Vectorizing a General Purpose Molecular Dynamics Simulation Program

Olle Teleman and Bo Jönsson
Dept. of Physical Chemistry 2, Chemical Centre, POB 124, S-221 00 Lund, Sweden

Received 28 January, 1985; accepted 17 June 1985.

A molecular dynamics program for arbitrary molecular mixtures is presented. All intramolecular degrees of freedom are treated explicitly, which means that the program is based on central forces only. A double time step technique has been devised in order to separate rapidly varying, covalent forces from slowly varying ones. Typically, the ratio between the different time steps is about 10, with only a minor computational effort spent in the evaluation of the covalent forces. The program source code is arranged so as to obtain maximal efficiency on a vector processor, while still being portable. On a Cray 1A, a typical simulation of an ion-chelate in aqueous solution with 984 atoms requires a total of 2.9 μs/interaction with a spherical cutoff distance of 10Å.

INTRODUCTION

Computer simulation techniques play an important role in the statistical mechanical theory of condensed matter.[1] Both the Monte Carlo method of Metropolis et al.[2] and the solution of Newton's equations of motion—the Molecular Dynamics method—are common numerical techniques today. Since the first Molecular Dynamics (MD) simulation of hard spheres by Alder and Wainwright,[3] there has been a steady improvement and refinement of the MD technique. Continuous interatomic potentials of the Lennard-Jones type have been used to study systems such as liquid argon,[4] and the method has been extended so as to handle rigid polyatomic molecules like water.[5]

To simulate flexible molecules, with certain geometry parameters (most likely bond lengths) kept fixed, is a more difficult problem. The Langrangian approach with the use of generalized coordinates is straightforward, but fairly tedious and is only applicable to simple molecules.[6,7] Another technique, which has been widely used, is the so called "method of constraints".[8] In principle this procedure allows any bond length or bond angle to be held fixed during the simulation. Formally, the constraints enter into the equations of motion as an additional force and require only minor changes in a standard MD program. It is assumed that this is an accurate procedure, as long as the motions of the constrained and unconstrained degrees of freedom occur on well separated time scales. It has been found that constraining bond angles affects the thermodynamics of the system, while using bond length constraints is an acceptable approximation.[9] However, there are some indications that time-dependent properties can be sensitive even to bond length constraints.[10]

In this article we present a different approach based on central forces between all atoms or pseudoatoms (a pseudoatom is a group of atoms treated as a unit). Thus neighbor atoms in the covalent bond structure interact with harmonic forces both in bonds and angles. More distant atoms interact vía noncovalent interaction potentials in the usual way.[11] With this approach a simulation of a complicated biomolecule in aqueous solution is no more difficult than a simulation of an atomic liquid. A well known drawback of such an approach is that the fast vibrational motions will limit the time step that can be used in the numerical integration. However, this can be overcome by a multiple time step method,[12,13] where rapidly varying forces are evaluated more frequently than slowly varying ones. In the present program we make use of two different time steps Δt and ΔT, where $\Delta T/\Delta t \approx 5$–10. The advantage of such a division is that the interaction terms constituting the rapidly varying covalent forces are few and easily evaluated with a small time step Δt. The main computational effort is still spent in the evaluation of noncovalent forces with the large time step ΔT. Thus in a simulation of a biomolecule in aqueous solution the large time step ΔT is not limited by the bond angle or bond length vibrations.

The program to be described in the following sections was originally developed for simulation of biomolecules, but is completely general and applicable to all kinds of mixtures. It is based on a neighbor list technique, and the present version can handle periodic boundary conditions as well as cluster simulations in the microcanonical ensemble. Extensions to other kinds of boundary conditions and ensembles are straightforward.

An MD program for simulation of biomolecules in solution must by necessity lead to large scale computations, which have to be performed on a vector processor. Thus we have taken some care to design the program so that the innermost loops will "vectorize". The vectorization logic is general and not confined to any special machine. As a consequence of the efficient vectorization achieved in the innermost force loop, other parts of the force evaluation become dominant. This has the effect that an increased complexity in the intermolecular potential hardly affects the total cpu time.

The program so far has been used to study pure water, organic chelators, and ionic micelles in aqueous solution, as well as the Ca^{2+} binding protein Parvalbumin. The largest simulation performed comprised 7979 atoms.

We conclude this introductory section by noting that several important issues, such as energy stability and long-range interactions, as well as the choice of numerical algorithm, will be very cursorily treated in this article. These are important aspects of an MD simulation, but the scope of this article has to be limited. Anyway, improved numerical algorithms and special techniques for handling long-range interactions can easily be incorporated in the present vectorized program.

General Outline of the Program

The general structure of the program is cyclic (see Fig. 1) and exclusively contains the code involved in the actual solution of the equations of motion. All analyses of trajectories, as well as the preparation of the file containing the interaction matrix are done separately in other programs. The program is installed on a scalar ND 500 computer, and on a Cray 1A. The program was developed from an earlier MD program due to Engström[14] written to simulate rigid molecules. The program is available on request from the authors.

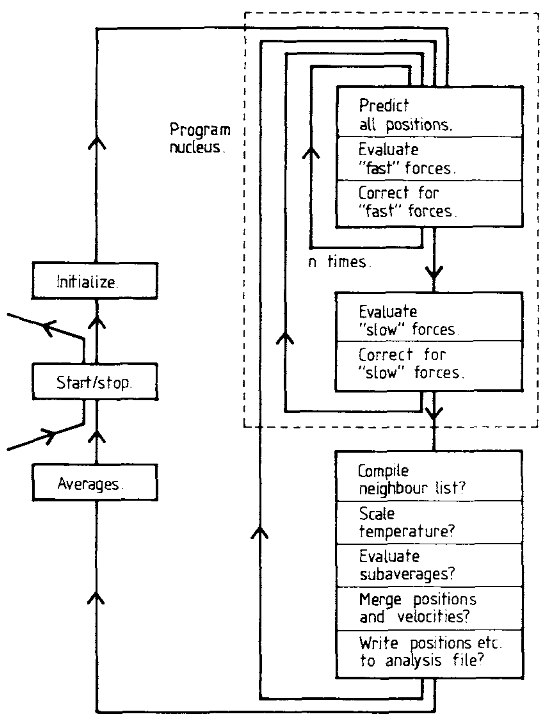


Figure 1. General structure of the MD program. Fast forces are those that vary rapidly. n is the ratio between the two time steps ΔT and Δt. A question mark indicates that the intervals between executions of the corresponding operation can be chosen.

To integrate the Newtonian equation of motion the program uses a predictor-corrector algorithm, where the predictor is an ordinary Taylor expansion and the corrector uses coefficients according to Gear.[15] The order of the algorithm may be chosen.

Systems containing any number of pseudoatom species may be simulated, and these pseudoatoms may be connected with harmonic potentials to form molecules in any way. The total number of pseudoatoms is limited only by the available core memory (see below). The MD program cannot handle rigid molecules, as it relies on central forces. The following two sections will describe the potential used for the simulations, and the subsequent ones will deal with the measures taken to generate fast code.

Covalent Potential

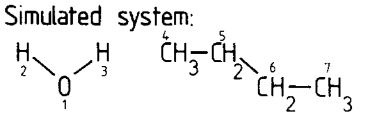
The covalent potential is assumed to be harmonic in bond lengths and bond angles and periodic and even in dihedral angles. Hydrogen bonds and improper dihedral angles[9] are not explicitly included in the covalent potential, as the former are just Coulombic interactions, and the bond angle potentials take care of the latter. The potential is described by eq. (1), and the parameters are taken from the literature.[6,9,16–18]

$$V_c = \sum_{\text{bonds}} B_i(r_i - r_{i,e})^2 + \sum_{\substack{\text{bond}\\\text{angles}}} A_j(\alpha_j - \alpha_{j,e})^2 + \sum_{\substack{\text{dihedral}\\\text{angles}}} \left(C_{k,o} + \sum_{l=1} C_{k,l}\cos l\varepsilon_k\right) \quad (1)$$

Other force fields are available (see ref. 19–28). $C_{k,l}$ is normally 0 for $l > 3$, so that in most cases the summation over l may be truncated after three terms. The constant $C_{k,o}$ ensures that $V_c \geq 0$. Forces are calculated as the gradient of the potential in Cartesian laboratory coordinates. This is trivial for bonds and bond angles, and straightforward but tedious for dihedral angles. An outline is given in ref. 29.

All pseudoatoms in the simulated system are explicitly numbered. From the bond structure of the system, lists are constructed that contain the atoms constituting each bond; similar lists are generated for bond angles and dihedral angles. In accordance with these lists arrays are constructed which explicitly contain the corresponding equilibrium bond distances and force constants, equilibrium bond angles and angular force constants, and the cosine coefficients of the dihedral angle potential, respectively [see eq. (1)]. This is shown diagrammatically in Figure 2 for a simple example. These lists and the corresponding parameter arrays are permanent, as it is assumed that no change in the covalent structure occurs, i.e., no reactions are allowed during the simulations. This extensive listing of covalent potential parameters facilitates the calculation of the corresponding forces, as the program only has to run through these lists. The technique also simplifies the code, but it will still not vectorize entirely, as the program has to access atom positions and store the calculated force contributions by indexed addressing. The

Simulated system:



Potential library:

Bonds.	r_e(Å)	B (kJ·mol^{-1}Å^{-2})		
Water O - H	1.00	2318		
CH3 - CH2	1.54	1674.		
CH2 - CH2	1.52	1674.		
Angles:	α (°)	A (kJ·mol^{-1}rad^{-2})		
Water: H - O - H	109.5	191.5		
CH3 - CH2 - CH2	109.5	125.5		
Dihedral angle.	C0	C1	C2	C3 (kJ/mol)
CH3 - CH2 - CH2 - CH3	2.09	0.0	0.0	2.09

Resultant explicit arrays:

Bonds:	r_e	B		
1 2	1.00	2318.		
1 3	1.00	2318.		
4 5	1.54	1674.		
5 6	1.52	1674.		
6 7	1.54	1674		
Angles:	α	A		
2 1 3	109.5	191.5		
4 5 6	109.5	125.5		
5 6 7	109.5	125.5		
Dihedral angle:	C0	C1	C2	C3
4 5 6 7	2.09	0.0	0.0	2.09

Figure 2. This explains how the explicit arrays used to calculate the forces due to the covalent potential are constructed for a system consisting of one water molecule and one butane molecule. Small numbers in the drawing show how the seven pseudoatoms are made up. The system contains five covalent bonds listed in BONDS (1:5, 1:2); therefore, r_e and B each will have five elements containing the corresponding equilibrium bond lengths and potential coefficients. Analogous arrays are used for bond angles and the dihedral angle.

same problem arises for the noncovalent forces, but it can, as we will see below, be partially overcome.

Noncovalent Potential

The noncovalent potential is assumed to consist of a 6-12 Lennard-Jones potential plus an electrostatic contribution, see eq. (2).

$$V_{nc} = \begin{cases} \sum_{\text{pairs}} \left(\dfrac{A_{ij}}{r_{ij}^6} + \dfrac{B_{ij}}{r_{ij}^{12}} + \dfrac{q_i q_j}{4\pi\varepsilon\varepsilon_o r_{ij}}\right), & r_{ij} \leq r_{\text{cut}} \\ 0, & r_{ij} > r_{\text{cut}} \end{cases} \quad (2)$$

Partial charges q_i are taken from the literature (refs. 6, 30, 31, and others). The Lennard-Jones coefficients A_{ij} and B_{ij} are calculated from the Kirkwood-Slater formula,[32] with parameters from refs. 9 and 32. A large variety of water models is available (see ref. 16 and references therein). The Simple Point Charge (SPC) model of Berendsen et al.[16] is parametrized according to eq. (2) and therefore has been used. Noncovalent potential parameters for many different pseudoatom species are also available in refs. 33–36.

V_{nc} [Eq. (2)] is used to describe not only the intermolecular interactions, but also the interactions between atoms belonging to the same molecule, but separated by a small number of covalent bonds, normally three or four. The relative dielectric constant ε is assumed to be 1, since all partial charges are treated explicitly.

The truncation of V_{nc} at $r_{ij} = r_{\text{cut}}$ greatly reduces the numbers of pairs to be evaluated, but necessitates keeping track of those pairs ij for which $r_{ij} < r_{\text{cut}}$. The MD program uses a neighbor list to do this.[37] Setting r_{cut} to 10 Å in an aqueous environment means that interactions with some 135 neighbor water molecules, or about 400 atoms, must be considered. For a system of 1000 atoms, one thus has to list 200 000 neighbors, if reciprocity is used. The neighbor list does not have to be compiled every time step, but the use of a large r_{cut} is a prerequisite for a long interval between compilations. For a cutoff distance of 10 Å and an assumed overlarge value of $8 \cdot 10^{-9}$ m^2/s for the water diffusion constant (at 300 K), one finds that about three water molecules enter or leave the neighbor volume in 5 fs. (We have set the interval between neighbor list compilations to 4.8 fs.) This is only a very small part of the total of about 135 water molecules in the neighbor volume. The value of $8 \cdot 10^{-9}$ m^2/s for the diffusion constant of water is much larger than the experimental value, but simulations of SPC water yield diffusion constants too large.[38]

The truncation of the noncovalent potential at $r = r_{\text{cut}}$ causes a drift in temperature. To remedy this, the temperature is evaluated at intervals and all velocities are uniformly scaled to attain the preset temperature. Use of a switching function[42] to truncate the potential smoothly would eliminate the temperature drift. Vectorization properties would not change, but computing time would increase by some 10–30%, depending on the order of the algorithm. While appealing, this technique is not universally of great advantage.[43]

In order to avoid ionic effects at the cutoff distance, water molecules are considered to be either entirely inside or entirely outside the neighbor volume. In consequence, water hydrogens may be excluded from the neighbor list (see below).

Nearly all of the computing time is spent in calculating the force contributions from eq. (2). To do this for one pseudoatom pair involves the calculation of r_{ij}^{-2}, r_{ij}^{-6}, r_{ij}^{-12}, r_{ij}^{-1} (which implies a square root), $q_i q_j$, and more. For optimal speed these floating point operations have to be done in fully vectorized loop, i.e., it cannot be done directly by using a column of the neighbor list as an index vector. Instead, the calculation is divided into three parts: (i) Collect neighbor positions and parameters into temporary contiguous arrays, which is a (nonvectorizing) "gather" operation; (ii) loop over these arrays to calculate a temporary contiguous force contribution array (vectorized operations); and (iii) add forces from (ii) to the correct pseudoatom total forces, which is a (nonvectorizing) "scatter" operation. The following section will deal with Parts (i) and (iii), and the subsequent one with Part (ii).

Gathering and Scattering

Part (i) uses the column i of the neighbor list to get the numbers j designating the neighbors of atom i, their positions, charges, and Lennard-Jones parameters A_{ij} and B_{ij}. Further, whenever a water oxygen is found in the neighbor list, two corresponding hydrogens are to be included as well. Straightforward Fortran code to do this will be slow, but can be improved upon by the following methods:

(a) Use of the ANINT (Fortran) or CVMGx (Cray Vector Merge)[39] functions will allow the code that provides for the periodic boundary conditions to vectorize. In consequence it is put in a loop of its own:

```
DO 14 J = 1,N
  FLX(J) = -XEDGE2 * ANINT(XEDG2I *
&      [XI-XV(J)])
  FLY(J) = -YEDGE2 * ANINT(YEDG2I *
&      [YI-YV(J)])
  FLZ(J) = -ZEDGE2 * ANINT(ZEDG2I *
&      [ZI-ZV(J)])
14 CONTINUE
```

where XI is the x coordinate of atom i, XV(J) those of its neighbors, XEDGE2 the x side of the box, XEDG2I its inverse, and N the number of nonwater neighbors j. The calculated vectors FLX(J) are added to XV(J) before cal-

```
      ...
      DO 3  J=1,NBMAX
        XDJ = XI - XV(J) + FLX(J)
        YDJ = YI - YV(J) + FLY(J)
        ZDJ = ZI - ZV(J) + FLZ(J)
        R2J = 1./(XDJ*XDJ + YDJ*YDJ + ZDJ*ZDJ)
        RJ = SQRT(R2J)
        R6J = R2J*R2J*R2J
        UH(J) = QIF*VQ(J)*RJ + (C6(J) + C12(J)*R6J)*R6J
        FHJELP = (QIF*VQ(J)*RJ  + (6.*C6(J)+12.*C12(J)*R6J)
     &    *R6J)*R2J
        VFX(J) = FHJELP*XDJ
        VFY(J) = FHJELP*YDJ
        VFZ(J) = FHJELP*ZDJ
 3    CONTINUE
      UHELP = SSUM(NBMAX,UH,1)
      FXI = SSUM(NBMAX,VFX,1)
      FYI = SSUM(NBMAX,VFY,1)
      FZI = SSUM(NBMAX,VFZ,1)
      ...
```

Figure 3. A well vectorized loop, for the calculation of noncovalent forces. Atom *i* has NBMAX neighbors *j*. XI is the *x* coordinate of atom *i*, XV(J) those of its neighbors. FLX(J) takes care of periodic boundary conditions in *x*. VFX(J) is the negative force on neighbor *j* in the *x* direction (analogously for *y* and *z*). The potential energy is obtained by summing UH(J) and the force on atom *i* by summing VFX, VFY, and VFZ. This is done with the Cray system subroutine SSUM.

culating the distance between *i* and *j* (see Fig. 3). The *y* and *z* coordinates are dealt with in an analogous way.

(b) If the subroutine that generates the neighbor list puts water neighbors at the end of each column and returns a variable to specify where they begin, it is possible to loop over nonwater and water neighbors separately; this enables water hydrogens to be included without testing. The increment of the loop in (a) will now be 3 and the loop itself will contain six further statements of the type

FLX(J + 1) = FLX(J)

as the periodic boundary conditions are to apply in the same way to all three atoms in the water molecule. These statements amount to an apparent vector dependency,[39] but as this dependency is not real, vectorization can be forced by compiler directives.

(c) Nonvectorized loops have severe loop overhead times, which are comparable to the execution time required for two to four Fortran assignment statements. Thus loops containing only a few statements should be avoided. "Stripmining" of nonvectorized loops will also help, as shown by the following example.

```
      REAL A(1000),B(1000)
      INTEGER IND(1000)
C
      DO 1 I = 1,1000
        A(IND)(I)) = B(I)
1     CONTINUE                                  (4)
      DO 1 I = 1,1000,2
        A(IND(I)) = B(I)
        A(IND)(I + 1)) = B(I + 1)
1     CONTINUE                                  (4')
```

Replacement of (4) by (4′) will cut Cray 1A execution time by 38%. Stripmining using an increment of 3 will save 51%.

Parts (i) and (iii) may also be written very compactly with extensive use of the Cray Fortran (CFT) system subroutines GATHER and SCATTER, but this code is slower than the one already outlined. GATHER and SCATTER are about 3.5 times faster than the explicit one statement loop Fortran code for an array of 1000 elements, but this is more than overcome by gathering more than one array in the same loop and by stripmining.

A Well-vectorized Loop

Part (ii) consists of one loop over contiguous arrays. This loop vectorizes well[39] (see Fig. 3). On a Cray 1A this code has been timed at 50+/−2 Mflops/s, where an estimate of four floating point operations have been used for the square root. The simulated system contained 984 atoms under periodic boundary conditions and at liquid density; r_{cut} was set to 10 Å. This speed does not vary much with the composition of the simulated system, but increases when r_{cut} increases and vice versa.

Core Memory Limitations

The available core memory determines the maximal size of the systems that can be simulated. With 200 listed neighbors for each atom the program needs about 280 words of core memory per atom. This figure depends slightly on the covalent bond structure of the system, since more memory per atom is required if the bond structure is complex.

If the system consists mainly of solvent molecules, only one atom in each neighboring solvent molecule needs to be included in the neighbor list, i.e., for an aqueous system the neighbor list is reduced by eight-ninths. The core memory requirement is then some

120 words per atom. On a computer with 1 Mword of core, the maximal system size is 3000–8000 atoms.

Memory demand can be reduced by coding the neighbor and covalent lists, or by using half word or bit techniques. However, the concomitant decoding will slow down the program, and a conflict arises between speed and memory economy. Program speed was considered to be more important for our simulations and consequently we have not packed the lists.

Use of Multiple Time Steps

The maximal allowable time step is regulated by the assumption of constant forces, i.e., by the decay of the force autocorrelation functions. In most cases this depends on bond forces in a bond involving a hydrogen atom. The oscillation period of these vibrations is of the order of 10^{-14} s, and a time step of <0.2 fs is necessary to integrate the vibrational motion accurately. The noncovalent forces vary much more slowly and a multiple time step method allows the use of a longer interval between their computation. The evaluation of noncovalent forces is very time-consuming hence the multiple time step method reduces the computer time requirement.

The force on a particle can arbitrarily be divided into two components, and from these two components we may write

$$\mathbf{F}_i(\mathbf{R}) = \mathbf{F}_{i,a}(\mathbf{R}) + \mathbf{F}_{i,b}(\mathbf{R})$$

$$\mathbf{a}_{i,a} \triangleq \frac{1}{m_i}\mathbf{F}_{i,a}(\mathbf{R}), \mathbf{a}_{i,b} \triangleq \frac{1}{m_i}\mathbf{F}_{i,b}(\mathbf{R}) \qquad (5)$$

where i refers to any atom in the system, $\mathbf{r}_i$ is the position of that atom and $\mathbf{R} = \mathbf{r}_i, \ldots, \mathbf{r}_N$, where N is the number of atoms in the system. It follows that

$$\frac{1}{m_i}\mathbf{F}_i(\mathbf{R}) = \frac{1}{m_i}[\mathbf{F}_{i,a}(\mathbf{R}) + \mathbf{F}_{i,b}(\mathbf{R})]$$
$$= \mathbf{a}_{i,a} + \mathbf{a}_{i,b} = \mathbf{a}_i = \ddot{\mathbf{r}}_i \qquad (6)$$

Thus the acceleration $\mathbf{a}_i$ equals the sum of $\mathbf{a}_{i,a}$ and $\mathbf{a}_{i,b}$. It is also clear that $\mathbf{r}_i$ can be divided in such a way that

$$\mathbf{a}_{i,a} = \ddot{\mathbf{r}}_{i,a}, \mathbf{a}_{i,b} = \ddot{\mathbf{r}}_{i,b}$$
$$\text{with} \quad \mathbf{r}_{i,a} + \mathbf{r}_{i,b} = \mathbf{r}_i \quad (7)$$

This is not unambiguous! Any contribution to $\mathbf{r}_i$ that is linear in time can be assigned either to $\mathbf{r}_{i,a}$ or $\mathbf{r}_{i,b}$ without violating eq. 7. We have

$$\ddot{\mathbf{r}}_{i,a} = \frac{1}{m_i}\mathbf{F}_{i,a}(\mathbf{R}), \ddot{\mathbf{r}}_{i,b} = \frac{1}{m_i}\mathbf{F}_{i,b}(\mathbf{R}) \qquad (8)$$

which amounts to a system of $2N$ coupled differential equations, instead of the usual system of N equations.

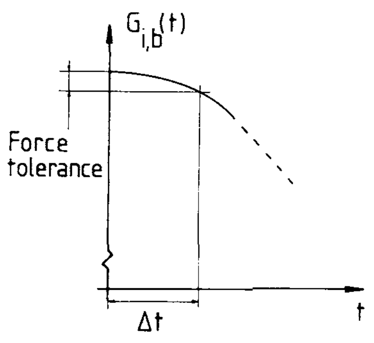


Figure 4. The autocorrelation function $G_{i,b}(t)$ of the most rapidly varying force is chosen to determine Δt. The larger the force tolerance, the less accurate the simulated trajectories will be.

The Taylor expansion has the property that taking n steps of Δt is equal to taking one step of $n \cdot \Delta t$. Furthermore, if Δt is short compared with the variations in force, the corrections applied by the Gear algorithm are very small. Using this and assuming $\mathbf{F}_{i,b}$ to vary faster than $\mathbf{F}_{i,a}$ a scheme comprising the two innermost loops of Figure 1 is obtained. Now Δt is obtained from $G_{i,b}(t) = \langle \mathbf{F}_{i,b}(t) \cdot \mathbf{F}_{i,b}(0)\rangle$, as in Figure 4, where the most rapidly decaying $G_{i,b}(t)$ should be selected. Application of the same technique to the most rapidly decaying $G_{i,a}(t) = \langle \mathbf{F}_{i,a}(0)\rangle$ yields an upper limit of ΔT, $\Delta T = n \cdot \Delta t$. In practice, n may have to be chosen smaller than that. The division of $\mathbf{F}_i$ into $\mathbf{F}_{i,a}$ and $\mathbf{F}_{i,b}$ is heuristic and as a consequence in general there will be a coupling between the two force components.

This approach differs from that of Swindoll and Haile[44] in that they expand the slowly varying force component itself in time and not a corresponding set of positions.

We have investigated two force subdivisions according to eq. (5). One of these subdivisions considers the covalent bond forces (due to the first term on the right-hand-side of eq. (1)) to vary rapidly. $\mathbf{F}_{i,a}$ is, therefore, made up of all the other terms in eqs. (1) and (2). Force autocorrelation functions for this force subdivision are plotted in Figure 5(a) for water oxygens. The oscillation in $G_{i,b}(t)$ corresponds to the bond vibration, that in $G_{i,a}(t)$ to

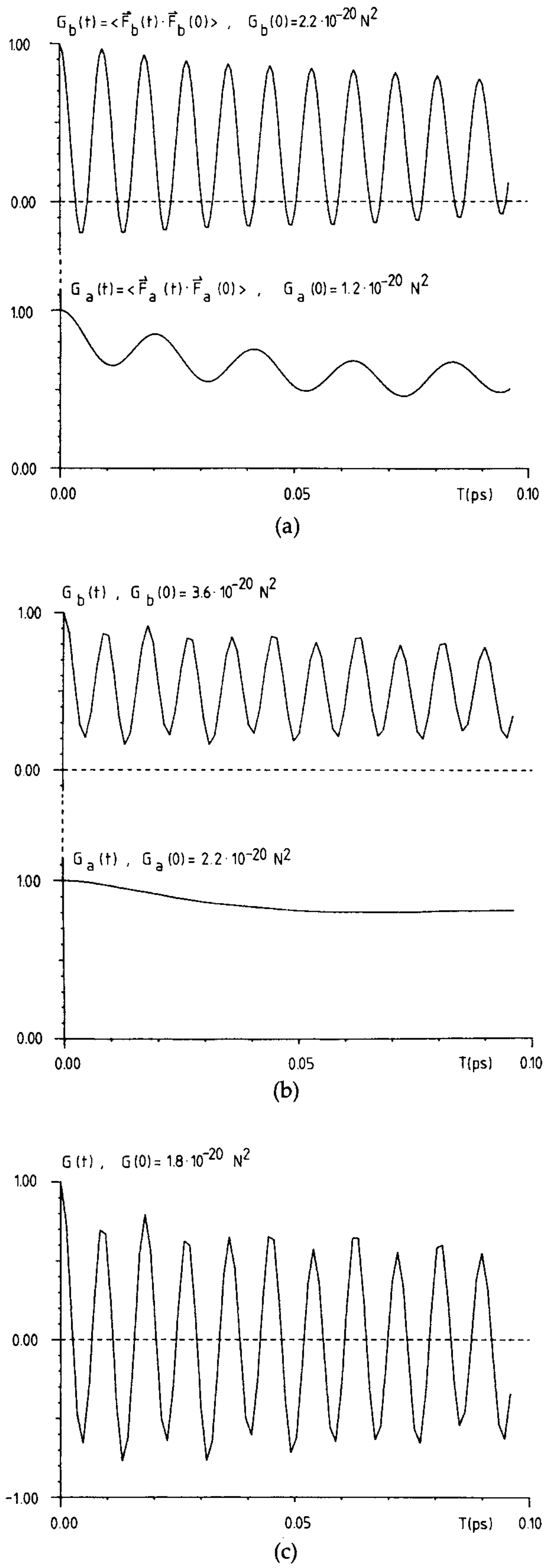


Figure 5. Force autocorrelation functions for bulk water oxygen atoms. (a) For covalent bond forces [$G_b(t)$] and all other force contributions [$G_a(t)$]; (b) for covalent bond and bond angle force contributions [$G_b(t)$] and all other force contributions [$G_a(t)$], i.e., noncovalent forces; and (c) for the total force.

the bond angle vibration. The latter has a period of only slightly more than twice that of the former, and a value of $n > 5$ is clearly inadequate. This is corroborated by the fact that the drift in total energy increases as soon as n is set larger than 5, when $\Delta t = 0.2$ fs. The other force subdivision assigns both bond and bond angle forces (from the first and second terms on the right-hand-side of eq. (1)) to the fast component $\mathbf{F}_{i,b}$. The corresponding $G_{i,b}(t)$ and $G_{i,a}(t)$ are shown in Figure 5(b), and it is obvious that $n = 8$ or even larger will be satisfactory. The total force autocorrelation function [see Figure 5(c)] is dominated by the covalent forces; this could also have been inferred from the values for $\langle|\mathbf{F}_i(0)|^2\rangle$ in Figure 5(b). The maximal values for Δt and n also depend on the composition of the simulated system. If the latter is devoid of explicit light atoms, i.e., hydrogens, Δt can be chosen to be larger. This applies to some in *in vacuo* simulations of macromolecules and lipid simulations.

The multiple time step method still can be easily applied if the force is separated into more than two components.

Owing to the ambiguity in the separation, eq. 7, the two solutions $\mathbf{R}_a$ and $\mathbf{R}_b$, as well as $\dot{\mathbf{R}}_a$ and $\dot{\mathbf{R}}_b$, will wander away in opposite directions. To prevent eventual numerical problems, $\mathbf{R}_b$ is transferred to $\mathbf{R}_a$ and $\dot{\mathbf{R}}_b$ to $\dot{\mathbf{R}}_a$ at long intervals. This phenomenon is small but noticeable on a 32 bit computer, but almost nonexistent on the Cray 1A, which uses 64 bit words.

The question arises whether a separation of short-range and long-range contributions to the noncovalent forces will allow an even larger time step for distant forces. Correcting less frequently for distant forces enables the use of a larger cutoff distance. In principle this is a feasible strategy. Examples of autocorrelation functions of long-range and short-range forces are plotted in Figure 6. Application of the procedure of Figure 4 would seemingly indicate that the time step for evaluation of distant electrostatically dominated forces only may be chosen slightly larger than that of the short-range forces. As the contribution to the total force is much smaller from distant forces than from short-range forces, a larger time step will not cause a corresponding inaccuracy in the calculated forces. This argument is based on one case

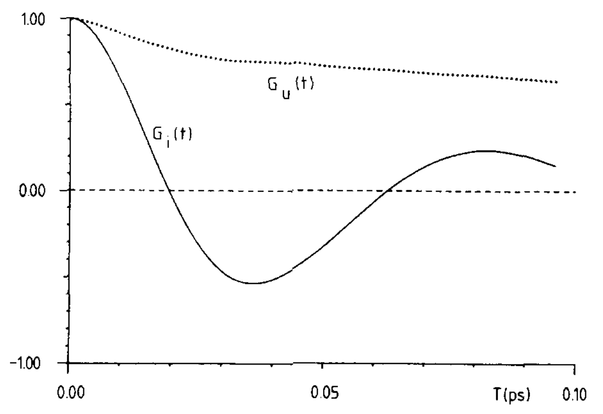


Figure 6. Autocorrelation functions for proximal and distant forces on a calcium ion in a simulation of Ca^{2+} + Ethylene-diammine tetraacetic acid (EDTA) and 317 water molecules. $G_i(t)$ corresponds to forces due to neighbors closer than 8 Å and $G_u(t)$ to those due to neighbors further away than 8 Å but closer than 10 Å; r_{cut} = 10 Å.

only, however, and a generalization may not be straightforward.

Simulation Times

Even with the use of multiple time step techniques, most of the computing time is spent evaluating noncovalent forces. The timings of three sample simulations are presented in Table I, which also includes force evaluation times per interaction pair. This figure compares favorably with those previously reported in the literature.[37,40,41] From Table I it is clear that the program is preeminently suited for simulation of aqueous solutions under periodic boundary conditions. For systems such as Parvalbumin *in vacuo* the number of noncovalent neighbor pairs is relatively small and the program will spend comparatively more time in its less well optimized parts. In the Parvalbumin simulation only 32% of the cpu time is spent evaluating noncovalent forces and, therefore, more attention should be given to the bond angle force calculation and the neighbor list compilation.

While it is important that the forces are evaluated efficiently, the choice of algorithm is of equal importance. We are aware that, in some cases, other algorithms may allow the use of larger time steps than those of the present simulations. However, the Gear algorithm is exceedingly well suited to multiple time step algorithms, and thus for the accurate treatment of disparate time scales.

Table I. Cray 1A execution times for three sample systems. Water refers to 216 water molecules under periodic boundary conditions and at experimental density. The system contained 648 atoms, 432 covalent bonds, 216 bond angles and no dihedral angles; r_{cut} = 8.5 Å. Parvalbumin was simulated *in vacuo*. The system contained 983 atoms, 993 covalent bonds, 1429 bond angles, and 1871 dihedral angles; r_{cut} = 10 Å. The micellar system contained a micelle comprising 15 sodium octanoate molecules surrounded by 1094 water molecules. Periodic boundary conditions were applied. The system contained 3447 atoms, 2323 covalent bonds, 1229 bond angles, and 105 dihedral angles; r_{cut} = 10 Å. In all three simulations $F_{i,b}$ consisted of covalent bond and bond angle forces. Numbers in parentheses are percent of total cpu time. The cpu time per interaction is given in microseconds for the different forces.

	Water	Parvalbumin	Micelle
Number of neighbor pairs	82900	83860	619670
Neighbor list interval	24 (4.8 fs)	24 (4.8 fs)	24 (4.8 fs)
No. of time steps (Δt)	48000 (9.6 ps)	1920 (0.384 ps)	960 (0.192 ps)
Total cpu time	2125.22 s	223.00 s	297.55 s
$n = \Delta T/\Delta t$	6	6	6
Predict	108. s (5.1)	6.6 s (3.0)	11.4 s (3.8)
Force evaluation	1705. s (80.3)	136.6 s (61.3)	244.8 s (82.3)
Correct	141. s (6.7)	7.2 s (3.2)	17.0 s (5.7)
Neighbor list compilation	153. s (7.2)	72.0 s (32.3)	24.3 s (8.2)
Temperature scaling	0.2 s (0.0)	0.0 s (0.0)	0.0 s (0.0)
Write, merge	15. s (0.7)	0.5 s (0.2)	0.0 s (0.0)
Forces: Noncovalent	1423. s (67.1)/ 2.2 μs	71.5 s (32.1)/ 2.7 μs	212.4 s (71.5)/ 2.1 μs
Bonds	92. s (4.3)/ 4.4 μs	8.5 s (3.8)/ 4.5 μs	10.0 s (3.4)/ 4.5 μs
Bond angles	187. s (8.8)/18.1 μs	45.8 s (20.6)/16.7 μs	21.7 s (7.3)/18.4 μs
Dihedral angles	0. s (0.0/ —	10.5 s (4.7)/18.5 μs	0.4 s (0.1)/25.6 μs
Noncovalent (i) "gather"	478. s (22.8)	31.9 s (14.5)	73.8 s (25.0)
forces: (ii) "calculate"	586. s (28.0)	24.1 s (11.0)	85.2 s (28.9)
(iii) "scatter"	342. s (16.3)	14.6 s (6.6)	51.8 s (17.6)
Total time/neighbor pair ev.	3.2 μs	8.3 μs	3.0 μs
Ps/cpu hour	16.2	6.20	2.32

CONCLUSION

The care taken to adapt the source code of the MD program has resulted in absolute code that is very well vectorized and, therefore, very fast. The reported execution times are even shorter than those reported for corresponding routines written in assembler code,[37] and this emphasizes the inherent potential of the Fortran language, at least for the present needs. The other main feature of this MD program is the use of two different time steps, which, in exchange for a marginally increased core memory requirement, facilitates the investigation of phenomena that occur simultaneously but on different time scales. The multiple time step technique allows expeditious simulation, while also including an explicit description of all intramolecular degrees of freedom.

Note added in proof: Further program refinement and a new Fortran compiler have reduced computing times by about 25% compared to the figures given in Table I.

We want to thank Dr. Sven Engström for placing the SEMD program at our disposal and for his advice and friendly help.

References

1. J.-P. Hansen and I. R. MacDonald, *Theory of Simple Liquids,* Academic Press, London, 1976.
2. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *J. Chem. Phys.,* **21,** 1087 (1953).
3. B.J. Alder and T.E. Wainwright, *J. Chem. Phys.,* **27,** 1208 (1957).
4. A. Rahman, *Phys. Rev.* **136,** A405 (1964).
5. A. Rahman and F. H. Stillinger, *J. Chem. Phys.,* **55,** 3336 (1971).
6. J.-P. Ryckaert and A. Bellemans, *Chem. Phys. Lett.,* **30,** 123 (1975).
7. J.-P. Ryckaert and A. Bellemans, *Faraday Discuss. Chem. Soc.,* **66,** 95 (1978).
8. J.-P. Ryckaert, G. Cicotti, and H. J. C. Berendsen, *J. Comp. Phys.,* **23,** 327 (1977).
9. W. van Gunsteren and M. Karplus, *Macromolecules,* **15,** 1528 (1982).
10. See Figures 5 and 8 of ref. 9.
11. B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus, *J. Comp. Chem.,* **4,** 187 (1983).
12. W. B. Street, D. J. Tildesley, and G. Saville, in *Computer Modeling of Matter,* P. Lykos, Ed., ACS Symposium Series 86, Washington, 1978.
13. W. B. Street, D. J. Tildesley, and G. Saville, *Mol. Phys.,* **35,** 539 (1978).
14. Program available from S. Engström, Dept. Physical Chemistry, Univ. of Umeå, S-901 87 Umeå, Sweden.
15. C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations,* Prentice Hall, Englewood Cliffs, N.J., 1971, p. 148–154.
16. H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, and J. Hermans, in *Intermolecular Forces,* B. Pullman, Ed., D. Reidel, Dordrecht, 1981, p. 331–342.
17. D. Dolphin and A.E. Wick, *Tabulation of Infrared Spectral Data,* Wiley Interscience, New York, 1977.
18. G. Herzberg, *Molecular Spectra and Molecular Structure: Infrared and Raman Spectra of Polyatomic Molecules,* Van Nostrand, Princeton, 1945.
19. M. Levitt, in *Protein Folding,* R. Jaenicke, Ed., Elsevier/North Holland, Amsterdam, 17, 1980.
20. M. Levitt, *J. Mol. Biol.,* **82,** 393 (1974).
21. A. Warshel, M. Levitt, and S. Lifson, *J. Mol. Spectrosc.,* **33,** 84 (1970).
22. S. Karplus and S. Lifson, *Biopolymers,* **10,** 1973 (1971).
23. P.J. Rossky, M. Karplus, and A. Rahman, *Biopolymers,* **18,** 825 (1979).
24. J.M. Blaney, P.K. Weiner, A. Dearing, P.A. Kollman, E.C. Jorgensen, S.J. Oatley, J.M. Burridge, and C.C.F. Blake, *J. Am. Chem. Soc.,* **14,** 6244 (1982).
25. S.J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, and P. Weiner, *J. Am. Chem. Soc.,* **106,** 765 (1984).
26. F.A. Momany, R.F. McGuire, A.W Burgess, and H.A. Scheraga, *J. Phys. Chem.,* **79,** 2361 (1975).
27. S. Lifson and A. Warshel. *J. Chem. Phys.,* **49,** 5116 (1968).
28. A. T. Hagler, E. Huler, and S. Lifson, *J. Am. Chem. Soc.,* **96,** 5319 (1974).
29. P.H. Berens and K.R. Wilson, *J. Comp. Chem.,* **4,** 313 (1983).
30. J.A. McCammon, P.G. Wolynes, and M. Karplus, *Biochemistry,* **18,** 927 (1979).
31. J. Hermans, H.J.C. Berendsen, W.F. van Gunsteren, and J.M. Postma, *Biopolymers,* **23,** 1513 (1984).
32. H. Margenau and N.R. Kestner, *Theory of Intermolecular Forces,* Pergamon Press, New York, 1969.
33. S.H.M. Nilar and S. Fraga, *J. Comp. Chem.,* **5,** 261 (1984).
34. S. Fraga, *J. Comp. Chem.,* **3,** 329 (1982).
35. E. Clementi and G. Corongiu, *Ions and Molecules in Solution,* N. Tanaka, H. Ohtaki, and R. Tamanushi, Eds., Studies in Physical and Theoretical Chemistry, Elsevier, Amsterdam, 1982, Vol. 27.
36. E. Clementi and G. Corongiu, *J. Biol. Phys.,* **11,** 33 (1983).
37. W. F. van Gunsteren, H. J. C. Berendsen, F. Colonna, D. Perahia, J.P. Hollenberg, and D. Lellouch, *J. Comp. Chem.,* **5,** 272 (1984).
38. J.M. Postma, H.J.C. Berendsen, and J.R. Haak, *Faraday Symp.,* **17,** 17/9 (1983).
39. Cray Research Inc., *Fortran (CFT) Reference Manual* (publication No. SR-0009), rev. 02, 1984, Part 3, pp. 2-1 to 2-16.
40. D.C. Rapaport and H.A. Scheraga, *Chem. Phys. Lett.,* **78,** 491 (1981).
41. W.F. van Gunsteren and M. Karplus, *Biochemistry,* **21,** 2259 (1982).
42. T.A. Andrea, W.C. Scope, and H.C. Andersen, *J. Chem. Phys.,* **79,** 4576 (1983).
43. M. Levitt and H. Meirovitch, *J. Mol. Biol.,* **168,** 595 (1983).
44. R.D. Swindoll and J.M. Haile, *J. Comp. Phys.,* **53,** 289 (1984).