JOURNAL OF COMPUTATIONAL PHYSICS **124,** 85–92 (1996)
ARTICLE NO. 0046

An Optimized Algorithm for Molecular Dynamics Simulation of Large-Scale Systems

E. GLIKMAN,* I. KELSON,* N. V. DOAN,† AND H. TIETZE†

*School of Physics and Astronomy, The Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel;
†Centre d'Etudes de Saclay, CEREM-DECM-SRMP, 91191 Gif-sur-Yvette Cedex, France

Received June 7, 1994; revised March 31, 1995

A method of optimizing molecular dynamics calculations is presented. The method employs multiple time steps across the computational crystal both for the force evaluation and the neighbor list updating. The time step for each individual atom is chosen according to general criteria which reproduce overall accuracy while saving CPU time. A detailed application is presented to demonstrate the reduction in computation time and the reproducibility of the results.

I. INTRODUCTION

Molecular dynamics (MD) calculations are widely used in physics and chemistry. One of the most successful subjects studied with MD is the process of irradiation damage [1], where its potential for providing fundamental understanding was recognized from the outset. In recent years there is a growing interest in studying irradiation damage caused by particles of increasingly higher energy [2–6]. Due to the high energy processes involved (resulting from particle initial kinetic energies of from 500 eV up to 30 keV) large computational crystals have to be used, with $N_{\text{atom}} \simeq 10^5 - 10^6$. The need to follow the evolution of large numbers of particles for long physical times, has prompted interest in developing qualitative new algorithms to speed up the calculation [7].

A standard MD calculation can be divided into two main parts for every time step. The first part is the calculation of the force acting on each atom. This entails the use of a *neighbor list*, namely a list which contains the identity of all neighbor atoms contributing to that force. The second part is the advancing of the atomic coordinates by integrating Newton's equations. The neighbor list itself has to be updated periodically to take into account the motion of the atoms. This is done usually at constant time intervals, whose magnitude depends on the specific physical problem.

Calculating the forces and managing the neighbor list are the two most time consuming phases of the computation, particularly when the number of particles exceeds 10^4. In this paper we introduce a new method of performing these tasks, which results in considerable CPU time savings. This method is based on selectively limiting the calculation of the forces and the neighbor list updating, according to actual need. In Section II we discuss the details of an optimized scheme of the force evaluation using a multiple time step method. Section III reviews methods of neighbor list calculation and describes the selective updating of atoms in the crystal. In Section IV the implementation of this method in real MD calculations is illustrated. The relevant conclusions are summarized and discussed in Section V, and further improvements and applications are indicated.

II. MULTIPLE TIME STEP METHOD

This section describes the method of using multiple time steps in the same computational crystal. We propose to use this method for simulations of systems with very inhomogeneous velocity (or kinetic energy) distributions, which are typical to high energy irradiation processes. This method is based on a simple observation. When a cascade is generated in a large crystal (with, say, 10^6–10^7 atoms) by a primary energetic particle (with kinetic energy in excess of 10 keV) most of the energetic atoms are concentrated in a small region of the crystal, while a vast majority of the atoms are hardly affected by the event. Following the motion of the fast atoms requires a time step which is much shorter than that necessary to describe accurately the much slower atoms. By allowing, in principle, each atom to be assigned an appropriate, individual time step, a significant saving of CPU time can be achieved, without impairing the accuracy of the calculation. In so doing, one dispenses with the need to calculate too frequently the force acting on most atoms.

There are several common ways to integrate Newton's equations [8, 9]. We use the Verlet method [10], whereby the positions of the atoms x_i are computed for every time step (Δt) through:

$$x_i(t + \Delta t) = 2x_i(t) - x_i(t - \Delta t) + \frac{\Delta t^2}{M_i} \sum_{j \neq i} F_{ij}(t), \qquad (1)$$

0021-9991/96 $18.00

where $x_i(t + \Delta t)$ is the atomic position at the time $t + \Delta t$, M_i the mass of atom i, and F_{ij} is the force exerted by atom j on the atom i. In the standard MD the value of the time step Δt is usually constant for all atoms i independently of their kinetic energy. The evaluation of the force F_{ij}, which is the most time consuming part of the calculation, is done normally at every time step.

In the present model we introduce the use of multiple time steps with values determined for each atom by its energy. Thus, the atoms in the crystal are divided into groups characterized by their energy range and, consequently, by their common time increment. We limit the acceptable time increment to the set $\{\Delta t\}$,

$$\Delta t \in \{2^0\,\Delta t_0, 2^1\,\Delta t_0, 2^2\,\Delta t_0, ..., 2^k\,\Delta t_0, ..., 2^{k_{\max}}\,\Delta t_0\}, \quad (2)$$

namely, to a set of successive multiples of 2 of some Δt_0, which is the time step corresponding to the most energetic atom in the crystal. This is done both for reasons of practical convenience and in order to provide a natural framework for keeping the difference equations centered in time. The largest time step, $2^{k_{\max}}\,\Delta t_0$, corresponds to the lowest kinetic energy of the atoms in the system which is normally the thermal energy. Since at that time the entities corresponding to *all* the atoms are computed, it will be called the *synchronization time.* $k_{\max}$ is normally 5 or 6, in practice, depending to the initial energy of the primary knock-on atom.

Division of Atoms into Groups

As we have indicated, the atoms in the crystal are divided into groups. Each group is characterized by its own Δt (limited to the specific set above) and the corresponding frequency with which the force on its member atoms is calculated. How are these groups constructed and how are they dynamically maintained throughout the calculation?

Three criteria are used in assigning the atoms to groups:

1. The basic and most important criterion is the requirement that an atom does not move more than a certain, predetermined distance ΔX_{const} during its characteristic Δt. This immediately defines the groups of atoms through

$$k^{(i)} = \left[\log_2 \frac{\Delta X_{\text{const}}/v^{(i)}}{\Delta t_0}\right], \quad (3)$$

where $v^{(i)}$ is the velocity and $k^{(i)}$ is the index defining Δt (in the time increment list) for the atom i.

2. For each atom i the difference $|k^{(i)} - k'^{(i)}|$ has to be less than or equal to 1, where $k'^{(i)}$ is the index of time step of the atom i at the previous synchronization time. This ensures a measure of smoothness in the variation of $k^{(i)}$ with time.

3. The difference between the group indices of neighboring atoms has to be less than or equal to 1: $|k^{(i)} - k^{(j)}| \le 1$. This provides for the smoothness of the spatial variation of $k^{(i)}$.

It is important to note that these criteria could depend on the details of the simulated system and might be modified as experience in actual computations is accumulated. They do, however, address the two basic issues in the new scheme: the limitation placed on atomic motion in a single time step and the need to avoid drastic spatial and temporal changes of the variable time step. They also adjust automatically the range of sizes of time steps to the rate of change of system variables and, hence, to a constant level of obtained accuracy of the calculation.

Integration of the Equations of Motion by Group

The unmodified scheme employed in the molecular dynamics calculation is an explicit scheme, in which all atoms can be advanced in time independently of each other. In the multiple time step method one has to modify the scheme (Eq. (1)) whenever an atom changes its characteristic time step and to organize properly the order of integration among the various time groups.

Let the characteristic time step of atom i change from Δt_{old} to Δt_{new} at time t. Expanding the position to second order and the velocity to first order around t one readily obtains the modified relationship

$$x_i(t + \Delta t_{\text{new}}) = 2x_i(t) - x_i(t - \Delta t_{\text{old}}) + \frac{1/2(\Delta t_{\text{new}}^2 + \Delta t_{\text{old}}^2)}{M_i} \sum_{j \neq i} F_{ij}(t). \quad (4)$$

A simple technical alternative for implementing the scheme modification is to integrate the equations of motion for all atoms using the *smallest* time increment Δt_0, but to hold constant the force acting on each atom for the duration of its own characteristic Δt. This alternative, which was found to yield quantitatively equivalent results, reproduces in most respects the approach of Zhu and Averback [7].

It is useful, nonetheless, to indicate the general integration-by-group algorithm which can be utilized because it conserves the formal structure of the Verlet method. Moreover, it is applicable to a more general class of multiple time step schemes, where the groups are characterized not just by their different time step, but by some additional intrinsic properties.

To advance (the coordinates of) an atom from time t to time $t + \Delta t$ one has to know the force acting on it at time t and, hence, the coordinates of all its neighbors at that time. At the synchronization time, and only then, *by definition*, all atomic coordinates are known concurrently, and

all atoms can thus be simultaneously advanced. At intermediate stages between consecutive synchronization times, the simple operational rule is: *advance first the currently most retarded atoms.* At time t, according to this rule, the neighbors of any given atom that is being advanced, have already been advanced themselves either to t or further on. In the latter case, their position at time t is obtained by interpolation between their current and previous positions. This can be either a linear, or—optionally—a higher order interpolation.

It is instructive to understand the formal implications of using the scheme as presented, and—in particular—the limitations which consequently hold in practice. The Verlet algorithm, when applied to motion governed by a conservative force, is fully reversible and preserves rigorously the phase space volume at each time step [11]. It was shown [12] that for such algorithms and for an appropriate finite time step range, one can define an effective pseudo-hamiltonian (with an associated conservation law) for the continuous-time evolution which is equivalent to the time-discretized dynamics. This ensures the stability of the algorithm and its robustness over long time spans, without the buildup of random, uncontrollable errors that might otherwise occur [13]. The algorithm which we used is not strictly reversible and thus must be integrated over long times with caution. Rigorous (in the above sense) multiple time step methods based on the Verlet algorithms have been formally devised [14]. However, we have limited our study to the simple intuitive multiple time step algorithm for a number of reasons. First, jumps in the time step size (which are the source of the irreversibility of the algorithm) amount to a small fraction, around 1% of the overall integration. Second, the total time span of the integration is limited by natural physical considerations—the dissipation of the initial cascade and its containment in the basic periodic crystal—so that runaway errors are not encountered.

III. NEIGHBOR LIST MANAGEMENT

In the previous section we introduced an optimized way of calculating the trajectories of the atoms. In this section we extend the same approach to the management of the neighbor list. We begin with a discussion of the standard neighbor list methods (Verlet and *link-cell* [15]) as well as of a new method which we call *the relational method.*

In the Verlet method one calculates for each atom its distance from all the other $(N_{\text{atom}} - 1)$ atoms. All atoms within a sphere of radius R_c (the interaction cutoff radius) around a particular atom are then assigned to the neighbor list of that atom. Clearly, the time required for computing the complete neighbor list, T_{list}, grows as N_{atom}^2, and becomes prohibitively large for very high N_{atom}. Using the symmetry of the forces, $\mathbf{F}_{ij} = \mathbf{F}_{ji}$, to cut the calculation time by a factor of 2 is only a minor improvement from this point of view.

The standard link-cell method is primarily designed to reduce the order of dependence of the neighbor list calculation time on N_{atom}. Its basic idea is to divide the crystal into cubic subcells, whose size is determined by the interaction cutoff radius R_c. Updating the neighbor list involves two steps: first, assigning each atom to its corresponding subcell according to its current geometric position; second, determining the neighbor list for a given atom i. In so doing the check is limited to atoms in its own subcell and to atoms in geometrically adjacent subcells. Clearly, for systems whose size is much larger than the cutoff radius R_c, the calculation time T_{list} becomes proportional to N_{atom}, with the proportionality constant, depending primarily on the number of atoms per subcell.

Before proceeding with the discussion of the optimization scheme, we introduce a new variant of the standard link-cell method. We call it the *relational method* because it uses directly the neighbor relationship between atoms (rather than their relative geometric position) in updating the neighbor list. Unlike the standard link-cell method, which can be used to construct the neighbor list directly from the ensemble of atomic data, the relational method is a *recursive* algorithm that uses the current neighbor list at time $t - \Delta t$ to create the neighbor list at time t. It is based on the simple observation that, in a simply connected domain, one atom cannot become a neighbor of (i.e., get close enough to) another, unless there is, at some earlier stage, some atom which is a neighbor of both. Thus, to update the neighbor list, one has to check for each atom i only its current neighbors (''first-order neighbors'') and the current neighbors of them (''second-order neighbors''). This effectively limits the check to an approximately spherical region around the atom i, whose radius is $2R_c$. The calculation time of the neighbor list can be estimated by

$$T_{\text{list}} \sim N_{\text{atom}} \cdot (N_{fn} + N_{sn}), \tag{5}$$

where N_{fn}, N_{sn} are the numbers of first- and second-order neighbors, respectively. For an f.c.c. lattice, with the commonly used interaction cutoff radius, one has $N_{fn} \simeq 50$ and $N_{sn} \simeq 2^3 \cdot N_{fn}$. We recall that because of its inherent recursive nature, an alternative method has to be used in order to initialize the neighbor list.

The neighbor list must represent fully and accurately the system state at all times. In principle, therefore, it should be updated every time step. This, however, is not only too time consuming, but it is also unnecessary under most ordinary circumstances. Consequently, the common practice in MD calculations is to update the neighbor list once every predetermined number of time steps. This is a compromise between the need to preserve overall accuracy and the wish to save CPU time.

Applying the arguments used in connection with the multiple time step method, it is clear that there is no need to update the neighbor list of *low energy* atoms as frequently as one updates the neighbor list of *high energy* atoms. The proposed optimization consists of updating the neighbor list of each atom as frequently as required by its own individual conditions. This will result in considerable time saving, because there are many more low energy atoms than high energy atoms in the crystal at all times.

Quantitative criteria are needed in order to select at each time step the atoms whose neighbor list is to be updated, or ''refreshed.'' Again, as in the corresponding criteria for the multiple time step method, case-specific considerations may have to be applied. In practice, we have tried two different alternatives for controlling this partial neighbor list updating.

i. In the first option one updates the neighbor list every time step for atoms whose kinetic energy is higher then some threshold every E_{thresh}. E_{thresh} depends on the basic time step size (which is a constant of the problem) and the maximum kinetic energy of a crystal atom. In order to take into account neighbor list modifications resulting from low energy and thermal motion of the atoms over extended time intervals, the full neighbor list is updated periodically.

ii. In the second option [16], one monitors for each atom the distance between its current position and its position at the last time its neighbor list was updated. Whenever the total displacement exceeds, for a given atom, some prescribed value ΔR, this atom's neighbor list is refreshed. ΔR is a function of the time step size and the interaction cutoff radius R_c and has to be properly adjusted. Note that under this option there is much less need to perform periodic recalculations of the full neighbor list.

The running time required to manage the neighbor list was substantially reduced under both options and for both relevant methods (link-cell and relational). This is because it behaves as

$$T_{\text{list}} \sim N_{\text{high}} \cdot (N_{fn} + N_{sn}), \tag{6}$$

where N_{high} is the number of atoms whose energy exceeds the threshold energy (or, alternatively, have been displaced more than ΔR) and

$$N_{\text{high}} \ll N_{\text{atom}}. \tag{7}$$

Finally, we note that the major objective of this treatment is not to reduce T_{list} (which is, anyway, much smaller than the force calculation time), but to put both phases of the MD calculation on a conceptually equivalent basis.

IV. NUMERICAL EXAMPLE

In this section we describe a MD calculation (on a copper crystal) which serves as a testing ground for the methods presented in the previous sections. The testing procedure consists of an ensemble of calculations with variable operational parameters, carried out both before and after the optimization methods are applied. The corresponding runs are then compared to determine the relative gain in CPU time and to check for the reproducibility of the results.

This application is the one originally employed for cascade calculations in copper [17] and adapted for treatment on nuclear stimulated desorption in palladium [18]. The computational crystal is an f.c.c. structure; for a detailed description of the technical aspects related to its construction see Ref. [17]. The interaction between the atoms is described by an empirical many-body potential derived from a second-moment approximation to the tight-binding scheme of electronic density of states [19]. The total cohesive energy is written as the sum of an attractive term representing the tight-binding band energy due to the d-electrons and a pairwise repulsive interaction term of the Born–Mayer type,

$$E_{\text{coh}} = \sum_i (E_i^A + E_i^R) \tag{8}$$

with

$$E_i^A = -\left[\sum_{i \neq j} \xi^2 \exp\left[-2q\left(\frac{r_{ij}}{r_0} - 1\right)\right]\right]^{1/2} \tag{9}$$

$$E_i^R = \sum_{i \neq j} A \exp\left[-p\left(\frac{r_{ij}}{r_0} - 1\right)\right], \tag{10}$$

where ξ is a hopping integral between nearest neighbor sites, r_{ij} is the distance between atoms i and j, and r_0 is the nearest neighbor distance.

The interaction parameters for copper are

$$A = 0.0905 \text{ eV}, \ \xi = 1.243 \text{ eV}, \quad p = 10.68, \quad q = 2.32. \tag{11}$$

Periodic boundary conditions are applied with a period equal to the microcrystallite size. Boundary effects do not perturb significantly the results, because of the short time of the cascade simulation (about 10^{-12}–10^{-11} s) and the short-range nature of the interaction.

The trajectories of the atoms in the microcrystallite were determined by the numerical solution of Newton's equations of motion. The total force F_i exerted on an atom i of mass m_i contains contributions from all the atoms located within a sphere of a prescribed cutoff radius around it,

$$F_i = -\sum_{j \neq i} \nabla_{r_i} E_{\text{coh}}. \tag{12}$$

Starting from a given equilibrium configuration, one specific atom in a predetermined site is given an initial kinetic energy. The resultant cascade is then followed in phase space for a sufficiently long physical time, typically until the number of induced defects has reached its asymptotic value. Each run was characterized by the dimensions of the crystal, the position of the primary energetic atom, its initial energy E_{init}, and its direction. The basic minimal time step Δt_0 was taken universally as 1×10^{-16} s, while the maximal time group index, $k_{\max}$, as defined in Eq. (2), was allowed to vary. The total energy, the kinetic energy, and the number of displaced atoms were monitored throughout the calculation, as well as entities related specifically to the optimizing scheme. In particular, the total number of atoms in each time group, N_k, was followed and recorded.

The treatment of the neighbor list, performed in conjunction with these calculations, was the following. At every synchronization time a partial ''refreshment'' of the neighbor list was carried out while every five synchronization time steps a full updating (using the link-cell method) was performed. We note that this occasional full updating is not really necessary and could be done with a lower frequency.

We present results obtained for a cascade generated by an atom with $E_{\text{init}} = 500$ eV in a crystal with 18000 atoms. The crystal was initially at a temperature of 20°K, and the parameter Δx_{const} was equal 0.01 Å. Figure 1 shows the variation with time of the numbers N_k, with $k_{\max} = 3$ (four groups with time increments equalling Δt_0, $2\Delta t_0$, $4\Delta t_0$, and $8\Delta t_0$, respectively).

The two following points are worth noting:

— The basic premise of the multistep scheme is apparent in the graph. Most atoms are advanced with the largest increment allowed in this case, $8\Delta t_0$, while consecutively smaller fractions are advanced with the smaller time steps. This is a dynamic process. Each of the numbers N_k for the smaller groups reaches a maximum, before decreasing and fading away. In fact, the scheme automatically treats the entire atom population with the time step structure appropriate to the designated accuracy.

— The main criterion for assigning atoms to time-groups (Eq. (3)) is a ''hard'' one. Namely, every time an atom crosses a critical velocity, it jumps from one group to another, causing a corresponding group shift among its neighbors. This abrupt behavior, which is a natural consequence of the criteria employed here, is seen in Fig. 1. It is possible to design ''softer'' criteria which will result in smoother behavior of these functions. In any event, this behavior does not influence the accuracy of the calculation.

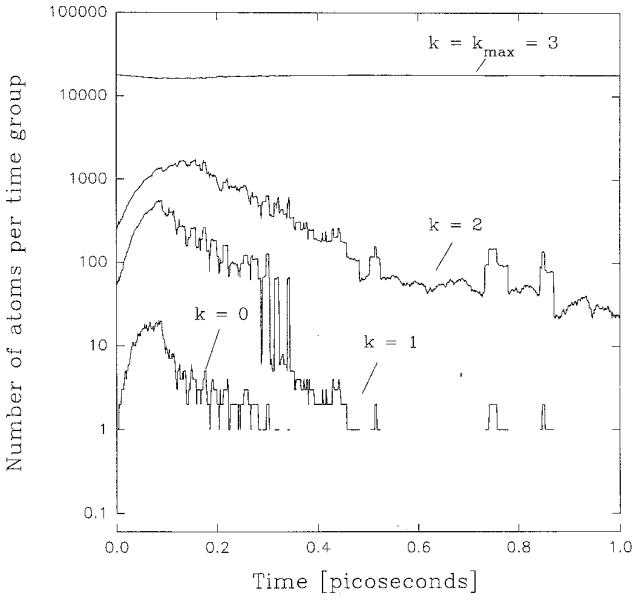

FIG. 1. The time dependence of N_k for a run with $N_{\text{atom}} = 18000$, $E_{\text{init}} = 500$ eV and $k_{\max} = 3$. The lowest group is advanced with $\Delta t = 1 \times 10^{-16}$s and the highest group with $\Delta t = 8 \times 10^{-16}$s. Note the predominance of the higher groups during the entire run. The abrupt fluctuations in N_k stem from the use of a ''hard'' criterion for the time increment assignment.

A cascade event transforms kinetic energy, concentrated initially on one atom, into crystalline heat and disordering energy. This disordering manifests itself in the form of atoms displaced from the equilibrium sites of the crystalline structure, and eventually gives rise to residual permanent defects. The number of displaced atoms as a function of time, $N_{\text{disp}}(t)$, and its asymptotic value provide a sensitive test for the accuracy and consistency of the calculation. Figure 2 shows that function for a series of runs of the same physical system ($N_{\text{atom}} = 18000$; $E_{\text{init}} = 500$ eV), where the number of allowed time groups was increased progressively from one to four ($k_{\max} = 0, ..., 3$). The minimal dedisplacement used in the compilation of these functions was 0.643 Å. The one-group case, where all particles are advanced with the basic Δt_0 throughout the computation, serves as a *reference* run for comparison. It is seen, that the function $N_{\text{disp}}(t)$ is practically identical for all levels of approximation of the numerical scheme. In some longer runs, not shown on the figure, it was found that the value of $N_{\text{disp}}(t)$ obtained at $t = 1$ ps remains unchanged as the integration is carried out to $t = 2$ ps, and thus represents, indeed, a consistent asymptotic value.

One of the crucial tests for the validity and accuracy of dynamic calculations of closed physical systems, is the conservation of total energy. Figure 3 shows the total energy (per particle) as a function of time for the runs shown in Fig. 2. The different approximations reproduce exactly

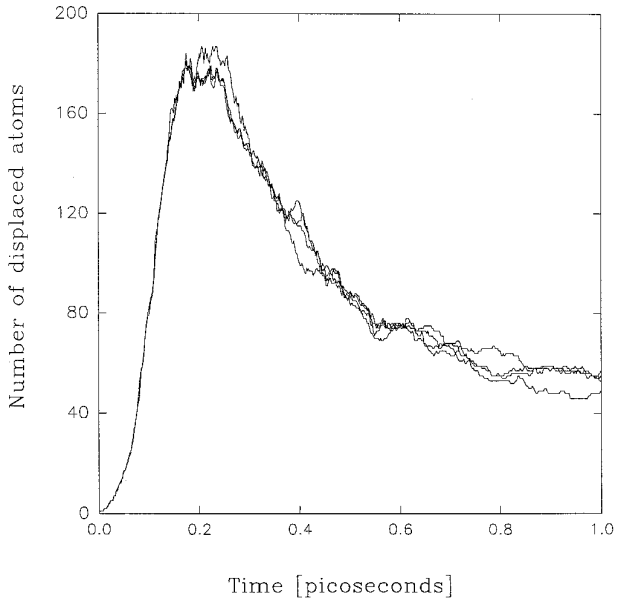

FIG. 2. The time dependence of the number of atoms displaced from their original site by more than 0.643 Å, for the run with $N_{\text{atom}} = 18000$ and $E_{\text{init}} = 500$ eV. The different curves correspond to the successive approximations, going from $k_{\max} = 0$ (the reference case) to $k_{\max} = 3$. Because of the great similarity between the curves, they are not individually identified.

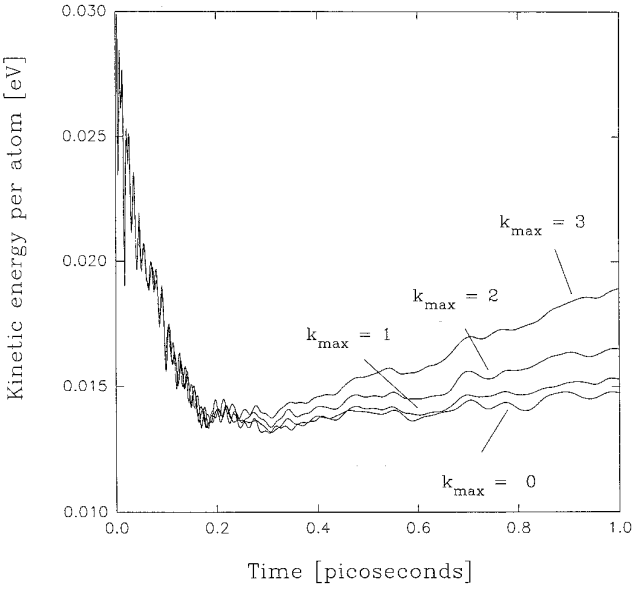

FIG. 4. The time dependence of the kinetic energy per atom for the run with $N_{\text{atom}} = 18000$ and $E_{\text{init}} = 500$ eV. The different curves correspond to the successive approximations, going from $k_{\max} = 0$ (the reference case) to $k_{\max} = 3$. The dispersion of the kinetic energy from the reference case is about one-fourth of the corresponding dispersion in the total energy.

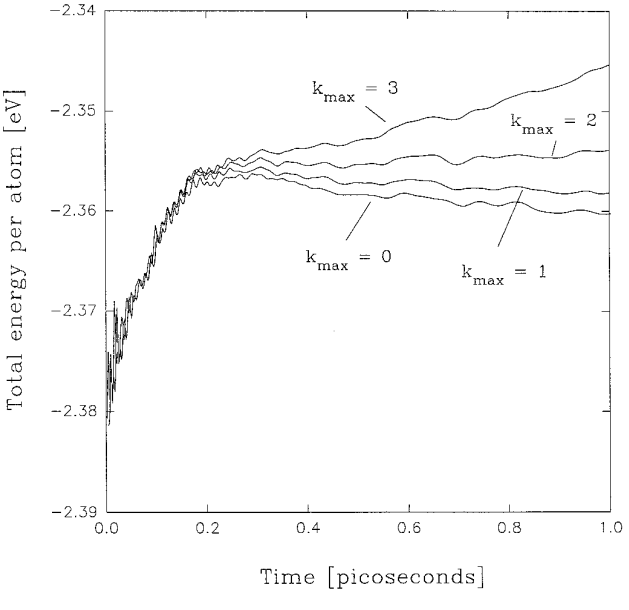

FIG. 3. The time dependence of the total energy per atom for the run with $N_{\text{atom}} = 18000$ and $E_{\text{init}} = 500$ eV. The different curves correspond to the successive approximations, going from $k_{\max} = 0$ (the reference case) to $k_{\max} = 3$. The integrated energy increment is about 1% for the first 2000 basic time steps and less than 0.5% over the next 8000 basic time steps.

the same time dependence in the first 0.2 ps. This might indeed be expected from the behavior of the functions N_k for lower k, which peak at around 0.2 ps. The relative incremental energy $\Delta E/E_{\text{tot}}$ per synchronization time step is around 4×10^{-5} during that period. Following that, there is a minor dispersion between the various runs, which increases monotonically with the order of the approximation (i.e., with $k_{\max}$). However, even for the coarsest approximation, the average $\Delta E/E_{\text{tot}}$ is only 4×10^{-6} per synchronization time step.

Since one of the principal factors which determine an atom's time step is its velocity, we plot in Fig. 4 the corresponding time evolution of the *kinetic* energy per particle. Again one sees a dispersion between the different approximations, which reaches about 0.002 eV/particle at $t = 1$ ps. This is very small compared to the *total* energy per particle. It is not small when compared to the kinetic energy itself. We note, however, that the entire behavior of the kinetic energy is rather spurious in this type of calculation, representing the effect of the finite size of the computed system. This artificial maintaining of thermal velocities is, in fact, what sets the limits on $k_{\max}$ in the present case.

Table I gives the CPU time spent on the various phases of the calculations in the successive approximations, along with CPU time gain factors Θ obtained in each case. The neighbor list figure quoted for the reference case corresponds to a full updating every 16 time steps (of 1×10^{-16})

TABLE I

The Actual Computation Times, in Seconds, of the First 0.12 ps (1200 time steps) of the $N_{atom} = 18000$, $E_{init} = 500$ eV case

	Ref.	$k_{max} = 1$	$k_{max} = 2$	$k_{max} = 3$
Force calculation	17,077	8,780	4,570	2,450
Neighbor list	443	94	94	94
Main	150	150	150	150
Overhead	—	76	58	30
Total time	17,670	9,100	4,872	2,724
Gain factor (Θ)	1	1.94	3.63	6.49

Note. The successive approximations, going from $k_{max} = 1$ to $k_{max} = 3$ are compared to the reference case. The gain factor Θ is the corresponding ratio of the total CPU times. The computations were carried out on a Silicon Graphics R4400 workstation.

with no ''refreshing'' in between. The ''overhead'' entry represents the additional computations which had to be carried out in implementing the optimizing scheme. It is interesting to note, that to a good approximation, one has in this case

$$\Theta \approx 2^{k_{max}}. \tag{13}$$

V. SUMMARY AND DISCUSSION

The numerical scheme which we have described and tested provides a set of different time steps used to advance the atoms in the crystal treated within the framework of molecular dynamics. The time steps are derived from some basic increment through multiplying by successive powers of two. This is a natural choice which provides for ease of manipulation, but it is not strictly required conceptually. There is, in practice, no real need to limit externally either the lowest or the highest allowed time step. The scheme automatically adjusts the time step assignment of all the atoms, according to criteria based on accuracy and continuity. The same underlying justification is applied to the neighbor list management. Namely, computations are carried out, as far as possible, only when and where needed. Essentially, the CPU gain factor, for the force calculation part, is simply proportional to the largest (synchronization) time step.

In the previous section we have presented one specific numerical example. It is interesting to see how the above observations would be affected when the parameters of the system are modified; in particular, the initial energy E_{init} and the number of particles N_{atom}. The two are, in fact, related. The initial energy, which is a *physical* parameter, dictates the number of particles required to deal appropriately with the system from the *computational* point of view. N_{atom} increases with E_{init} in order to allow the entire cascade to be contained within the basic simulation domain and to avoid unphysical self-interference due to the periodic boundary conditions. The detailed dependence of N_{atom} on E_{init}, and of the CPU time on both, may vary with the specific problem. In general, both for the simple and the multistep calculation of any problem, the CPU time is simply proportional to N_{atom}. Note, however, that the CPU gain factor may also increase with N_{atom} since a larger fraction of the additional atoms included in the expanded problem may be treated with an increased maximal time step. Thus, one should expect *a weaker overall dependence of the CPU time on* E_{init} for the optimized scheme.

There is a limit on the CPU time gain which can, in principle, be achieved. When the modified (optimized) force calculation is reduced to the extent that its time consumption is about equal to that of the unaffected segments of the computation, no further gain is obtained. At this stage one must resort to parallel computing, in order to treat larger problems within a reasonable time. We note that the structure of the present scheme is inherently suitable for parallel computation.

ACKNOWLEDGMENTS

We thank Professor G. Martin for his critical comments and valuable suggestions. This work was supported by the United States–Israel Binational Science Foundation.

REFERENCES

1. J. B. Gibson, A. N. Goland, M. Milgram, and G. H. Vineyard, *Phys. Rev.* **120,** 129 (1960).
2. J. P. Biersack (Ed.), *Proceedings Int. Conf. On Computer Simulations of Radiation Effects in Solids, Radiat. Eff., 1993.*
3. T. Diaz de la Rubia and M. W. Guinan, *Phys. Rev. Lett.* **66,** 2766 (1991).
4. T. Diaz de la Rubia, A. Caro, and M. Spaczer, *Phys. Rev. B* **47,** 11483 (1993).
5. F. Gao, D. J. Bacon, *Philos. Mag. A* **67,** 275, 289 (1992).
6. A. F. Calder and D. J. Bacon, *J. Nucl. Mater.* **207,** 25 (1993).
7. H. Zhu and R. S. Averback, NIMB **83,** 334 (1993).
8. D. W. Heermann, *Computer Simulation Methods in Theoretical Physics,* 2nd ed. (Springer-Verlag, Berlin/Heidelberg/New York, 1989), p. 13.
9. H. J. C. Berendsen and W. F. van Gunsteren, in *Molecular Dynamics Simulation of Statistical Mechanical Systems,* edited by G. Ciccotti and W. G. Hoover (North Holland, Amsterdam, 1986).
10. L. Verlet, *Phys. Rev.* **159,** 98 (1967).
11. M. Creutz and A. Gocksch, *Phys. Rev. Lett* **63,** 9 (1989).
12. S. Auerbach and A. Friedman, *J. Comput. Phys.* **93,** 189 (1991).
13. S. Toxvaerd, *Phys. Rev. E* **50,** 2271 (1994).
14. M. Tuckerman, B. J. Berne, and G. J. Martyna, *J. Chem. Phys.* **97,** 1990 (1992).
15. D. M. Heynes and W. Smith, *in Information Quarterly for Computer Simulation of Condensed Phases*, No. 26 (Science and Engineering

Research Council, Daresbury Laboratory, Daresbury, England, 1987, p. 68.

16. M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford Univ. Press, Oxford, 1989).
17. F. Rossi and N. V. Doan, *Nucl. Inst. Methods Phys. Res. B* **61,** 27 (1991).
18. E. Glikman, I. Kelson, and N. V. Doan, *J. Vac. Sci. Technol. A* **9,** 321 (1989).
19. V. Rosato, M. Guillope, and B. Legrand, *Philos. Mag.* **35,** 379 (1977).