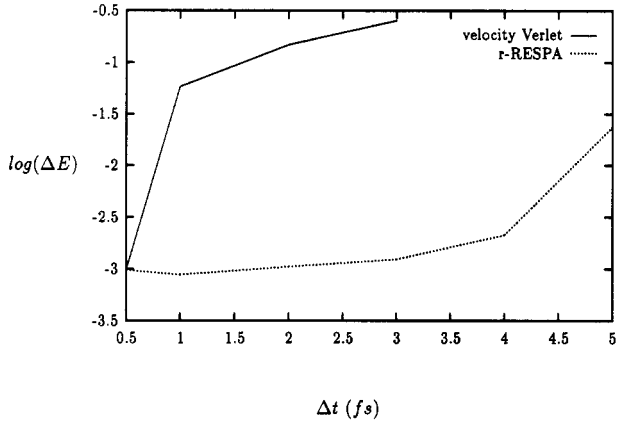

Figure 3. Dependence of energy conservation as defined by eq 48 vs long-range time step for velocity Verlet using the constant long-range force approximation (solid line) and r-RESPA using the propagator in eq 51 (dotted line).

energy conservation of the r-RESPA method is now a function of n_3, the propagator given by

$$G(\Delta t) \equiv e^{i(n_3\delta\tau_1/2)L_4}[e^{i\delta\tau_1(L_1+L_2+L_3)}]^{n_3}e^{i(n_3\delta\tau_1/2)L_4} \quad (51)$$

Here we have used a short-range cutoff $r_c = 6.0$ Å and a healing length Δr of 2.0 Å. The results indicate that for these parameters the r-RESPA method is able to take a time step $\Delta t = n_3\delta\tau_1$ nearly 6 times larger than that of the velocity Verlet for a similar level of accuracy. Table 2 summarizes the CPU times and energy conservation results for the short/long-range nonbonded force decomposition as a function of cutoff radius and time step.

It is also interesting to compare r-RESPA 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 a velocity Verlet is used. In order to make the comparison, we choose to employ the same short/long-range force

TABLE 3: Combined Stiff/Soft, Internal/External, and Short/Long-Range Nonbonded Force Decomposition. Comparison of Energy Conservation and Associated CPU Times Spent in the Various Force Routines for Velocity Verlet ($n_1 = n_2 = n_3 = 1$) and r-RESPA Using the Propagator Given by eq 38 and eq 43[a]

Δt	$\delta\tau_1$	n_1	n_2	n_3	$\log(\Delta E)$	R	$T_{stretch}$	T_{bend}	$T_{torsion}$	$T_{nonbonded}$
0.25	0.25	1	1	1	-3.7073	0.0022	30.7	186.0	399.7	10085.9
0.50	0.50	1	1	1	-3.0123	0.0087	15.5	93.3	201.2	5073.4
1.00	1.00	1	1	1	-2.2388	0.0398	7.7	47.9	102.8	2579.6
2.00	2.00	1	1	1	-1.0475	0.1981	4.2	23.6	52.5	1326.7
3.00	0.25	1	6	2	-2.9880	0.0102	31.9	189.1	407.6	1090.0
3.00	0.25	2	3	2	-2.9995	0.0102	32.1	92.3	198.7	1062.9
3.00	0.50	1	3	2	-2.7944	0.0127	15.6	93.6	201.0	1078.2

[a] $\log(\Delta E)$ and R are given by eq 48 and eq 49. Δt and $\delta\tau_1$ are in fentoseconds. Here $r_c = 6.0$ Å and $\Delta r = 2.0$ Å for all r-RESPA cases considered. $T_{stretch}$, T_{bend}, $T_{torsion}$, and $T_{nonbonded}$ are the CPU times in seconds spent in the stretch, bend, torsion, and nonbonded force routines, respectively. CPU time spent calculating r-RESPA neighbor lists is included in $T_{nonbonded}$.

breakup discussed above, using the switching function for each of the two cases. We mention, however, that other long/short breakups have been used in the case of constant long-range force approaches, such as those based upon the use of neighbor lists.[16] Here, we choose to use the switching function based decomposition for both r-RESPA and the velocity Verlet with the constant long-range force approach, so that any differences can be attributed to the integrators themselves. Identical parameters were used in each of the two cases, that is, $r_c = 6.0$ Å and $\Delta r = 2.0$ Å. In Figure 3 we compare the energy conservation obtained by using this constant long-range force approximation in conjunction with the velocity Verlet integrator with that of r-RESPA as a function of the long-range time step, i.e. the number of small time steps between long-range nonbonded force evaluation. The results indicate that the constant long-range approximation leads to very poor energy conservation for this case, whereas r-RESPA remains quite stable to significantly larger time steps for the same amount of CPU time.

Next, we compare the velocity Verlet algorithm with that of r-RESPA using a combination of the stiff/soft, internal/external, and the short/long-range nonbonded force separations. The results for the energy conservation and CPU times are given in Table 3. Here it is clear that r-RESPA allows for a reduction in the nonbonded derivative CPU time by a factor of almost 5 as compared to the velocity Verlet case with $\Delta t = 0.5$ fs for a comparable level of energy conservation. Additionally, in order to explore the question as to whether the r-RESPA method does indeed generate the correct dynamics of the system, we compare the spectral density $I(\tilde{\nu})$ as a function of the frequency $\tilde{\nu}$ in wavenumbers, obtained from the two methods, where

$$I(\tilde{\nu}) = \frac{1}{2\pi c}\int_0^\infty C_v(t)\cos(2\pi c\tilde{\nu}t)\,dt \tag{52}$$

and $C_v(t)$ is the normalized velocity autocorrelation function of the system,

$$C_v(t) = \frac{\langle \sum_{i=1}^{N} \mathbf{v}_i(t)\cdot\mathbf{v}_i(0)\rangle}{\langle \sum_{i=1}^{N} \mathbf{v}_i(0)\cdot\mathbf{v}_i(0)\rangle} \tag{53}$$

where c is the speed of light, $\mathbf{v}_i(t) = (v_x(t), v_y(t), v_z(t))$, the velocity of atom i at time t, and N is the total number of atoms. In Figure 4 we compare the r-RESPA (case 1) $\delta\tau_1 = 0.25$ fs, $n_1 = 1$, $n_2 = 6$, and $n_3 = 2$ to velocity Verlet with $\Delta t = 0.25$ fs. The results for r-RESPA (case 2) $\delta\tau_1 = 0.25$ fs, $n_1 = 2$, $n_2 = 3$, and $n_3 = 2$ and those for velocity Verlet with $\Delta t = 0.5$ fs are shown in Figure 5. In order to establish a quantitative estimate of the accuracy of the resulting spectral densities, we consider[15,20]

$$D \equiv \arccos\left(\frac{\vec{S}_1\cdot\vec{S}_2}{|\vec{S}_1||\vec{S}_2|}\right) \tag{54}$$

where

$$\vec{S} = (s_1, \ldots, s_n) \tag{55}$$

and the s_i are the spectral components at frequency i. 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 two r-RESPA cases considered above and compare them to the velocity Verlet case with $\Delta t = 0.5$ fs. For r-RESPA (case 1) we obtain $D = 0.250$, for r-RESPA (case 2) $D = 0.299$, and $D = 0.662$ for that of velocity Verlet using a time step of 0.5 fs. Thus, in each case, the r-RESPA method allows for a reduction in the nonbonded CPU time of almost a factor of 5 with a spectral accuracy better than that of 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. This can be seen more clearly if, for example, we examine a detailed comparison of the spectral densities for the two velocity Verlet cases (see Figure 6) in the vicinity of the small O–H stretch peak near 3722.5 cm^{-1}. The r-RESPA (case 1) is included for comparison.

Conclusion

In molecular dynamics simulations of macromolecules, one is often limited by the high computational expense associated with calculating quantities of physical interest to a reliable degree of accuracy. This is primarily due to the cost of calculating the nonbonded interactions which scales as N^2, where N is the number

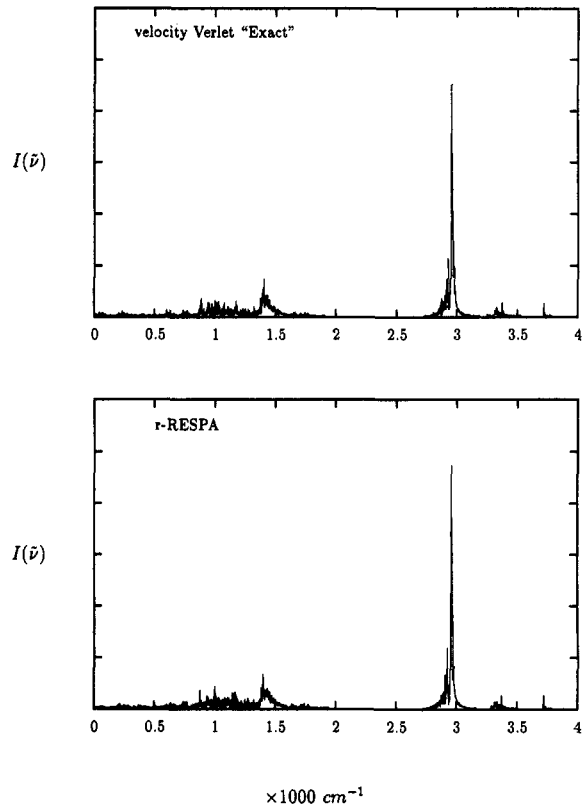

Figure 4. Comparison of spectral density $I(\tilde{\nu})$ of eq 52 as a function of wavenumber for velocity Verlet $\Delta t = 0.25$ fs (top) and r-RESPA using the propagator given by eq 38 and eq 43 with $n_1 = 1$, $n_2 = 6$, $n_3 = 2$, and $\delta t_1 = 0.25$ fs (bottom). Intensities are in arbitrary units.

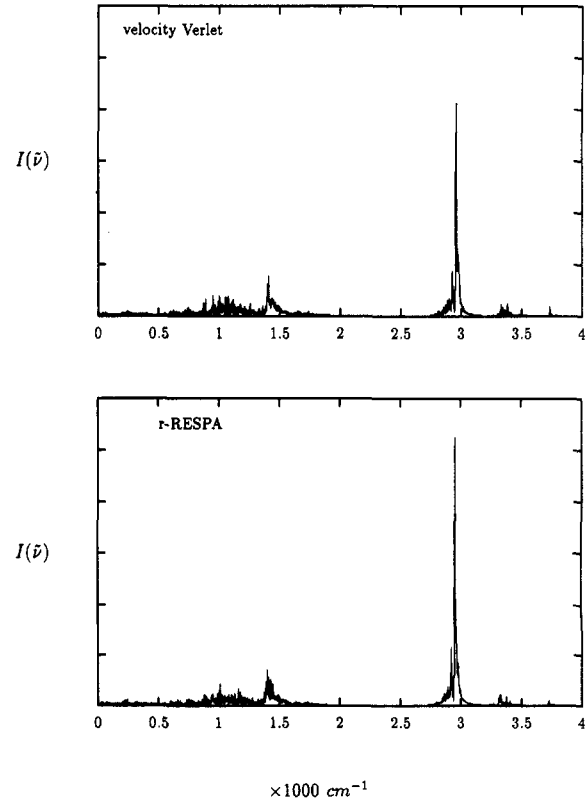

Figure 5. Comparison of spectral density $I(\tilde{\nu})$ of eq 52 as a function of wavenumber for velocity Verlet $\Delta t = 0.50$ fs (top) and r-RESPA using the propagator given by eq 38 and eq 43 with $n_1 = 2$, $n_2 = 3$, $n_3 = 2$, and $\delta t_1 = 0.25$ fs (bottom). Intensities are in arbitrary units.

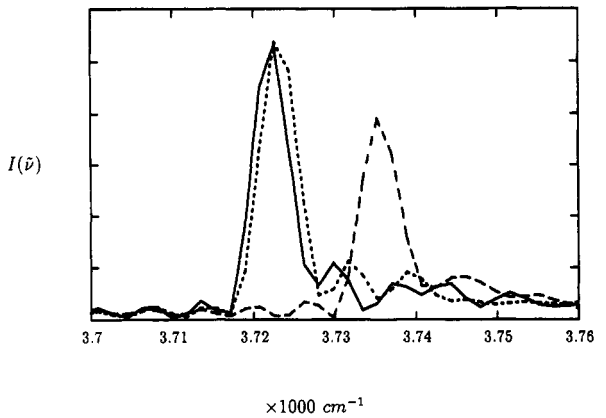

Figure 6. Detail of spectral density $I(\tilde{\nu})$ of eq 52 as a function of wavenumber for "Exact" (i.e. velocity Verlet Δt = 0.25 fs) (solid line), velocity Verlet Δt = 0.5 fs (dashed line), and r-RESPA using the propagator given by eq 38 and eq 43 with $n_1 = 1$, $n_2 = 6$, $n_3 = 2$, and δt_1 = 0.25 fs (dotted line). Intensities are in arbitrary units.

of particles of the system. Because the size of the time step is limited by the most rapidly varying degrees of freedom, standard numerical integration techniques become inefficient because such a very large number of nonbonded interactions must be calculated during a typical simulation of a macromolecule. Here we have presented a more efficient alternative which is based upon using a multiple-time-step integration scheme, whereby interactions evolving according to different time scales may be integrated with different time steps. The method, which is based upon a Trotter factorization of the classical Liouville operator, is shown to achieve a comparable level of accuracy at a fraction of the computational cost of the standard velocity Verlet integrator. For a molecular dynamics simulation of the small protein crambin *in vacuo*, this corresponds to a savings of approximately 4–5 times the CPU time spent calculating nonbonded interactions. Since the computational cost scales as N^2 with system size, it is expected that the method will be even more efficient for larger macromolecules. For example, when the method was applied to solid fullerene containing 32 C_{60} molecules,[15] or a total of N = 2880 interaction sites, one obtains a 20–40-fold speedup. Similar speedups might be expected for protein molecules with a comparable number of sites. Furthermore, we note that one need not simply limit decomposition of the nonbonded forces into that of short- and long-range contributions. The method is easily generalized to the case where the nonbonded forces are subdivided into a number of regions, or multiple "shells", each associated with a different time step. This approach may be advantageous for systems significantly larger than the one studied here. It is also possible that other subdivisions of the forces may lead to further reduction in CPU times for comparable accuracy, e.g. separating torsion interactions into "stiff" and "soft" contributions and perhaps separating interactions between light and heavy atoms. Additionally, the r-RESPA approach should be complimentary to other efficient numerical techniques such as the fast multipole method.[21]

Future applications of the method will include testing its use as a structure refinement tool in connection with the protein folding studies being conducted in our laboratory.[22,23]

Acknowledgment. We would like to thank Professor W. C. Still for providing us with the source code for the MACROMODEL/Batchmin molecular modeling program and for many helpful discussions. This work was supported by grants from the National Institutes of Health (NIH GM43340-01-A1) and from the NIH Division of Research Resources (SP41RR06892).

Appendix

The following is a schematic FORTRAN implementation of the molecular dynamics algorithm discussed in the text using the propagator as defined in eq 38 and the inner reference system propagator in eq 43. The Cartesian positions and velocities of an atom I, with mass AMASS(I), are given by X(I), Y(I), Z(I), and VX(I), VY(I), VZ(I). The atomic forces are contained in the arrays F1_X(I)-F4_Z(I). Here DELTAT corresponds to $\delta\tau_1$.

```
      CALL FORCES_F1
      CALL FORCES_F2
      CALL FORCES_F3
      CALL FORCES_F4
      DO N=1,NSTEPS
        DO M=1,NATOMS
          VX(M)=VX(M)
     $         +0.5*DBLE(N1*N2*N3)*DELTAT*F4_X(M)/AMASS(M)
          VY(M)=VY(M)
     $         +0.5*DBLE(N1*N2*N3)*DELTAT*F4_Y(M)/AMASS(M)
          VZ(M)=VZ(M)
     $         +0.5*DBLE(N1*N2*N3)*DELTAT*F4_Z(M)/AMASS(M)
        ENDDO
        DO I=1,N3
          DO M=1,NATOMS
            VX(M)=VX(M)
     $           +0.5*DBLE(N1*N2)*DELTAT*F3_X(M)/AMASS(M)
            VY(M)=VY(M)
     $           +0.5*DBLE(N1*N2)*DELTAT*F3_Y(M)/AMASS(M)
            VZ(M)=VZ(M)
     $           +0.5*DBLE(N1*N2)*DELTAT*F3_Z(M)/AMASS(M)
          ENDDO
          DO J=1,N2
            DO M=1,NATOMS
              VX(M)=VX(M)
     $             +0.5*DBLE(N1)*DELTAT*F2_X(M)/AMASS(M)
              VY(M)=VY(M)
     $             +0.5*DBLE(N1)*DELTAT*F2_Y(M)/AMASS(M)
              VZ(M)=VZ(M)
     $             +0.5*DBLE(N1)*DELTAT*F2_Z(M)/AMASS(M)
            ENDDO
            DO K=1,N1
              DO M=1,NATOMS
                VX(M)=VX(M)
     $               +0.5*DELTAT*F1_X(M)/AMASS(M)
                VY(M)=VY(M)
     $               +0.5*DELTAT*F1_Y(M)/AMASS(M)
                VZ(M)=VZ(M)
     $               +0.5*DELTAT*F1_Z(M)/AMASS(M)
              ENDDO
              DO M=1,NATOMS
                X(M)=X(M)
     $               +DELTAT*VX(M)
                Y(M)=Y(M)
     $               +DELTAT*VY(M)
                Z(M)=Z(M)
     $               +DELTAT*VZ(M)
              ENDDO
              CALL FORCES_F1
              DO M=1,NATOMS
                VX(M)=VX(M)
     $               +0.5*DELTAT*F1_X(M)/AMASS(M)
                VY(M)=VY(M)
     $               +0.5*DELTAT*F1_Y(M)/AMASS(M)
                VZ(M)=VZ(M)
     $               +0.5*DELTAT*F1_Z(M)/AMASS(M)
              ENDDO
            ENDDO
            CALL FORCES_F2
            DO M=1,NATOMS
              VX(M)=VX(M)
     $             +0.5*DBLE(N1)*DELTAT*F2_X(M)/AMASS(M)
              VY(M)=VY(M)
     $             +0.5*DBLE(N1)*DELTAT*F2_Y(M)/AMASS(M)
              VZ(M)=VZ(M)
```

```
$              +0.5*DBLE(N1)*DELTAT*F2_Z(M)/AMASS(M)
        ENDDO
      ENDDO
      CALL FORCES_F3
      DO M=1,NATOMS
        VX(M)=VX(M)
$            +0.5*DBLE(N1*N2)*DELTAT*F3_X(M)/AMASS(M)
        VY(M)=VY(M)
$            +0.5*DBLE(N1*N2)*DELTAT*F3_Y(M)/AMASS(M)
        VZ(M)=VZ(M)
$            +0.5*DBLE(N1*N2)*DELTAT*F3_Z(M)/AMASS(M)
      ENDDO
    ENDDO
    CALL FORCES_F4
    DO M=1,NATOMS
      VX(M)=VX(M)
$          +0.5*DBLE(N1*N2*N3)*DELTAT*F4_X(M)/AMASS(M)
      VY(M)=VY(M)
$          +0.5*DBLE(N1*N2*N3)*DELTAT*F4_Y(M)/AMASS(M)
      VZ(M)=VZ(M)
$          +0.5*DBLE(N1*N2*N3)*DELTAT*F4_Z(M)/AMASS(M)
    ENDDO
  ENDDO
```

References and Notes

(1) Verlet, L. *Phys. Rev.* **1967**, *159*, 98.
(2) Swope, W. C.; Anderson, H. C.; Berens, P. H.; Wilson, K. R. *J. Chem. Phys.* **1982**, *76*, 637.
(3) Brooks, C. L., III; Pettitt, B. M.; Karplus, M. *J. Phys. Chem.* **1985**, *83*, 5897.
(4) Ryckaert, J. P.; Ciccotti, G.; Berendsen, H. J. C. *J. Comput. Phys.* **1977**, *23*, 327.
(5) Anderson, H. C. *J. Comput. Phys.* **1983**, *52*, 24.
(6) van Gunsteren, W. F.; Karplus, M. *Macromolecules* **1982**, *15*, 1528.
(7) Toxaerd, S. *J. Chem. Phys.* **1987**, *87*, 6140.
(8) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Oxford University Press: Oxford, 1987.
(9) Teleman, O.; Jönsson, B. *J. Comput. Chem.* **1986**, *7*, 58.
(10) Tuckerman, M. E.; Berne, B. J. *J. Chem. Phys.* **1991**, *95*, 8362.
(11) Swindoll, R. D.; Haile, J. M. *J. Comput. Phys.* **1984**, *53*, 289.
(12) Tuckerman, M. E.; Berne, B. J.; Martyna, G. J. *J. Chem. Phys.* **1992**, *97*, 1990.
(13) Trotter, H. F. *Proc. Am. Math Soc.* **1959**, *10*, 545.
(14) Watanabe, M.; Karplus, M. *J. Chem. Phys.* **1993**, *99*, 8063.
(15) Procacci, P.; Berne, B. J. *J. Chem. Phys.*, in press.
(16) Mohamadi, F.; Richards, N. G. J.; Guida, W. C.; Liskamp, R.; Lipton, M.; Caufield, C.; Chang, G.; Hendrickson, T.; Still, W. C. *J. Comput. Chem.* **1990**, *11*, 440.
(17) Weiner, S. J.; Kollman, P. A.; NGuyen, D. T.; Case, D. A. *J. Comput. Chem.* **1986**, *7*, 230.
(18) Tuckerman, M. E.; Martyna, G. J.; Berne, B. J. *J. Chem. Phys.* **1991**, *94*, 6811.
(19) van Gunsteren, W. F.; Berendsen, H. J. C. *Mol. Phys.* **1977**, *34*, 1311.
(20) Skilling, J.; Bryan, R. K. *Mon. Not. R. Astron. Soc.* **1984**, *211*, 111.
(21) Greengard, L.; Rokhlin, V. *J. Comput. Phys.* **1985**, *60*, 187.
(22) Monge, A.; Friesner, R. A.; Honig, B. *Proc. Natl. Acad. Sci. U.S.A.* **1994**, *91*, 5027.
(23) Gunn, J. R.; Monge, A.; Friesner, R. A.; Marshall, C. H. *J. Phys. Chem.* **1994**, *98*, 702.