Molecular Simulation, 1991, Vol. 6, pp. 121–142
Reprints available directly from the publisher
Photocopying permitted by license only

GENERALIZED VERLET ALGORITHM FOR EFFICIENT MOLECULAR DYNAMICS SIMULATIONS WITH LONG-RANGE INTERACTIONS

H. GRUBMÜLLER, H. HELLER, A. WINDEMUTH and K. SCHULTEN*

Beckman Institute and Department of Physics, University of Illinois, 405 N. Mathews Avenue, Urbana, IL 61801, USA

(*Received March 1990, accepted June 1990*)

For the purpose of molecular dynamics simulations of large biopolymers we have developed a new method to accelerate the calculation of long-range pair interactions (e.g. Coulomb interaction). The algorithm introduces distance classes to schedule updates of non-bonding interactions and to avoid unnecessary computations of interactions between particles which are far apart. To minimize the error caused by the updating schedule, the Verlet integration scheme has been modified. The results of the method are compared to those of other approximation schemes as well as to results obtained by numerical integration without approximation. For simulation of a protein with 12 637 atoms our approximation scheme yields a reduction of computer time by a factor of seven. The approximation suggested can be implemented on sequential as well as on parallel computers. We describe an implementation on a (Transputer-based) MIMD machine with a systolic ring architecture.

KEY WORDS: Molecular dynamics simulation, distance classes, parallel programming.

1. INTRODUCTION

Molecular dynamics simulations have become a widely used tool to investigate biological and chemical systems. Simulations which account for the long-range interactions between partial atomic charges consume long computing times. As a result only rather small biopolymers (of a few thousand atoms) and short simulation periods (of a few nanoseconds) are possible today. This is illustrated by the fact that the simulation of a large biopolymer with about 12 600 atoms over a period of 1 ns requires several weeks of CPU-time on the fastest current computer hardware.

To overcome the computational barrier we have developed and programmed a MIMD (*M*ultiple *I*nstructions *M*ultiple *D*ata) computer with a systolic ring architecture [1,2,3,4]. The computational nodes of this computer are currently based on T800 Transputers. Our computer allows extremely cost-effective molecular dynamics simulations. For a review on molecular dynamics on parallel computers see [5].

To simulate macromolecules faster, it is necessary, to accelerate the algorithm adopted for simulations. Since the evaluation of long-range pair interactions is the computationally most demanding part of a simulation, this task needs to be accelerated. Existing simulation programs adopt rather crude approximation schemes

*To whom correspondence should be sent.

for this purpose. One method often employed chooses a cut-off of pair interactions beyond a certain distance. This can lead to the prediction of artificial structural rearrangements within a macromolecule (see Section 5.2). In order to gain a substantial speedup while avoiding such artefacts, we developed a new algorithm to account for long-distance interactions. This algorithm judiciously schedules updates of non-bonding interactions avoiding unnecessarily frequent computations of interactions between particles which are far apart. The algorithm is closely related to the multiple-time scale (MTS) algorithm suggested in [6]. An excellent discussion of this algorithm and other issues regarding molecular dynamics algorithms can be found in [7]. The algorithm developed by us differs from the MTS algorithm in that we apply it to long-range Coulomb interactions rather than short-range Lennard-Jones potentials, in that we employ a large number of distance classes rather than two, in that we modify the algorithm to yield results as close as possible to the Verlet algorithm without distance classes, in that we apply the algorithm in the context of a simulation program for complex molecular systems, namely biological macromolecules like proteins and DNA, rather than Lenard-Jones liquids, and finally that we adopted the algorithm for a molecular dynamics program running on massive parallel computers.

After a brief review of conventional approximation schemes in Section 2, we will derive our new scheme in Section 3. In Section 4 the actual implementation of the algorithm on a parallel computer is described. In order to test the accuracy of our algorithm, we carried out simulations of a large biopolymer, which are described in Section 5.

2. CONVENTIONAL METHODS TO INCREASE EFFICIENCY

Computer simulation of a classical many-particle system is based on the solution of the Newtonian equations

$$m_k \frac{d^2}{dt^2}\mathbf{r}_k = \mathbf{F}_k(\mathbf{r}_1,\ldots,\mathbf{r}_{\mathcal{N}}), \tag{1}$$

where $\mathbf{r}_k$ denotes the position and m_k the mass of particle k, $k = 1,2,\ldots,\mathcal{N}$, $\mathcal{N}$ being the number of atoms in the molecule. $\mathbf{F}_k(\mathbf{r}_1,\ldots,\mathbf{r}_{\mathcal{N}}) = -\nabla_k E_{\text{pot}}(\mathbf{r}_1,\ldots,\mathbf{r}_{\mathcal{N}})$ represents the force acting on particle k.

A commonly used algorithm to integrate Equation (1) is the 'Verlet algorithm' [8]. According to this algorithm, the configuration $\mathbf{x}_{i+1} = (\mathbf{r}_1,\ldots,\mathbf{r}_{\mathcal{N}}) \in \mathcal{R}^{3\mathcal{N}}$ of a macromolecule (the index denotes the integration step) at instance $(i+1)\Delta t$ is

$$\mathbf{x}_{i+1} = 2\mathbf{x}_i - \mathbf{x}_{i-1} + (\Delta t)^2\,\mathbf{f}(\mathbf{x}_i), \tag{2}$$

where $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$ denote the configuations at time $i\Delta t$ and $(i-1)\Delta t$, respectively, and $\mathbf{f}(\mathbf{x}_i) = \left(\frac{\mathbf{F}_1}{\mathbf{m}_1},\ldots,\frac{\mathbf{F}_{\mathcal{N}}}{\mathbf{m}_{\mathcal{N}}}\right)$ denotes the accelerations[1] acting at time $i\Delta t$. Δt is the integration step size. This algorithm is exact to second order in Δt.

$\mathbf{f}(\mathbf{x}_i)$ needs to be computed every integration step. This vector, in general, will depend on the positions of all particles in the system. As a result, evaluation of $\mathbf{f}(\mathbf{x}_i)$ is the computationally most intensive part of a simulation. Futhermore, the computa-

[1]We will refer to this quantity as a force, even though it is strictly an acceleration.

tional effort increases with the square of the number of particles. It appears highly desirable to reduce the number of arithmetic operations in simulation calculations. Two approximation schemes which reduce the number of arithmetic operations will be described now.

2.1 Cut-Off Scheme

The most commonly used approximation cuts-off all pair interactions beyond a certain maximum distance R_{cut}. The cut-off is achieved by multiplying the force by a 'cut-off'-function, which depends only on the distances and reduces the force to zero in a continous differentiable, manner. A typical 'cut-off'-function assumes the constant value unity up to a distance R_{on}, then decreases to zero between R_{on} and R_{cut} and remains zero for all distances greater than R_{cut}. As a result, only interactions up to the distance R_{cut} have to be evaluated.

The appropriate choice of R_{cut} is a compromise between accuracy and computing time. On the one hand, the smaller R_{cut} is chosen, the faster the computation will be carried out. On the other hand, a large R_{cut} will reduce the error of the approximation.

The 'cut-off', although often employed, cannot be an appropriate approximation for molecular dynamics simulations since a neglect of the Coulomb interaction is believed to cause major rearrangements within the molecule. An improved approximation scheme involves ordering of the long-range interactions according to distance classes [9], as described below. A similar algorithm had been suggested in [6] for short-range forces in Lenard-Jones liquids.

2.2 Distance Class Schemes

A 'distance class' is defined as follows: Let $\{R_0 = 0, R_1,\ldots,R_n\}$ be a set of radii with $R_j < R_{j+1}$ for all $j = 0,1,\ldots,n-1$. Then the set of particles k with positions $\mathbf{r}_k$ satisfying the condition $R_j \leqslant |\mathbf{r}_i - \mathbf{r}_k| < R_{j+1}$ is called the *distance class j with respect to particle i.* The class scheme is illustrated in Figure 1.

The basic idea of the distance class algorithm relies on the observation that the change in time of pair interactions acting between two particles in general is smaller the further the particles are separated. Thus, it is possible to approximate the time development of pair interactions by a 'step function', i.e. the interactions are assumed to remain constant during a number of integration steps. For closely spaced particles the 'step function' extends only over a single instance in time, for particles separated further the 'step function' extends over several instances in time, the number of steps it spans increasing with the increasing separation between the particles. The 'step function' extending over j steps implies that the force has to be computed only once during j integration steps. For any given particle one can order all other atoms according to the distance classes introduced above. The interactions with atoms in the inner distance classes have to be computed more often than those of the outer classes. Assuming that the forces originating from the particles within distance class j are evaluated once every 2^j integration steps, an obvious choice of a distance class algorithm is of the form,

$$\mathbf{x}_{Ni+k+1} := 2\mathbf{x}_{Ni+k} - \mathbf{x}_{Ni+k-1} + (\Delta t)^2 \sum_{j=0}^{n} \mathbf{f}^{(j)}_{Ni+2^j[k/2^j]}, \tag{3}$$

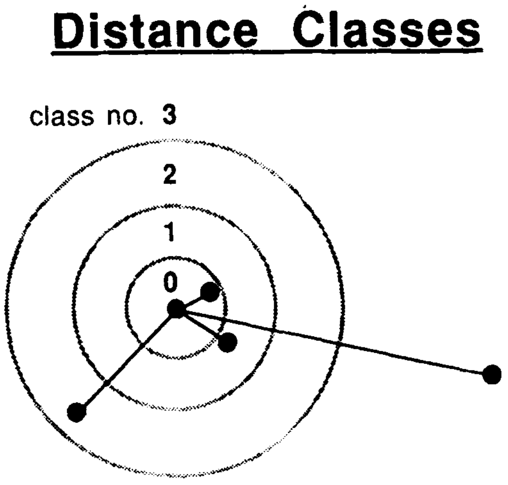

Figure 1 Scheme of the distance classes for a particle located at the center. The distance classes are numbered 0,1,2,3. One other particle and its distance to the center particle is shown for each distance class.

where k, denoting the integration step, runs from 0 to $N - 1$. The index i counts groups of integration steps, each group consisting of $N = 2^n$ single integration steps. One such group will be referred to as a 'macro-integration step'. The number of distance classes is $n + 1$. The force $\mathbf{f}^{(0)}$ is caused by the particles of the innermost (0-th) distance class, $\mathbf{f}^{(1)}$ by the particles of the class next to the innermost, and so on. Force $\mathbf{f}^{(n)}$ is caused by the particles of the outermost (n-th) distance class. The brackets ($\lfloor\ldots\rfloor$) denote the floor function, i.e. the positive integer less than or equal to the argument.

Table 1 Computation scheme for the distance class algorithm. Shown are the forces orginating from the different classes required during several integration steps. Boxed forces need to be computed; the others can be copied from previous integration steps.

integration			*distance class*		
step	*0*	*1*	*2*	*3*	...
0	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	...
1	$\boxed{1}$	0	0	0	...
2	$\boxed{2}$	$\boxed{2}$	0	0	...
3	$\boxed{3}$	2	0	0	...
4	$\boxed{4}$	$\boxed{4}$	$\boxed{4}$	0	...
5	$\boxed{5}$	4	4	0	...
6	$\boxed{6}$	$\boxed{6}$	4	0	...
7	$\boxed{7}$	6	4	0	...
8	$\boxed{8}$	$\boxed{8}$	$\boxed{8}$	$\boxed{8}$	...
9	$\boxed{9}$	8	8	8	...
10	$\boxed{10}$	$\boxed{10}$	8	8	...
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

The corresponding computation scheme is illustrated in Table 1, where the forces used for the actual integration step are listed for the different distance classes. The numbers represent the integration step, during which the corresponding force has been computed. The boxed numbers represent forces that need to be evaluated, while the non-boxed forces need not be computed, since they can be copied from previous integration steps.

As explained above, the forces of the outer distance classes, according to the distance class algorithm, are not updated within every integration step. Instead, the most recently evaluated forces are used as an approximation for the actual forces. As a result of this approximation the total energy of the system will not be constant. In the following chapter two methods to overcome this disadvantage will be presented. The first one (referred to as 'Verlet-I algorithm') is a generalization of the distance class algorithm and can be regarded as a first step towards the second algorithm, the 'Verlet-II algorithm'.

3. GENERALIZATION OF THE VERLET ALGORITHM

3.1 First Generalization

To devise a scheme which reduces the error resulting from applying the distance class algorithm (Equation (3)), we consider the special choice of distance classes $[R_i; R_{i+1}]$, as follows: $R_i := 0$ for all $i \leqslant n$ and $R_i := \infty$ for $i > n$. For this choice all particle pairs fall into distance class number n. Interactions are calculated every N-th integration step ($N = 2^n$). In this case, in which all pair interactions are treated identically, the algorithm should yield the same result as the conventional Verlet algorithm with an integration step $N\,\Delta t$.

However, 'Verlet equivalence' is not satisfied by the distance class algorithm defined by Equation (3). We want to rewrite the scheme by generalizing Equation (3) such that 'Verlet equivalence' results. Verlet equivalence appears to be desirable since it ascertains energy conservation and close agreement with predictions resulting from the Verlet algorithm. For this purpose we modify Equation (3) as follows

$$\mathbf{x}_{Ni+k+1} := 2\mathbf{x}_{Ni+k} - \mathbf{x}_{Ni+k-1} + (\Delta t)^2 \sum_{j=0}^{n} a^{(j)}_{k-2^j\lfloor k/2^j\rfloor}\, \mathbf{f}^{(j)}_{Ni+2^j\lfloor k/2^j\rfloor}\,. \tag{4}$$

Here k denotes the integration step, running from 0 to $N - 1$. The definitions of Equation (4) are the same as in Equation (3). The modification introduces weight factors $a_l^{(j)}$, which will be chosen to establish the Verlet equivalence. To determine these $a_l^{(j)}$ we define

$$\left.\begin{aligned} y_k &:= \mathbf{x}_{Ni+k} \\ b_k &:= (\Delta t)^2 \sum_{j=0}^{n} a^{(j)}_{k-2^j\lfloor k/2^j\rfloor}\, \mathbf{f}^{(j)}_{Ni+2^j\lfloor k/2^j\rfloor}\,. \end{aligned}\right\} \tag{5}$$

The N Equations (4) can then be written

$$y_{k+1} := 2y_k - y_{k-1} + b_k; \quad k = 0,\ldots,N-1. \tag{6}$$

In order to compare Equation (6) with the conventional Verlet algorithm Equation (2), for an integration step size of $N\,\Delta t$, one needs to express y_{2N} in terms of y_N and y_0. As will be shown, the resulting expression is of the form

$$y_{2N} := h(y_N, y_0, b_1, \ldots, b_{2N-1}) \,. \tag{7}$$

This expression will be used to determine the coefficients $a_l^{(j)}$. In order to guarantee the uniqueness of the solution we require $a_l^{(j)} \geqslant 0$. This restriction is reasonable since such weights conserve the direction of the corresponding force.

We will now determine expression h defined in Equation (7). Since there are $2N - 1$ unknowns, namely $y_1, \ldots, y_{N-1}, y_{N+1}, \ldots, y_{2N-1}, y_{2N}$, a system of $2N - 1$ equations is needed. For this purpose we use Equation (6) with k now running from 1 to $2N - 1$. This implies the use of the equations of *two* macro-integration steps. Since the first N equations are connected to the second N equations via an index transformation $k \rightarrow k + N$, one could argue that the extension is redundant. This is not the case, because the force terms b_l and b_{l+N} differ. It should be emphasized that neither y_0 nor y_N are considered unkown, since they are parameters of the function h. Equation (6) can be written

$$\begin{bmatrix}
2 & -1 & & & & & & & & \\
-1 & 2 & -1 & & & & & & & \\
 & -1 & \ddots & \ddots & & & 0 & & & \\
 & & \ddots & 2 & -1 & & & & & \\
 & & & -1 & 2 & & & & & \\
 & & & & -1 & -1 & & & & \\
 & & & & & 2 & -1 & & & \\
 & 0 & & & & -1 & \ddots & \ddots & & \\
 & & & & & & \ddots & 2 & -1 & \\
 & & & & & & & -1 & 2 & -1
\end{bmatrix}
\begin{bmatrix}
y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \\ y_{N+1} \\ y_{N+2} \\ y_{N+3} \\ \vdots \\ y_{2N}
\end{bmatrix}
=
\begin{bmatrix}
y_0 - b_1 \\ -b_2 \\ \vdots \\ -b_{N-2} \\ y_N - b_{N-1} \\ -2y_N - b_N \\ y_N - b_{N+1} \\ -b_{N+2} \\ \vdots \\ -b_{2N-1}
\end{bmatrix} . \tag{8}$$

By eliminating all unknowns except for y_{2N}, one obtains

$$y_{2N} = 2y_N - y_0 + \sum_{k=0}^{N-2} (k + 1) b_{k+1} + \sum_{k=N}^{2N-2} (2N - k - 1)\, b_{k+1} + N b_n \,. \tag{9}$$

To establish Verlet equivalence Equation (9) has to be compared to a step in the Verlet algorithm for a step size $N \cdot \Delta t$. Since all forces originate from distance class n it holds that

$$\mathbf{f}^{(j)} = 0 \text{ for all } j \neq n \,. \tag{10}$$

The definitions (5) yield

$$\left.\begin{aligned}
\mathbf{x}_{N(i+2)} &= 2\mathbf{x}_{N(i+1)} - \mathbf{x}_{Ni} + (\Delta t)^2\,(A\mathbf{f}^{(n)}_{Ni} + (B + N)\,\mathbf{f}^{(n)}_{N(i+1)})\\
A : &= \sum_{k=1}^{N-1} k a^{(n)}_k\\
B : &= \sum_{k=1}^{N-1} (2N - k) a^{(n)}_k\,.
\end{aligned}\right\} \qquad (11)$$

Since the conventional Verlet algorithm contains only forces $\mathbf{f}^{(n)}_{N(i+1)}$, A has to vanish. With $a^{(n)}_k \geqslant 0$ we obtain

$$a^{(n)}_k = 0 \text{ for all } k = 1,2,\ldots,N - 1\,. \qquad (12)$$

Therefore, B vanishes, too, and Equation (11) transforms to

$$\mathbf{x}_{N(i+2)} = 2\mathbf{x}_{N(i+1)} - \mathbf{x}_{Ni} + (N\Delta t)^2 \cdot \frac{1}{N}\, a^{(n)}_0\, \mathbf{f}^{(n)}_{N(i+1)}\,. \qquad (13)$$

Obviously, Equation (13) is equivalent to the conventional Verlet algorithm, if and only if $a^{(n)}_0 = N = 2^n$.

Given a number of distance classes, this procedure can be applied recursively (starting with the innermost distance class) to all classes. This yields $a^{(j)}_{k-2^j\lfloor k/2^j\rfloor} = 2_j\, \delta_{k,2^j\lfloor k/2^j\rfloor}$ and Equation (4) transforms to

$$\mathbf{x}_{Ni+k+1} := 2\mathbf{x}_{Ni+k} - \mathbf{x}_{Ni+k-1} + (\Delta t)^2 \sum_{j=0}^{n} 2^j\, \delta_{k,2^j\lfloor k/2^j\rfloor}\, \mathbf{f}^{(j)}_{Ni+2^j\lfloor k/2^j\rfloor} \qquad (14)$$

$$k = 0,\ldots,N - 1\,.$$

This scheme will be referred to as Verlet-I algorithm.

3.2 Discussion of the Verlet-I algorithm

The Verlet-I algorithm (14) is constructed to be equivalent to the Verlet algorithm, if only *one* distance class is occupied, i.e. if the distances between all pairs of particles lie within *one* distance class. In this case, the additional error of the distance class algorithm (compared with the Verlet algorithm) could be avoided. However, this is a case that never occurs for realistic simulations. It was considered only to find conditions for a unique description of the algorithm. We will now consider the properties of the Verlet-I algorithm in the case of several distance classes being occupied.

The physical interpretation of the Verlet-I algorithm is the following: In using the conventional distance class algorithm of Equation (3), during one macro-integration step one transfers the momentum $\mathbf{F}\cdot\Delta t$ onto a given particle at N instances of time. The difference between these instances of time is Δt. In contrast to that, the Verlet-I algorithm transfers a momentum N times larger, i.e. $N \cdot \mathbf{F} \cdot \Delta t$, only once. If only the outermost distance class is occupied, there will be no difference to the conventional Verlet algorithm (with an integration step of $N\Delta t$), since, in this case, the physical interpretation is the same.

In order to understand qualitatively why the results turn out to be worse when not only one, but several distance classes are occupied, we discuss the following example: Consider a system of three bodies, two of which (A and B) are located close together compared to the distance between each of those two particles and the third one (C). Assuming Coulomb interaction, the charge of particle A shall be positive, whereas B carries negative charge. In integrating the Newtonian equations of this system, one will usually choose an integration step size Δt of approximately two orders of magnitude less than the period of the fastest oscillation in the system, which in this case is the rotation of particles A and B around each other. Let us assume that $n + 1$ distance classes are used, where only class 0 and class n are occupied. Now the Verlet-I algorithm will apply the force caused by particle C acting on both, particle A and B, only every 2^n-th integration step. However, this time interval might be near the resonance frequency of the system. Since the charges of the first two particles are opposite, the forces act in opposite directions, too. As a result, conservation of energy will be violated[2]. In the worst case, an artificial resonance can occur. Thus, the Verlet-I algorithm can not be used for more than approximately 3 or 4 distance classes. However, if the forces are purely attractive, as for example in gravitational systems, the situation is more favorable, because the forces caused by particle C on particles A and B point in the same direction.

3.3 Second Generalization

To avoid the disadvantage of the Verlet-I algorithm described above, we introduce a second generalization of the Verlet algorithm, the 'Verlet-II algorithm'. The basic idea is the following:

As shown in Equation (11), the demand that $A\mathbf{f}_{Ni}^{(n)} = 0$, i.e. all forces of the previous macro-integration step vanish, is a sufficient condition for a unique determination of the weight factors $a_l^{(j)}$. To gain additional degrees of freedom for those weight factors, the Verlet-II algorithm includes the forces of the *previous* macro-integration step in addition to the forces of the *actual* macro-integration step. This inclusion actually corresponds to an interpolation of forces. Such interpolation had been also suggested in [6,7]. These authors evaluated, for the purpose of such interpolation, time derivatives of the forces. In the algorithm detailed below the interpolation is based on a knowledge of forces evaluated at different time instances. Also the interpolation is described by weight factors which optimize agreement with a linear interpolation scheme as well as agreement with the conventional Verlet algorithm.

The algorithm can be written in the following form

$$\mathbf{x}_{Ni+k+1} := 2\mathbf{x}_{Ni+k} - \mathbf{x}_{Ni+k-1} + (\Delta t)^2 \sum_{j=0}^{n} \left(a^{(j)}_{k-2^j\lfloor k/2^j\rfloor}\, \mathbf{f}^{(j)}_{Ni+2^j\lfloor k/2^j\rfloor} + b^{(j)}_{k-2^j\lfloor k/2^j\rfloor}\, \mathbf{f}^{(j)}_{Ni+2^j\lfloor k/2^j\rfloor-2^j}\right), \tag{15}$$

with the definitions being identical to the ones used in Equations (3) and (4). The only difference to Equation (4) is the addition of the new force coefficients $b_l^{(j)}$. These coefficients belong to the force of the previous macro-integration step and have to be determined along with the weights $a_l^{(j)}$. In contrast to the Verlet-I algorithm, there will

[2]As would be the case when using the conventional Verlet algorithm with an integration step size of 2^n Δt.

be no restrictions on the sign of neither $a_l^{(j)}$ nor $b_l^{(j)}$. Nevertheless, the $a_l^{(j)}$ turn out to assume always positive values.

We like algorithm (15) to satisfy Verlet equivalence as well. For a many-body system, in which there are only forces belonging to distance class n, Equation (15) shall represent a Verlet algorithm with an integration step size $N\Delta t$.

We define abbreviations in analogy to Equation (5)

$$\left.\begin{aligned} y_k &:= \mathbf{x}_{Ni+k} \\ b_k &:= (\Delta t)^2 \sum_{j=0}^{n} \left(a^{(j)}_{k-2^j[k/2^j]}\, \mathbf{f}^{(j)}_{Ni+2^j[k/2^j]} + b^{(j)}_{k-2^j[k/2^j]}\, \mathbf{f}^{(j)}_{Ni+2^j[k/2^j]-2^j}\right). \end{aligned}\right\} \quad (16)$$

With these definitions Equation (15) transforms into Equation (6), i.e. the solution Equation (9) of Section 3.1, can be used. Resubstitution of the abbreviations together with the restriction $\mathbf{f}^{(j)} = 0$ for all $j \neq n$ yields

$$\left.\begin{aligned} \mathbf{x}_{N(i+2)} &= 2\mathbf{x}_{N(i+1)} - \mathbf{x}_{Ni} + (\Delta t)^2 \left(A\mathbf{f}^{(n)}_{N(i+1)} + B\mathbf{f}^{(n)}_{Ni} + C\mathbf{f}^{(n)}_{N(i-1)}\right) \\ A &:= \sum_{k=0}^{N-1} (N-k)\, a_k^{(n)} \\ B &:= \sum_{k=1}^{N-1} k a_k^{(n)} + \sum_{k=0}^{N-1} (N-k)\, b_k^{(n)} \\ C &:= \sum_{k=1}^{N-1} k b_k^{(n)} \ . \end{aligned}\right\} \quad (17)$$

Since, again, the force terms in Equation (17) have to be equal to $N^2 \cdot \mathbf{f}^{(n)}_{N(i+1)}$, the coefficients must satisfy

$$\left.\begin{aligned} &\sum_{k=0}^{N-1} (N-k)\, a_k^{(n)} = N^2 \\ &\sum_{k=1}^{N-1} k a_k^{(n)} + \sum_{k=0}^{N-1} (N-k)\, b_k^{(n)} = 0 \\ &\sum_{k=1}^{N-1} k b_k^{(n)} = 0 \ . \end{aligned}\right\} \quad (18)$$

In contrast to the derivation of the Verlet-I algorithm, Equations (18) do not determine a unique solution for the weight factors $a_k^{(n)}$ and $b_k^{(n)}$. If this were the case, the introduction of the forces $\mathbf{f}^{(n)}_{N(i-1)}$ would yield no improvement. Now it is possible to optimize the algorithm by introducing further conditions on the force coefficients. The conditions have to be chosen in a way that guarantees evenly distributed force coefficients.

As a first step, one might wish to make use of the fact that for each integration step forces of *two* previous integration steps are known, namely $\mathbf{f}^{(n)}_{N(i-1)}$ and $\mathbf{f}^{(n)}_{Ni}$. Based on these two forces, one could find an approximation for the actual force $\mathbf{f}^{(n)}_{Ni+k}$ by the linear exptrapolation

$$\mathbf{f}^{(n)}_{Ni+k} \approx \mathbf{f}^{(n)}_{Ni} + \frac{k}{N}\left(\mathbf{f}^{(n)}_{Ni} - \mathbf{f}^{(n)}_{N(i-1)}\right), \ k = 0,\ldots,N-1 \ . \quad (19)$$

The force coefficients would then become:

$$a_k^{(n)} = 1 + \frac{k}{N} \text{ and } b_k^{(n)} = -\frac{k}{N}\,. \tag{20}$$

Unfortunately, these force coefficients do not satisfy equations (18).

However, our consideration leads us to a suitable optimization criterion, namely the weights should

(i) satisfy equation (18) and
(ii) differ as little as possible from the coefficients obtained by the linear extrapolation(19).

Condition (i) is the Verlet equivalence and was already explained above. Condition (ii) is new; it will be realized by minimization of

$$Q = \sum_{k=0}^{N-1} (a_k^{(n)} - \tilde{a}_k^{(n)})^2 + \sum_{k=0}^{N-1} (b_k^{(n)} - \tilde{b}_k^{(n)})^2. \tag{21}$$

The solution, i.e. the force coefficients $a_k^{(n)}$ and $b_k^{(n)}$, which minimizes (21) and satisfies Equation (18), can be found by the method of Lagrangian multipliers. The result is

$$\left.\begin{aligned} a_k^{(j)} &= 3\cdot 2^{j+1}\,\frac{(2^{2j} - 2^j + 1) - k(2^j - 2)}{(2^j + 1)\,(2^{2j+1} + 1)} \\ b_k^{(j)} &= \frac{2(-\,2^{2j+1} + 3\cdot 2^j - 1) + 6k(2^j - 1)}{2^{2j+1} + 1}\,. \end{aligned}\right\} \tag{22}$$

Substitution of the force coefficients $a_k^{(n)}$ and $b_k^{(n)}$ into Equation (15) yields the Verlet-II algorithm.

The aim of the second generalization was to avoid force coefficients which are unevenly distributed. It can be easily shown that all force coefficients lie within the interval $[-2; +3]$, in contrast to the Verlet-I algorithm, for which the first coefficient grows as N^2, while all others vanish. As a result, according to the Verlet-II algorithm, the forces acting on a given particle are distributed evenly over all integration steps within one macro-integration step. For this reason, it can be expected that the Verlet-II algorithm avoids the disadvantages of the Verlet-I algorithm, as described above.

However, one has to pay for the expected increase in numerical accuracy: In order to simulate a given number of particles using the Verlet-II algorithm, one has to keep in memory *two* forces per particle and distance class which is about twice the amount of memory required for the Verlet-I algorithm. This reduces the maximum number of particles that can be simulated on a given computer system by half.

4. IMPLEMENTATION OF THE VERLET-II ALGORITHM

4.1 Initialization Phase of the Algorithm

The initialization phase of the Verlet-II algorithm is more complex than that of the conventional Verlet algorithm. To start a simulation of a many-particle system one chooses positions $\mathbf{x}_0 \in \mathscr{R}^{3,1}$ (often determined experimentally by X-ray diffraction or by NMR) and velocities $\mathbf{v}_0 \in \mathscr{R}^{3,1}$ of all particles of the system. Since the Verlet algorithm requires knowledge of particle positions of *two* successive instances of the discretized time one needs to evaluate a second set of coordiantes $\mathbf{x}_1$ from $\mathbf{x}_0$ and velocities $\mathbf{v}_0$

$$\mathbf{x}_1 := \mathbf{x}_0 + \Delta t\, \mathbf{v}_0 + \frac{1}{2} (\Delta t)^2 \cdot \mathbf{f}(\mathbf{x}_0) \ , \tag{23}$$

where Δt denotes the integration step size. After this initialization step the algorithm works as explained above: For the coordinates $\mathbf{x}_1$, the forces $\mathbf{f}_1$ can be calculated and these forces, in turn, are then used to carry out an integration step (2), yielding another set of new coordinates, and so on.

Let us now discuss the initialization of the Verlet-II algorithm. In the following, we will call the forces connected to the weights $a_i^{(j)}$ the 'new' forces (they stem from the actual macro-integration step), and the forces connected to the weights $b_i^{(j)}$ the 'old' forces (they stem from the previous macro-integration step). In contrast to the Verlet algorithm Equation (3), the Verlet-II algorithm, according to Equation (15), implies the use of *two* sets of forces *per distance class*, and, therefore, a single initialization step cannot suffice. Actually, for n distance classes, 'old' forces, calculated up to 2^n integration steps before the current integration step are required, but are not available at the beginning of the simulation.

One solution to this problem[3] is to inhibit the use of all outer distance classes during initialization, thus avoiding the need for forces that cannot yet be provided. This is done by introducing the following rule during initialization: Each pair of particles that would normally lie within 'inhibited' distance classes, is treated as if the pair was contained in the outermost 'allowed' distance class. Following this rule, one finds that after each integration step the necessary forces of the previous integration steps are available. As a result, from time to time, the actual innermost 'inhibited' distance class can be made 'allowed'. After all classes have become 'allowed', the initialization is completed. Consequently, the initialization scheme falls into n groups of steps, each group consisting of twice as many steps than the previous one.

Table 2 demonstrates the initialization scheme for the case of four distance classes. Shown are the indices of the 'old' and the 'new' forces (belonging to the four different distance classes) that are used during the actual integration step. Only the boxed forces have to be computed; the others can be taken from previous steps. The substitution of 'old' forces with 'new' ones is represented by an arrow ($\searrow$). Exclusion of 'inhibited' distance classes is accomplished by using the same force coefficients $a_i^{(j_0)}$ and $b_i^{(j_0)}$ for all forces $\mathbf{f}_i^{(j_0+1)},\ldots,\mathbf{f}_i^{(n-1)}$ belonging to the 'inhibited' classes $j_0 + 1,\ldots,j_{n-1}$ during integration step i. Table 2 illustrates this by grouping appropriate forces with brackets ((...)).

Let us explain the example in Table 2 in greater detail: The first step (no. 0) is equivalent to the one of the conventional Verlet algorithm. Using the initial conditions (positions and velocities), one computes a second set of coordinates (Equation (23)). In addition to that, forces, which will be used later, are stored as 'old' forces. Having evaluated the 'new' forces, integration step 1 can be carried out, with force coefficients chosen as if all pairs were located within the innermost (0-th) distance classs.

[3]A possibly simpler solution of the problem is to carry out conventional Verlet-steps in the beginning until all necessary forces can be provided, which will be the case after 2^n steps. However, as this approach is not very efficient, it was not followed by us.

Table 2 Computation scheme for the initialization of the Verlet-II algorithm. As an example, four distance classes were assumed. Shown are the indices of the 'old' and 'new' forces belonging to the four distance classes, which are used during the actual integration step. Boxed forces have to be computed; substitution of 'old' forces with 'new' ones is represented by an arrow (↘). Distance classes which are grouped together by brackets ((. . .)) are treated as one class as explained in the text.

	distance classes							
	0		*1*		*2*		*3*	
Step	*new*	*old*	*new*	*old*	*new*	*old*	*new*	*old*
0	($\boxed{0}$		$\boxed{0}$		$\boxed{0}$		$\boxed{0}$	)
		↘		↘		↘		↘
1	($\boxed{1}$	0	$\boxed{1}$	0	$\boxed{1}$	0	$\boxed{1}$	0)
		↘						
2	$\boxed{2}$	1	($\boxed{2}$	0	$\boxed{2}$	0	$\boxed{2}$	0)
		↘						
3	$\boxed{3}$	2	(2	0	2	0	2	0)
		↘		↘				
4	$\boxed{4}$	3	$\boxed{4}$	2	($\boxed{4}$	0	$\boxed{4}$	0)
		↘						
5	$\boxed{5}$	4	4	3	(4	0	4	0)
		↘		↘				
6	$\boxed{6}$	5	$\boxed{6}$	4	(4	0	4	0)
		↘						
7	$\boxed{7}$	6	6	4	(4	0	4	0)
		↘		↘		↘		
8	$\boxed{8}$	7	$\boxed{8}$	6	$\boxed{8}$	4	$\boxed{8}$	0
		↘						
9	$\boxed{9}$	8	8	6	8	4	8	0
		↘		↘				
10	$\boxed{10}$	9	$\boxed{10}$	8	8	4	$\boxed{8}$	0
		↘						
11	$\boxed{11}$	10	10	8	8	4	8	0
		↘		↘		↘		
12	$\boxed{12}$	11	$\boxed{12}$	10	$\boxed{12}$	8	8	0
		↘						
13	$\boxed{13}$	12	12	10	12	8	8	0
		↘		↘				
14	$\boxed{14}$	13	$\boxed{14}$	12	12	8	8	0
		↘						
15	$\boxed{15}$	14	14	12	12	8	8	0
		↘		↘		↘		↘
16	$\boxed{16}$	15	$\boxed{16}$	14	$\boxed{16}$	12	$\boxed{16}$	8
		↘						
17	$\boxed{17}$	16	16	14	16	12	16	8
		↘		↘				
⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮

Integration step 2 marks the beginning of the second group of steps: At this moment, all forces required for using distance class 1 have been computed (namely $\mathbf{f}_0$ and $\mathbf{f}_2$). It is then possible to separate the innermost distance class from the rest of the pairs, which in turn is now defined to belong to distance class 1. As a result of this, during integration step 3 computation of forces of the outer classes is not necessary.

The third group starts with integration step 4. From now on there are three distance classes, and both, class 0 and class 1, can be treated in the usual manner.

In the example, the initialization ends with integration step 7. Step 8 is the first integration step for which the 'oldest' force required by the Verlet-II algorithm, $\mathbf{f}_0^{(4)}$, had been stored and the Verlet-II algorithm proper sets in. Notice that $\mathbf{f}_0^{(4)}$ is the only force computed during step 0 which has not yet been overwritten by some 'newer' force. The algorithm outlined is summarized in Table 2 which shows one macro-integration step as well as two successive integration steps in addition to the initialization phase.

4.2 Pair- and Bit-Lists

In order to take advantage of the principle of distance classes, an efficient algorithm to identify all pairs of particles belonging to a given distance class has to be provided. One can think of three possibilities (among others) to perform this task:

1) The most simple is to explicitly compute the distance of every pair of particles for every integration step. This will be necessary if there are stringent memory limits on the computer system used. However, in implementing this method, one gains only an increase of efficiency by a constant factor; the numerical effort will still increase proportionally to the square of the number of particles, due to the number of distances that have to be computed.

2) The second possibility is to provide a so-called 'bit-list', which is any array of bit-fields; one field for each pair of particles (see also discussion of this issue in [7]). Asuming $n + 1 = 2^m$ distance classes, every field consists of m single bits. Those bits provide sufficient information for each pair of particles to know which distance class it belongs to, and, therefore, whether the corresponding force has to be computed or not. Of course, this bit-list has to be updated every few integration steps, e.g. after every macro-integration step, to keep track of movements of particles. But, as there is still a constant number of operations (namely reading the bit-list) to be performed for *each* pair of particles–even if the corresponding force does not have to be computed–, the numerical effort will still increase with the square of the number of particles as in method 1).

3) The method which allows to avoid this disadvantage uses 'pair-lists': For each particle and each distance class there is a list of all other particles belonging to that class with respect to the given particle (see also discussion of this issue in [7]). In performing the Verlet-II algorithm, only the listed particles are considered, which results in a linear increase of computational effort in the limit of a large number of particles. However, this method requires a tremendous amount of computer memory: For the simulation of a molecule of about 12 000 atoms a memory of 150 MBytes would be necessary.

One can see that the efficiency increases with the amount of memory required. Thus a method should be choosen that makes use of the available memory as much as possible to maximize efficiency. Since the most efficient method of pair-lists in general cannot be implemented due to a lack of memory, the method of choice might be the method of bit-lists. In our application we choose 8 distance classes, requiring a coding by three bits per particle pair. In this case one macro-integration step lasts 128 integration steps. Since all forces and all distances have to be computed according to the Verlet-II algorithm once every macro-integration step, the bit-lists are updated every 128 integration steps as well.

Actually an optimal choice is to use a combination of both, bit-lists and pair-lists. The basic idea is the following: According to the Verlet-II algorithm, the forces belonging to the innermost distance classes have to be computed most often, the frequency of computation decreasing for classes with larger distances. The corresponding fraction of integration steps is 50 percent for the innermost distance class, 75 percent for both, the innermost and the second class, 87.5 percent for the first three classes, etc. Despite the rare use of the outer distance classes, in using only the method of bit-lists, the whole list has to be searched through without doing any force computation. Since usually only few particles belong to the innermost distance classes, it is a good idea to make use of pair-lists for these classes, in addition to the bit-lists. These pair-lists will then be used instead of the bit-list for every integration step that doesn't require evaluation of the forces of the outer distance classes. The number of classes for which pair-lists are kept will be determined by the available amount of memory. In this way one can adjust the efficiency according to the available memory.

In our case of about 12 000 particles, only one pair-list (for the innermost distance class) was implemented. Although this required only a small additional amount of memory, namely 1.3 percent, the increase in efficiency was about 25 percent.

4.3 Implementation on a Parallel Computer

For our simulation we used a parallel computer (MIMD-*M*ultiple *I*nstructions, *M*ultiple *D*ata) with 12 nodes (so called 'Transputers' from INMOS), each containing its own memory. The Transputers are connected in a systolic ring[1,10]. The basic idea of the parallelization is the following [2]: Each node of the computer keeps one twelfth of the coordinates of all particles (called the 'own' particles) in memory. In addition, a second set of coordinates (called the 'external' ones) is kept in memory at every node. During computation of one integration step, this second set of coordinates travels around the ring of nodes in such a way that every node is able to compute the interactions of its 'own' particles with all the other particles in the system. This scheme allows the total amount of necessary computations to be evenly distributed among the computational nodes.

In using this scheme, one cannot make use of the Newtonian law 'actio = reactio' and, thereby, cut into half the amount of computations of pair interactions. As it turns out, in order to implement this law, one has to send also the *forces* along the ring. Although, in the case of the Verlet-II algorithm, the forces have to be computed only every 2^j-th integration step (j being the number of the corresponding distance class), one finds that (according to Equation (15)) for each distance class the two most recently computed forces have to be known within every integration step. As a result, in using $n + 1$ distance classes, a total of $2(n + 1)$ force vectors have to be kept in memory for each particle. Furthermore, these $2(n + 1)$ force vectors have to be sent along the ring, such that the total amount of data to be transferred between nodes increases by a factor of about $(n + 1)$, compared with the conventional Verlet algorithm. Taking into account the fact that the average computation time for one integration step decreases, we observed in our simulation an increase of the communication rate by a factor of roughly 60.

5. ACCURACY OF THE VERLET-II ALGORITHM

In order to test the accuracy of the Verlet-II algorithm, we performed molecular dynamics simulations on the photosynthetic reaction center of *Rhodopseudomonas viridis* starting from the X-ray structure of Deisenhofer and Michel [11]. This molecule consists of 12 637 atoms[4]. We actually started our simulations with a relaxed structure, as described in [1].

The simulations of the reaction center were carried out employing the energy function of the molecular dynamics program CHARMM [12] including the following energy contributions: electrostatic energy, van der Waals energy, bond energy, angle energy, dihedral energy and improper energy. H-bonds have been taken into account by a so-called implicit computation. The first two energy contributions (electrostatic and van der Waals) are long-range interactions. For this reason, the number of atom pairs contributing to the total electrostatic energy increases proportionally to the number of atoms squared, and so does the computational effort when using the conventional Verlet algorithm. In contrast to that, the other energy contributions (bond, angle, dihedral and improper energy) represent short-range interactions, so the computational effort increases only linearly with the number of atoms. Thus, a very large fraction of the total computation time has to be spent in evaluating non-bonding forces. For this reason we used the Verlet-II algorithm for evaluation of Coulomb interactions, but not for other interactions[5]. However, the short-range contributions had to be fitted into the distance class scheme, too, and this was simply done by defining the corresponding forces to be within the innermost distance class. It should be emphasized that this possibility is a feature of the Verlet-II algorithm which makes it very well suited for molecular dynamics simulations on parallel computers.

5.1 Parameters of the Simulations

All simulations were carried out using an integration step size of 0.25 fs, which is relatively short compared with other simulations [9,13]. This short integration time step was chosen to accurately reproduce the fastest vibrations in the biomolecular system. The choice is not connected to the Verlet-II algorithm. In fact, if only the Coulomb forces were acting on the atoms a much longer integration time step could be chosen. At least 600 integration steps were performed, i.e. the simulation lasted 150 fs. A second set of simulations lasting 1.5 ps (6000 integration steps) was carried out to compare the long time behavior of the Verlet and Verlet-II algorithms.

Four different algorithms were compared, using identical conditions for the different simulations. The algorithms are:

- Computation of all interactions within every integration step, according to the conventional Verlet algorithm (referred to as 'all interactions' or 'Verlet').
- Implementation of a 'cut-off', as described in Section 2.1. This algorithm is also used by the program XPLOR [14], from which we adapted the cut-off function. The

[4]Some groups were collectively treated as compound atoms.

[5]The van der Waals interaction was included, since it is formally very similar to the electrostatic interaction, although it would as well be possible to simply cut it off because of the rapid r^{-6} decrease of this interaction.

cut-off radii were chosen as proposed in [13]: 11.0 Å for the electrostatic interaction and 9.5 Å as well as 10.0 Å for the van der Waals interaction[6].

- The conventional distance class algorithm, as explained in [9]. This algorithm will be referred to as 'distance class'.
- The Verlet-II algorithm, as explained above.

Since the radii of the distance classes determine the accuracy of the distance class algorithms, we would like to comment on the choice of appropriate radii. No specific rule for an optimal choice can be given; rather the choice will depend on the size and the shape of the molecule to be simulated as well as on the specific charge distribution within the molecule. In order to judge the quality of a given set of distance class radii, we used the following criteria, which are presented in order of decreasing importance:

1) Criteria of accuracy

(a)The most important criterion is *appropriate accuracy.* Roughly speaking, the error of the Verlet algorithm with respect to a given particle is proportional to the time derivative of the total force acting on the particle times the integration step size. An estimate for an upper limit for this derivative can be derived from the total force itself, acting on the particle. Since the force contributions of the outer distance classes are less than those of the inner classes, one can choose different integration step sizes for different distance classes, as done by the distance class algorithms. Appropriate accuracy, then, means that the error of the outer distance classes (estimated by the force contribution times the integration step size) should not exceed the error of the innermost distance class.

(b)As one can conclude from (a) it is reasonable to chose the radii such that errors for the different distance classes are within the same order of magnitude.

2) Criteria of efficiency

(a)Since the interactions of the inner distance classes are computed most often, the corresponding radii should be as small as possible.

(b)For reasons of economical memory usage, the radius denoting the beginning of the outermost distance class should be less than the maximum size of the molecule; that is, 'empty' classes should be avoided.

Using these criteria, we tested several different sets of distance class radii. The set with the best performance, which was then used for both simulations, 'distance class' and 'Verlet-II', is given in Table 3. Since the distance class lists are updated only once per macro-integration step, some atoms might get closer to each other than they seem to be according to the distance class lists. In order to compensate for these atomic motions, all radii were provided with an offset, which is already included in Table 3.

5.2 Results of the Simulations

In this Section we will present results of simulations of the complete photosynthetic reaction center *Rhodopseudomonas viridis* over two time periods, a time period of

[6]The switching function for the van der Waals interaction requires two parameters.

Table 3 Radii of the distance classes used for the simulations.

class	*radii [Å]*		
0	0	...	5
1	5	...	8
2	8	...	14
3	14	...	23
4	23	...	35
5	35	...	50
6	50	...	68
7	68	...	∞

150 fs and a time period of 1.5 ps. The simulations were carried out on a Transputer-based parallel computer using a molecular dynamics program in Occam II, both developed by us [1]. Over the brief (150 fs) period three simulations were performed, a simulation according to the conventional Verlet scheme including all interactions, the same simulation assuming a cut-off of long-range interactions with a cut-off radius of 10 Å, and a simulation according to the Verlet II algorithm described above. Over the longer (1.5 ps) period two simulations were carried out, a simulation according to

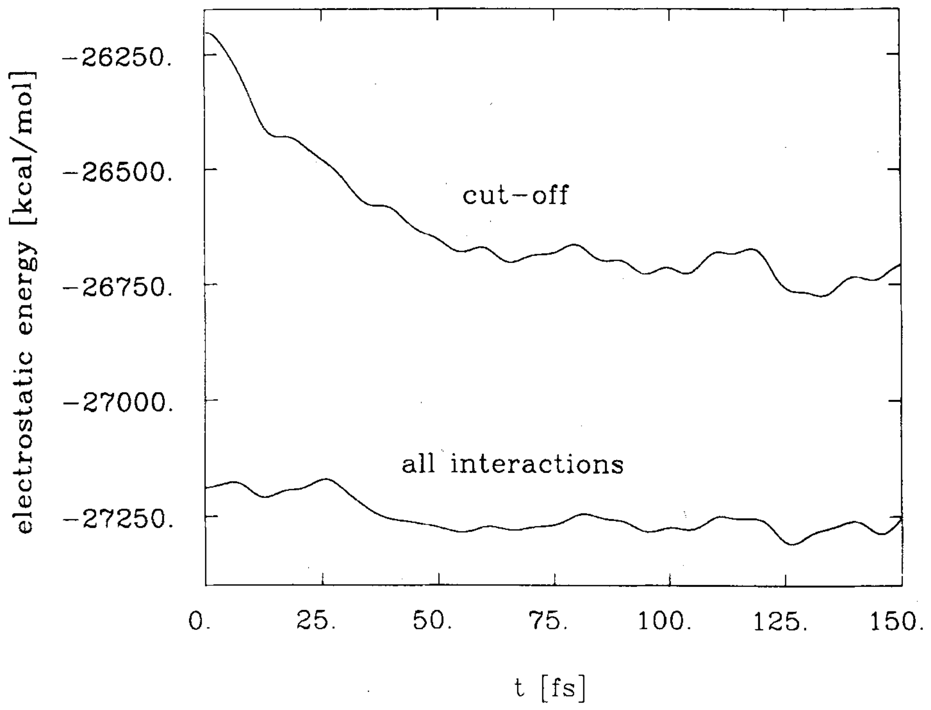

Figure 2 Comparison of the time development of the electrostatic energy resutling from a Verlet-type simulation assuming a cut-off of 11.0 Å for the electrostatic energy and a cut-off of 9.5 Å as well as 10.0 Å for the van der Waals interaction as described in Sections 2.1, 5.1 (upper curve) and from a Verlet-type simulation including all interactions (lower curve). The same equilibrated starting structure of *Rhodopseudomonas viridis* and the same time step of 0.25 fs have been used for both simulations.

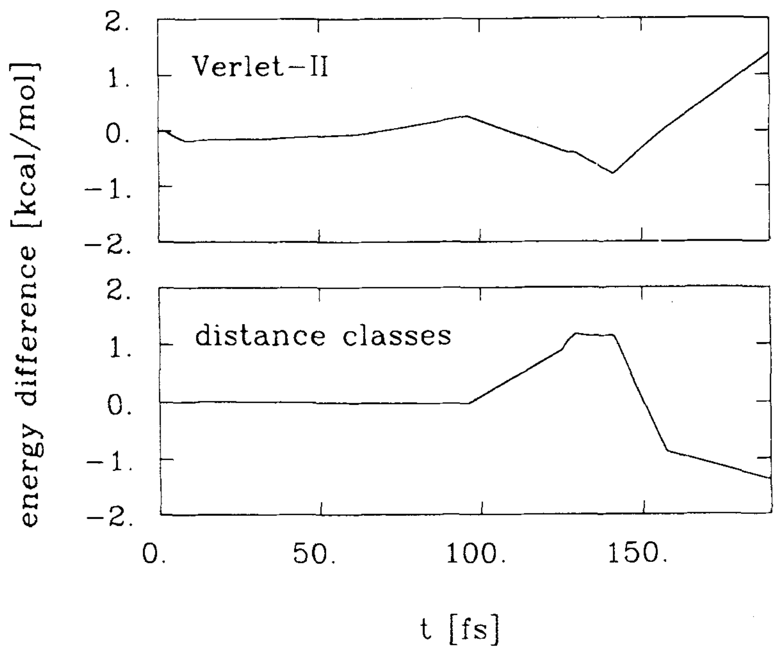

Figure 3 Time development of the electrostatic energy resulting from a Verlet-II type simulation and a distance class simulation (see text). In the upper figure are shown the *differences* of the energy between a Verlet-II type simulation and a simulation including all interactions. In the lower figure are shown the *differences* of the energy between a distance class type simulation and a simulation including all interactions. The same starting structure and time step as for Figure 2 have been used.

the conventional Verlet integration scheme and a simulation according to the Verlet II scheme. In order to compare the accuracy of the different schemes we consider the time-development of several observables, the total energy, which should remain constant, the total electrostatic energy which should provide a sensitive measure in how far the approximation of the Coulomb forces in the Verlet II scheme is satisfactory, the temperature which monitores the mean kinetic energy of atoms and, finally, the deviation from the X-ray structure of the average structures resulting from the simulations.

We first want to demonstrate that the cut-off of long-range non-bonding forces leads to unsatisfactory deviations in the total electrostatic forces of a system and induces a drift of the system. For this purpose we present in Figure 2 the total electrostatic energy resulting from conventional Verlet-type simulations, one including all non-bonded forces, the other assuming a cut-off at a radius of 10 Å. The results in Figure 2 show an initial energy difference of about 1,000 kcal/mol between the two simulations, resulting from the neglect of long-range interactions by the cut-off. More serious than this energy difference is the fact that a cut-off induces artificial forces which lead to a structural relaxation in the system. The relaxation manifests itself in the decrease of electrostatic energy by 500 kcal/mol shown in Figure 2. This finding should serve as ample evidence that an integration scheme which allows simulations which account for long-range Coulomb forces are desirable.

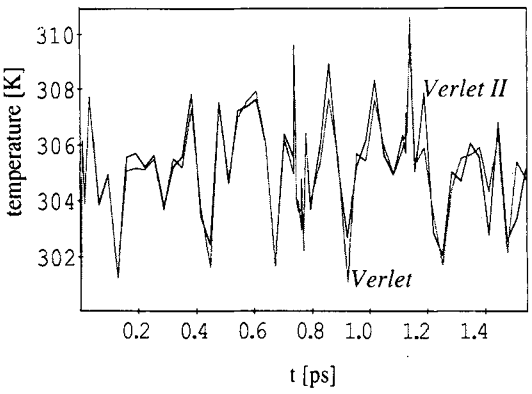

Figure 4 Comparision of the long-term development of the temperature of *Rhodopseudomonas viridis* when using the Verlet algorithm (shaded line) and the Verlet-II algorithm (solid line), respectively. Both simulations used the same equilibrated starting structure, which however is different from the one used in Figures 2 and 3. The same time step of 0.25 fs was used in both simulations.

Such scheme is provided by the conventional distance class algorithm as well as by the Verlet-II algorithm. Figure 3 presents the results of simulations involving these two algorithms. To monitor the accuracy of the simulations we consider the total electrostatic energy of each of the simulations from which we subtracted the corresponding (equal time) total electrostatic energy of simulations according to the conventional Verlet algorithm including all interactions. As can be seen, both the distance class and the Verlet II algorithm reduces the error in energy by nearly three orders of magnitude compared with that of the cut-off scheme (see Figure 2).

In order to demonstrate the accuracy of the Verlet II algorithm for longer simulation periods we compare in Figures 4,5 and 6 observables resulting from Verlet-type and Verlet II-type simulations. In Figure 4 we present the temperature, i.e. essentially the mean kinetic energy resulting from the two simulations lasting 1.5 ps. The results

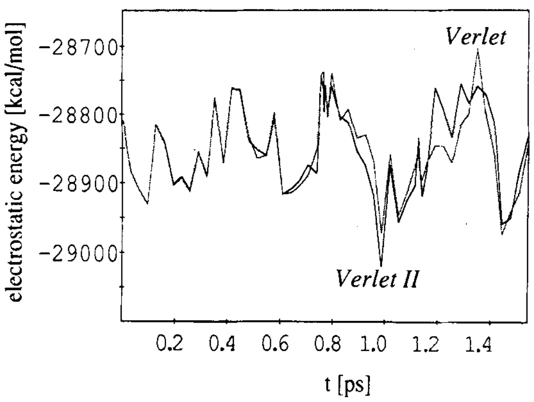

Figure 5 Comparision of the long-term development of the electrostatic energy of *Rhodopseudomonas viridis* when using the Verlet algorithm (shaded line) and the Verlet-II algorithm (solid line), respectively. The data have been extracted from the same simulations as described in Figure 4.

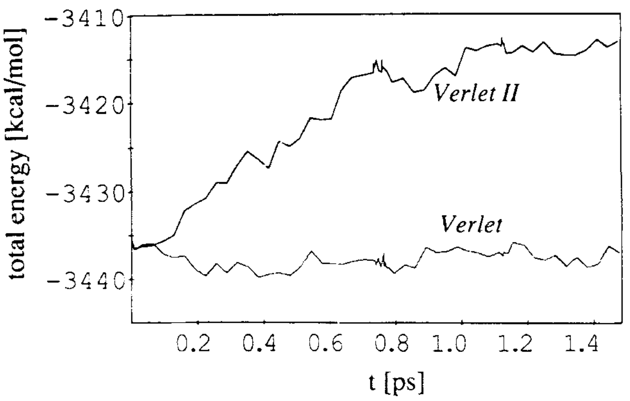

Figure 6 Comparision of the long term time development of the total energy of *Rhodopseudomonas viridis* when using the Verlet algorithm (shaded line) and the Verlet-II algorithm (solid line), respectively. The data have been extracted from the same simulations as described in Figure 4. The devergence needs further investigation.

show good agreement for the complete simulation period. In Figure 5 we present for the same two simulations the total electrostatic energy. This quantity also shows a satisfactory behaviour over 1.5 ps, demonstrating that major structural drifts do not occur between the two simulations. A most stringent test is presented in Figure 6 which tests the conservation of energy at a high resolution. The results presented show that a Verlet-type simulation satsifies energy conservations within an error of about 3 kcal/mol. The corresponding error of the Verlet II-type simulation is 20 kcal/mol. This error is small on the scale of thermal energy measuring only about $10^{-5} \cdot$NkT. Longer simulations showed that the energy drifts monotoneously by about 10 kcal/mol each ps over a 5 ps period.

Another interesting observable to monitor the accuracy of the Verlet II scheme is provided by the positions of the C^{α}-atoms of the protein backbone during the simulations. We have determined during the Verlet-type and the Verlet II-type simulations the mean rms deviations of the C^{α}-atoms from three structures, the X-ray structure, the average structure of the Verlet-type simulation and the average structure of the Verlet II-type simulation. The differences are presented in Table 4.

Figure 7 compares the total energy during two simulations which result from the distance class algorithm and from the Verlet II algorithm. The results show that the Verlet II algorithm at short times reproduces energy conservation better than the distance class algorithm. This difference between the algorithms motivated the generalization of the Verlet integration scheme presented in Section 3.

Table 4 Mean rms deviations in Å of the C^{α}-atoms of the reaction center of *Rhodopseudomonas viridis*. Shown are the values for comparisons between the X-ray structure (X-ray), an averaged structure computed without approximations (Verlet) and one computed with the Verlet-II algorithm (Verlet-II) descirbed in this text. The averaging has been done over a trajectory of 1.5 ps length, starting with an equilibrated structure.

X-ray — Verlet	*X-ray — Verlet-II*	*Verlet — Verlet-II*
2.9552	2.9554	0.0305

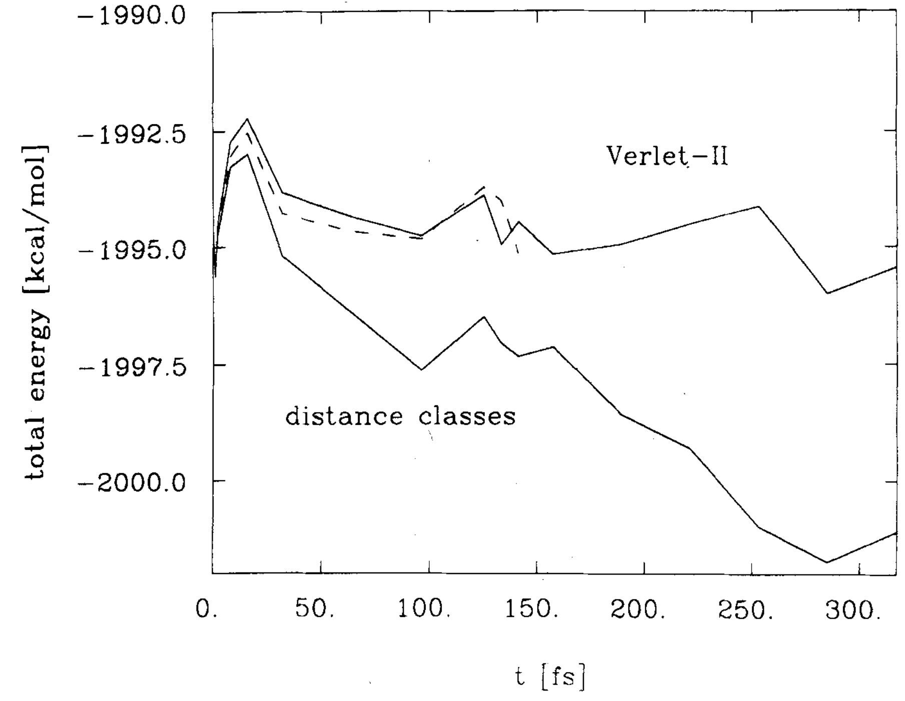

Figure 7 Time development of the total energy of *Rhodopseudomonas viridis* when using the distance class algorithm and the Verlet-II algorithm, respectively. The dashed line represents the development of the total energy computing all interactions. The same simulations as described in Figure 2 have been used.

The results given show that the two simulations show essentially the same mean deviation from the X-ray structure. Also the deviations of C^{α}-atoms between the two simulations are very small. The results indicate that over the simulation period of 1.5 ps no major structural differences can be discerned. We interpret this behaviour as evidence that the Verlet II type algorithm yields simulation results in good agreement with the more time-consuming Verlet-type simulations.

5.3 Performance of the Verlet-II Algorithm

In using the Verlet-II algorithm we achieved considerable reductions of computing times. By implementing bit-lists, as described in Section 4.2, the time required for one integration step (simulating *Rhodopseudomonas viridis* using 12 Transputers) was reduced from 386.6 s to an average value[7] of 71.6 s. Actually, we expect that the fast multipole algorithm (FMPA) [15] applied to a 12 600 atom system would be about 25 percent faster than the Verlet II algorithm. However, the FMPA is notoriously difficult to program and, as a result, can only be parallelized with a great deal of effort.

After implementing a pair-list for the innermost distance class (in addition to the bit-lists), the average computing time for one integration step could be reduced by

[7]The duration of an integration step is not longer constant for the Verlet-II algorithm.

another 26 percent to 53.0 s. Thus, in comparison to the simulation evaluating all forces within each integration step, a speedup of more than seven has been achieved.

The simulations carried out showed that conventional methods to decrease computing time, e.g. the use of cut-off, are accompanied by numerical errors. In contrast to this, the Verlet-II algorithm yields a considerable reduction of computing times without any significant loss of accuracy. Using the Verlet-II algorithm, it seems possible to reproduce conservation of energy even for large many-body systems with a high degree of accuracy.

References

[1] H. Heller, H. Grubmüller, and K. Schulten, "Molecular dynamics simulation on a parallel computer", *Molecular Simulation*, **5** (3,4), 133–165 (1989).

[2] H. Heller, "Dynamiksimulation sehr großer Makromoleküle am Beispiel des photosynthetischen Reaktionszentrums von Rhodopseudomonas viridis", Master's thesis, Technical Universtiy of Munich, Physics Department, T 30, 1988.

[3] H. Grubmüller, "Dynamiksimulation sehr großer Makromoleküle auf einem Parallelrechner", Master's thesis, Technical University of Munich, Physics Department, T 30, 1989.

[4] A.R.C. Raine, D. Fincham, and W. Smith, "Systolic loop methods for molecular dynamics simulation using multiple Transputers", *Comput. Phys. Commun.*, **55**, 13–30 (1989).

[5] D. Fincham, "Parallel computers and molecular simulation", *Molecular Simulation*, **1** (1,2), 1–45 (1987).

[6] W.B. Streett and D.J. Tildesley, "Multiple time-step methods in molecular dynamics", *Molecular Physics*, **35**(3), 639–648 (1978).

[7] M.P. Allen and D.J. Tildesley, "Computer Simulation of Liquids". Clarendon Press, Oxford, 1987.

[8] L. Verlet, "Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules", *Physical Review*, **159** (1), 98–103 (1967).

[9] A. Windemuth, "Dynamiksimulation von Makromolekülen", Master's thesis, Technical University of Munich, Physics Department, T 30, James-Franck-Street, 8046 Garching, August 1988, in this work the conventional distance class algorithm is defined as follows: the Forces for particles in distance class n are updated every 2^n integration steps; between updates the forces are assumed to be constant; the distance classes are the same as defined in Table 3.

[10] W.D. Hillis, "The connection machine", *Scientific American*, **256** (6), 86–93 (1987).

[11] J. Deisenhofer and H. Michel, "The crystal structure of the photosynthetic reaction center from *Rhodopseudomonas viridis*", in *The photosynthetic bacterial reaction center: Structure and dynamics*. Plenum Press, London, (1987).

[12] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus, "CHARMM: a program for macromolecular energy, minimization, and dynamics calculations", *Journal of Computational Chemistry*, **4** (2), 187–217 (1983).

[13] H. Treutlein, "Molekulardynamiksimulation des Reaktionszentrums von Rhodopseudomonas viridis", PhD thesis, Technical University of Munich, Physics Department, T 30, James-Franck-Street, 8046 Garching, (1988).

[14] A.T. Brünger, "X-PLOR", The Howard Hughes Medical Institute and Department of Molecular Biophysics and Biochemistry, Yale University, 260 Whitney Avenue, P.O. Box 6666, New Haven, CT 06511, May (1988).

[15] L. Greengard and V. Rohklin, "A fast algorithm for particle simulation", *Journal of Computational Physics*, **73**, 325–348 (1987).