The Complexity of Parallel Symplectic Molecular Dynamics Algorithms

Roman Trobec

Jožef Stefan Institute, University of Ljubljana, Jamova 39, Ljubljana, Slovenia

Franci Merzel and Dušanka Janežič*

National Institute of Chemistry, Hajdrihova 19, Ljubljana, Slovenia

Received April 1, 1997[⊗]

Parallelized algorithms for molecular dynamics integration, the explicit Leap-Frog-Verlet (LFV), the implicit Gauss−Legendre Runge−Kutta (GLRK), and the new Split Integration Symplectic Method (SISM) are described and compared. All three methods are evaluated on a system of N linear molecules of the form H−(−C≡C−)$_r$−H with the number of atoms in the chain equal to $n = 2r + 2$. The parallel algorithms, memory requirements, and the complexity of computations are estimated for all three methods. Run-time results of the implemented simulations are given for different high-performance computers.

1. INTRODUCTION

Molecular dynamics (MD) simulation, in which the classical equations of motion for all particles of a system are integrated over a finite period of time, is a widely used tool in many fields of science.[1,2] Algorithms for MD simulation are typical candidates for implementation on parallel computers because the forces on each particle of the system can be computed independently.[3] MD integration methods can be implemented using different integration algorithms.[4,5] However, the algorithms are not equally efficient on any type of parallel computer. The obvious question is, "Which method should be chosen to get optimal results on the available computer?"

For most of the proposed Leap-Frog-Verlet (LFV) integrations,[6] based on the second-order generalized leap-frog algorithm, the major obstacle that reduces the efficiency of the simulation methods lays in the numerical instability unless the time step is small enough.[7] Consequently, a smaller time step requires a larger number of steps to simulate the system for a given period of time; therefore, these methods are less efficient for solving larger macromolecular systems, particularly for longer simulation periods.[8] On the other hand, the LFV algorithm can be parallelized efficiently on a message passing computer with the ring-like topology.[9]

Enlarging of the time step in the MD can be partially overcome by the use of implicit methods,[10−12] particularly the symplectic Runge−Kutta methods for the numerical solution of Hamilton equations.[13] Hamiltonian systems possess an important property in which the flow in the phase space is symplectic, thus implying the existence of the phenomenon of the Poincaré recurrence.[14] It was shown that an algorithm based on the two-stage fourth-order implicit Gauss−Legendre Runge−Kutta method (GLRK) enables the enlargement of the time step for a factor of 2.[15,16] Unfortunately, this gain is penalized with an eight times greater computation complexity for the particular example of GLRK.[17] In spite of the parallel implementation, the GLRK method is less effective as the standard LFV.[18]

A new promising approach is based on the fact that the total system Hamiltonian can be split into the harmonic part and the remaining part in such a way that both parts can be efficiently computed.[19] High-frequency harmonic vibrations are defined as the dynamically leading term; the difference between the total potential and its harmonic terms is the remaining part of the Hamiltonian. The harmonic vibrations are treated analytically, independent of the size of the integration time step, by normal mode analysis which is carried out only once, at the beginning of calculation. Then the enlargement of the integration time step up to an order of magnitude is obtained using the Split Integration Symplectic Method (SISM)[20,21] for MD simulation in comparison with the LFV for the same accuracy.

In this paper we compare the parallel LFV, GLRK, and SISM algorithms for MD integration. The layout of the paper is as follows. In the next section the underlying models and methods with the simplified program code of algorithms are described; the codes were written on purpose to make possible the comparisons on equal footing. In Section 3 the comparison of the performances is given in terms of memory requirements and the theoretical time complexity. The measured run-time results for all three methods on sequential and parallel types of computers are presented in Section 4. The paper is concluded by Section 5 in which the presented results are discussed.

2. MODEL AND METHODS

The Model. The evaluation model was composed of a system of N linear molecules of the form H−(−C≡C−)$_r$−H with the number of atoms in the chain equal to $n = 2r + 2$. The same cube of the size L was chosen for all three methods to give realistic merits of numeric accuracy and efficiency. The periodic boundary conditions were imposed to overcome the problem of surface effects.[2] Initial conditions for coordinates and velocities of the system were obtained from an initial molecular dynamics run of randomly distributed molecules long enough to ensure a Maxwell distribution of velocities according to the temperature $T = 300$ K. Bond lengths and stretching constants were taken from van Gunsteren and Berendsen,[22] and Lennard-Jones parameters were taken from Allen and Tildesley.[2] Atomic charges of

* Author to whom correspondence should be addressed. E-mail address: dusa@kihp5.ki.si.

[⊗] Abstract published in *Advance ACS Abstracts,* November 1, 1997.

S0095-2338(97)00226-6 CCC: $14.00

```
main()
{
      int k;

      init_state();

      for (k=0; k<No_steps; k++) {  /* main loop */
           prop();           /* half step propagation of coordinates */
           iforce();         /* internal forces */
           grad();           /* external forces and
                                full step propagation of momenta */
           prop();           /* half step propagation of coordinates */
           energy();         /* total energy */
      }
}
```

Figure 1. Simplified C-code for the main program of the LFV algorithm.

the linear molecules were calculated by fitting the point charges to the *ab initio* molecular electrostatic potential rather than using Mulliken charges.[23]

To perform the MD simulation of a system with a finite number of degrees of freedom, it is a common approach to solve the Hamilton equations of motion

$$\frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial H}{\partial q_i}, \quad \frac{\mathrm{d}q_i}{\mathrm{d}t} = \frac{\partial H}{\partial p_i}, \qquad i = 1, ..., d \tag{1}$$

where H is the Hamiltonian, q_i and p_i are the coordinate and momentum, respectively, and d is the number of degrees of freedom. The fundamental property of the Hamiltonian system is that for each fixed value of t the corresponding t-flow $\phi_{t,H}$ is a symplectic or canonical mapping.[24] Integration of the Hamilton system requires the integrator to be symplectic.[25]

For the model Hamiltonian used it was assumed that bond stretching satisfactorily describes all internal vibrational motions for linear molecules and the model Hamiltonian was of the form

$$H = T + V \tag{2}$$

$$T = \sum_i \frac{\mathrm{p}_i^{\,2}}{2m_i} \tag{3}$$

$$V = \sum_{\text{bonds}} K_b(b - b_0)^2 + \sum_{i>j} \frac{e_i e_j}{r_{ij}} + \sum_{i>j} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{4}$$

where m_i is the mas of the i-atom, b_0 is the reference value for bond length, K_b is the corresponding force constant, i and j run over all atoms, e_i denotes the charge on the i-atom, and r_{ij} is the distance between atoms i and j, and ϵ_{ij} and σ_{ij} are the corresponding constants of Lennard-Jones potential.

LFV Method. The LFV integration method, which is known to be symplectic, propagates coordinates and momenta on the basis of the equation of motion (1) by the following relations

$$q'_i = q_i + \frac{p_i}{m}\frac{\Delta t}{2} \tag{5}$$

$$p_{i+1} = p_i - \Delta t \left(\frac{\partial V}{\partial q}\right)_{q=q'_i} \tag{6}$$

$$q_{i+1} = q'_i + \frac{p_{i+1}}{m}\frac{\Delta t}{2} \tag{7}$$

where q_i is the coordinate, p_i is the momentum, $\dim(p_i, q_i) = d$, d is the number of degrees of freedom, Δt is the time step, and m is the mass of the corresponding atom.

In Figure 1 the simplified C-code for the main program of the LFV algorithm is given. After the initialization of coordinates, momenta, and force buffers, the main loop of the LFV algorithm is repeated for the *No–steps* times. The new coordinates **q** are calculated and the periodic boundary conditions are tested in the procedure *prop*. Intramolecular forces—stretching—and new half-time step momenta **p** are calculated within the procedure *iforce*. The calculation of intermolecular forces is implemented in the procedure *grad*. The procedure *prop* calculates the second half-time step momenta. Then the total energy is calculated in the procedure *energy* (see ref 6 for details).

GLRK Method. The general implicit Runge−Kutta (RK) method requires the equation of motion to be written as a system of first-order differential equations

$$\dot{\mathbf{x}} = \mathbf{v} \qquad \dot{\mathbf{v}} = \mathbf{f}(\mathbf{x}) \tag{8}$$

The system (8) is Hamiltonian since the force is the gradient of the potential $-V(\mathbf{x})$

$$\mathbf{f} = -\nabla V(\mathbf{x}) \tag{9}$$

A general s-stage implicit RK method for the solution of the first-order initial value problem

$$\mathbf{y}' = \mathbf{F}(\mathbf{y}) \qquad \mathbf{y}(0) = \mathbf{y}_0 \tag{10}$$

is of the form

$$\mathbf{Y}_k = \mathbf{y}_m + \Delta t \sum_{l=1}^{s} a_{kl}\mathbf{F}(\mathbf{Y}_l) \qquad k = 1, ..., s \tag{11}$$

$$\mathbf{y}_{m+1} = \mathbf{y}_m + \Delta t \sum_{k=1}^{s} b_k \mathbf{F}(\mathbf{Y}_k) \tag{12}$$

where $\mathbf{y}$ is a d dimensional vector, determined by the $s \times s$ matrix $\mathbf{A} = [a_{kl}]_{k,l=1}^{s}$, and the s-dimensional vector $\mathbf{b} = [b_k]_{k=1}^{s}$, and Δt is the time step. The coefficients of the method are collected in the form of Butcher's tableau[26] which provides a condensed representation of the RK method

c	A
	$\mathbf{b}^T$

where $\mathbf{c} = [c_k]_{k=1}^{s}$ and $c_l = \sum_{l=1}^{s} a_{kl}$

Since the coefficients of the s-stage RK method of order $2s$ satisfy the relation

$$b_k a_{kl} + b_l a_{lk} - b_k b_l = 0, \quad 1 \le k, l \le s \tag{13}$$

the method is symplectic.[13] The unique s-stage RK method of order $2s$, the Gauss−Legendre (GL) scheme, obeys eq 13 for every s.

The application of the two-stage fourth-order GLRK method to MD from eq 8 leads to the following relations which implicitly define the intermediate stages (point values) $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{X}_1$, and $\mathbf{X}_2$

$$\mathbf{V}_1 = \mathbf{v}_m + \Delta t \left(\frac{1}{4}\mathbf{f}(\mathbf{X}_1) + \frac{3-2\sqrt{3}}{12}\mathbf{f}(\mathbf{X}_2)\right)$$

$$\mathbf{V}_2 = \mathbf{v}_m + \Delta t \left(\frac{3+2\sqrt{3}}{12}\mathbf{f}(\mathbf{X}_1) + \frac{1}{4}\mathbf{f}(\mathbf{X}_2)\right)$$

$$\mathbf{X}_1 = \mathbf{x}_m + \Delta t \left(\frac{1}{4}\mathbf{V}_1 + \frac{3-2\sqrt{3}}{12}\mathbf{V}_2\right)$$

$$\mathbf{X}_2 = \mathbf{x}_m + \Delta t \left(\frac{3+2\sqrt{3}}{12}\mathbf{V}_1 + \frac{1}{4}\mathbf{V}_2\right) \tag{14}$$

where $\mathbf{V}_k$ and $\mathbf{X}_k$ correspond to $\mathbf{Y}_k$ from eq 11 and $\mathbf{f}$ to the force in eq 9.

The new step velocities and the new step positions are in accordance with eq 12 given by

$$\mathbf{v}_{m+1} = \mathbf{v}_m + \frac{\Delta t}{2}(\mathbf{f}(\mathbf{X}_1) + \mathbf{f}(\mathbf{X}_2))$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \frac{\Delta t}{2}(\mathbf{V}_1 + \mathbf{V}_2) \tag{15}$$

The procedure is repeated until the desired number of calculation steps is performed.

The simplified C-code of the two-stage fourth-order GLRK method is given in Figure 2. The algorithm is composed of the same basic parts as the LFV method. The setting of initial positions and velocities, and the initialization of data buffers, is performed first in the procedure *init_state*. The main loop is started with the procedure *prop* which propagates the velocities and positions. Then predictors (forces) are calculated in the procedure *force*; at each step, four iterations for the corrector are performed in two intermediate points in the procedures *grad_1st*, *grad_2nd*, which leads to eight calculations of the sum of forces. Then new step velocities and positions and the global kinetic energy are calculated in the procedures *prop* and *energy* (see ref 17 for details).

SISM Method. The SISM was derived in terms of the Lie algebraic language.[24] The formula

$$\mathbf{x}|_{t_0+\Delta t} = \exp(\Delta t \hat{L}_H)\mathbf{x}|_{t_0} \tag{16}$$

where $\hat{L}_H$ is the Poisson bracket operator and $\mathbf{x} = (\mathbf{q}, \mathbf{p})$ is a vector in phase space composed of the coordinates and momenta of all particles, provides a way for integrating the Hamiltonian system in terms of Lie operators.[27]

The construction of an efficient algorithm rests on the ability to separate the Hamiltonian into parts which are themselves integrable and also efficiently computable. Suppose that the Hamiltonian H is split into two parts

$$H = H_0 + H_r \tag{17}$$

where

$$H_0 = \text{i.p.}\left(\sum_i \frac{\mathbf{p}_i^2}{2m_i}\right) + \sum_{\text{bonds}} K_b(b - b_0)^2 + H_{\text{tran/rot}} \tag{18}$$

$$H_r = \sum_{i>j} \frac{e_i e_j}{r_{ij}} + \sum_{i>j} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{19}$$

and i.p. refer to the internal part of the kinetic energy which corresponds to molecular vibrations, m_i is the mass of the i-atom, b_0 is the reference value for the bond length, K_b is the corresponding force constant, $H_{\text{tran/rot}}$ corresponds to the translational and rotational part of kinetic energy, i and j run over all atoms, e_i denotes the charge on the i-atom, r_{ij} is the distance between atoms i and j, and ϵ_{ij} and σ_{ij} are the corresponding constants of the Lennard-Jones potential. Then the following approximation for eq 16 can be used

$$\mathbf{x}|_{t_0+\Delta t} \approx \exp\left(\frac{\Delta t}{2}\hat{L}_{H_0}\right)\exp(\Delta t \hat{L}_{H_r})\exp\left(\frac{\Delta t}{2}\hat{L}_{H_0}\right)\mathbf{x}|_{t_0} \tag{20}$$

which prescribes how to propagate from one point in phase space to another. First, the system is propagated a half-step evolution with H_0, then a whole step with H_r, and finally another half-step with H_0. This scheme is called the generalized leap-frog.[28]

This integration scheme was used as a basis for development of SISM, a second-order symplectic integration algorithm, for MD integration. The Hamiltonian H_0 describes the internal vibrational motion of the molecules and also translation and rotation of the molecules. It represents the dynamically leading contribution whose potential depends only on constant parameters of the simulation. This separation of the potential function allows calculation of the normal modes only once, at the beginning of the calculation.[29−31] The analytical treatment of high-frequency terms in the Hamiltonian permit the SISM to employ up to an order of magnitude larger integration step size than can be used by other methods of the same order.[19,32]

The simplified C-code of the SISM method is given in Figure 3. The algorithm starts, as do its predecessors, with the setting of initial positions, velocities, and data buffers within the procedure *init_state*. The procedure *norm_mode* calculates normal modes of the vibrational part of H_0 to get frequencies and normal modes of vibration. The procedure *split* selects the internal displacement coordinates and momenta (describing vibrations), the coordinates and velocities of the centers of molecular masses, and angular momenta of the molecules from the Cartesian coordinates and from momenta. The procedures *convers_in* transform internal displacement coordinates and momenta into internal normal coordinates and to momenta. The main loop is started with the procedure *evol* which considers half-step

```
main()
{
      int k,i,j;

      init_state();

      for (k=0; k<No_steps; k++) {  /* main loop */

            prop();     /* propagation of velocities and positions */
            for (i=0; i<4; i++){ /* 4 iteration steps */
                  force();        /* predictor */
                  grad_1st();     /* correctors in the first point */
                  force();        /* predictor */
                  grad_2nd();     /* correctors in the second point */
            }
            prop();               /* new velocities and positions */
            energy();             /* total energy  */
      }
}
```

Figure 2. Simplified C-code for the main program of the GLRK algorithm.

```
main()
{
      int k;

      init_state();          /* initial structure */
      norm_mode();           /* normal modes of vibration */
      split();               /* select internal from Cartesian coor. */
      convers_in();          /* transf. from internal to normal coor. */

      for (k=0; k<No_steps; k++) {  /* main loop */
            evol();          /* half step vibration */
            convers_ni();    /* transf. from normal to internal coor.*/
            momin();         /* moment of inertia */
            tranrot()        /* half step translation and rotation */
            merge();         /* transf. to Cartesian coor.*/
            cfforce();       /* centrifugal forces */
            grad();          /* nonbonded forces and
                                full step propagation of velocities */
            split();         /* transf. from Cartesian to internal coor. */
            convers_in();    /* transf. from internal to normal coor. */
            evol();          /* half step vibration */
            momin();         /* moment of inertia */
            tranrot();       /* half step translation and rotation*/
            convers_ni();    /* transf. from normal to internal coor. */
            merge();         /* transf. to Cartesian coor. */
            energy();        /* total energy */
      }
}
```

Figure 3. Simplified C-code for the main program of the SISM algorithm.

vibrations by evolving the internal normal coordinates and momenta. The procedures *convers–ni* transform internal normal coordinates and momenta into internal displacement coordinates and momenta. The procedure *momin* determines the moments of inertia of the molecules. The procedure *tranrot* translates and rotates the molecules for half a step. The procedure *merge* transforms the internal displacement coordinates and momenta and the coordinates and momenta of the centers of masses back to the Cartesian coordinates and momenta. In the procedure *cfforce* the centrifugal forces are calculated. The procedure *grad* derives the nonbonding forces due to the Coulomb and Lennard-Jones potentials. Procedures *split*, *convers–in*, *evol*, *momin*, *tranrot*, *convers–ni*, and *merge* are the same as those above. The main loop is concluded by the procedure *energy*, which calculates the total energy of the system (see ref 32 for details).

3. COMPARISON OF PERFORMANCES

So far the sequential methods have been considered. It was shown elsewhere[3,9,33,34] that for an ensemble of N particles there exist several implementations of MD algorithms. In this paper the parallel algorithm described in ref 9 is utilized for all three methods presented in the previous section.

N denotes the number of molecules, n is the number of atoms in each molecule, and p the number of processing nodes connected in the ring network. Particles are divided evenly among the available processors. Each processor manages Nn/p local particles. The step calculation is divided into $p/2$ passes. In the first pass, the calculation of local interactions is reduced using the self-interactions $f_{ii} = 0$ and symmetry of the pair forces $f_{ij} = -f_{ji}$. In the consequent $\lfloor p/2 \rfloor - 1$ passes, the copy of step particle positions is

transferred in parallel to the nearest clockwise neighbor in the processor ring, and the new pass interactions are calculated in all processors. The complete number of interaction calculations in each processor per step is

$$\text{calc} = \frac{Nn}{p}\left(\frac{(Nn/p) - 1}{2} + \frac{\lfloor p/2 \rfloor Nn}{p}\right) \approx \frac{(Nn)^2}{2p^2}(p + 1) \quad (21)$$

for $p > 1$. The partial forces are summed at each pass into mirror force buffers and are transferred at the end of each pass to the counterclockwise neighbors (pass $-$ 1) apart. In order to comprise all interactions, $\lceil p/4 \rceil(\lceil p/2 \rceil - 1)$ transfers of the mirror force buffer are required. The number of buffer communications on each processor per step is

$$\text{comm} = (\lceil p/2 \rceil - 1)\left(1 + \frac{\lceil p/2 \rceil}{2}\right) \approx \frac{p^2}{8} + \frac{p}{4} - 1 \quad (22)$$

for $p > 1$. The parallel algorithm is scalable with the number of processors p and has the time requirements practically proportional to $(Nn)^2/(2p)$ if Nn/p is large enough. The calculation behavior may be influenced also by the implemented cutoff method.[35] In our analysis we assumed a spherical cutoff method with the estimated reduction of the nonbonding interactions equal to the ratio (cutoff sphere volume)/(box volume) which is approximately $2(r_{\text{coff}}/L_{\text{box}})^3$. See ref 17 for details.

The MD is a floating point extensive program; therefore, only the floating point calculation is considered, and data and program code manipulations are supposed to run in parallel. The equations for the calculation time were extracted directly form the implemented program code; additionally, the CPU time was also measured. It was assumed that the average computation time for addition, subtraction, and multiplication is 1f and for division, square root, trigonometric function, and complex test is 2f. On the basis of measurements on a cluster of HP C180s connected with the Fast Ethernet and on the distributed memory computer Cray T3E, we further assumed that the time to exchange a single Floating Point Data Item (FPI) between neighboring nodes is equal to 6f. Each processing node has to communicate concurrently with two neighbors. Step computation always precedes step communication. The amount of memory to be able to store a single FPI is denoted by m. Usually 8 bytes suffices to represent enough precision for an FPI calculation in an MD program.

LFV. In the parallel LFV algorithm a local copy of position buffers is made on each processor for N/p molecules. All procedures except *energy* and *grad* can be implemented locally, on local subsets of particles, with no interprocessor communication. In the procedure *grad* the described parallel method of force calculation is applied. The same method for the calculation of interparticle interactions is used also in the procedure *energy*. According to the above explanation and the analysis of implemented program code, the LFV time step can be expressed as the sum of calculation and communication time

$$T^{\text{LFV}} = \frac{fNn}{p}\left(\left(2\left(\frac{r_{\text{coff}}}{L_{\text{box}}}\right)^3(p + 1)\left(\frac{30Nn}{p} - 19n + 53\right) + 78 - \frac{24}{n}\right) + 6m\left(\frac{p^2}{8} + \frac{p}{4} - 1\right)\right)$$

To implement the above algorithm, the memory buffer of length $3Nm/p$ for the coordinates of the mass centers for all molecules is needed. Additionally, four buffers of length $3Nnm/p$ for the Cartesian coordinates, velocities, local forces, and pair forces for all atoms are necessary. Finally, a copy of the step position buffer and mirror force buffer of length $3Nnm/p$ have to be provided for all atoms in order to be transferred to the neighboring processors. The memory requirement for the implementation of the LFV algorithm, at each processor, equals

$$M^{\text{LFV}} = \frac{Nm}{p}(18n + 3)$$

GLRK. According to the algorithm given in the previous section, the time step for the GLRK algorithm is similar to that given above except that the sum of partial forces must be calculated eight times at each time step. For the same accuracy, the GLRK method permits use of a two times larger time step as in the LFV method.[17] In further analysis the GLRK algorithm is omitted since it behaves similar to the LFV algorithm, except that for the same functional results it consumes approximately four times more computational time and double the amount of memory space in comparison with the LFV algorithm.[18]

SISM. Even if the complexity of the SISM seems to be high compared to that of the LFV algorithm, it is not so since the most demanding part of the SISM, i.e. the calculation of nonbonding forces and energy, is the same as that for the LFV. All the extra work (coordinate transformations) is thus prevailed by nonbonding force and energy calculation.[20] One SISM step requires

$$T^{\text{SISM}} = \frac{fNn}{p}\left(\left(2\left(\frac{r_{\text{coff}}}{L_{\text{box}}}\right)^3(p + 1)\left(\frac{30Nn}{p} - 19n + 53\right) + 17n + 184 + \frac{97}{n}\right) + 6m\left(\frac{p^2}{8} + \frac{p}{4} - 1\right)\right)$$

calculation and communication time.

To implement the above algorithm, five memory buffers of length $3Nm/p$ for the Cartesian coordinates and velocities of the centers of masses, unit vectors of direction, angular velocities, and momenta are needed for all molecules. Additionally, four buffers of length $3Nnm/p$, for the Cartesian coordinates and momenta, local forces, and pair forces for all atoms are necessary. Furthermore, the internal normal coordinates and momenta and the internal displacement coordinates and momenta require four separate buffers of length Nnm/p. Finally, a copy of a step position and mirror force buffer of length $3Nnm/p$ are needed in order to be transferred to the neighboring processors. The complete memory requirement for the implementation of the SISM per each processor equals

$$M^{\text{SISM}} = \frac{Nm}{p}(22n + 15)$$

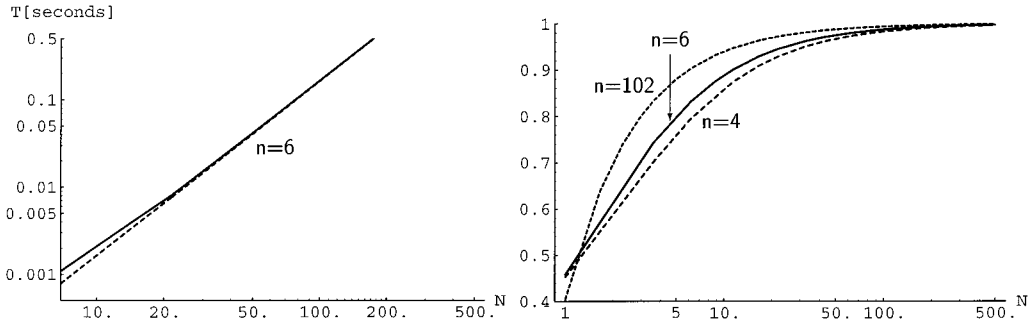

Figure 4. (Left panel) LFV (dotted curve) and SISM (solid curve) time step as a function of N for $f = 15$ ns, $p = 1$, $n = 6$, and not cutoff. (Right panel) Ratio $T^{\mathrm{LFV}}/T^{\mathrm{SISM}}$ for different values of n.

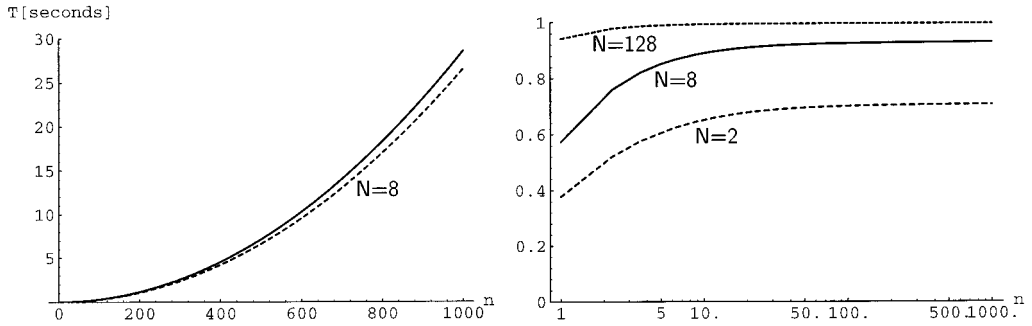

Figure 5. (Left panel) LFV (dotted curve) and SISM (solid curve) time step as a function of n for $f = 15$ ns, $N = 8$, $p = 1$, and no cutoff. (Right panel) Ratio $T^{\mathrm{LFV}}/T^{\mathrm{SISM}}$ for different values of N.

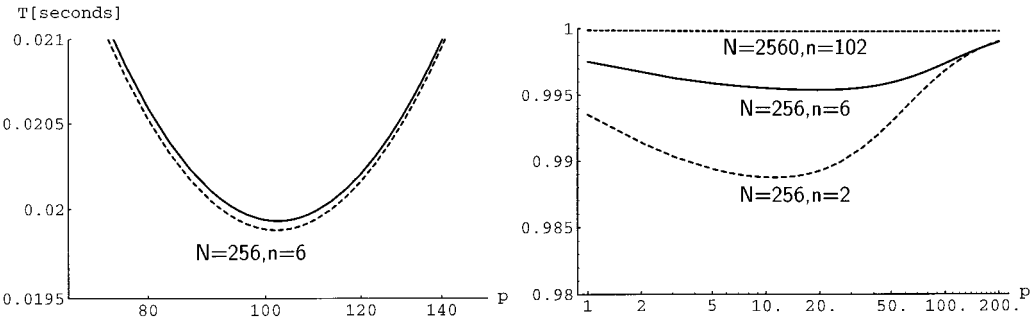

Figure 6. (Left panel) LFV (dotted curve) and SISM (solid curve) time step as a function of p for $f = 15$ ns, $N = 256$, $n = 6$, $m = 6$ and no cutoff. (Right panel) Ratio $T^{\mathrm{LFV}}/T^{\mathrm{SISM}}$ for different values of N and n.

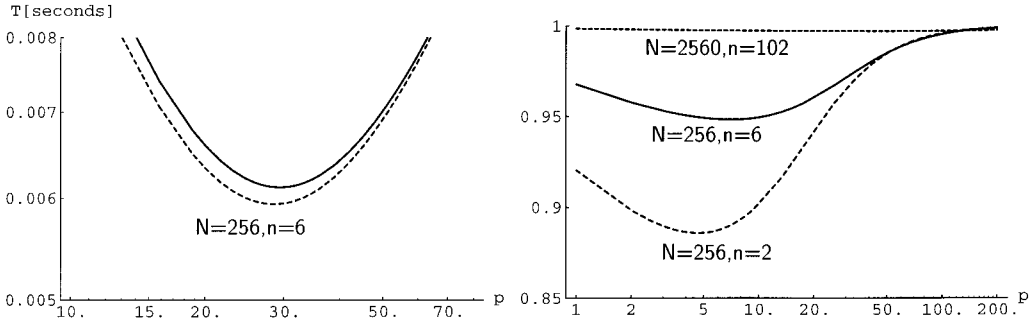

Figure 7. Same as Figure 6 except the spherical cutoff implementation is applied for $L_{\mathrm{box}} = 36$ Å and $r_{\mathrm{coff}} = 12$ Å.

The graphical presentation of the above time complexity equations, including computation and communication time, is shown in Figures 4−7.

Figure 4 shows that the SISM is more complex as the system consists of a smaller number of molecules. It turns out that when the number of molecules in the system exceeds 100, the complexities for both methods are the same.

Figure 5 demonstrates that the number of atoms in the molecule has almost no impact on the performances of the SISM. The LFV, even so simple, appears to be preferable only when smaller systems are of interest.

From the results shown in Figure 6 it is evident that the optimal number of processors for LFV and SISM algorithms is approximately 100 for the chosen number of particles. From the right hand picture it is evident that the LFV gives better results for smaller systems, in particular, for the smaller number of processors. In the example of $N = 2560$ and $n = 102$ the time step complexity is practically the same for both methods since the calculation of interactions dominates.

The results displayed in Figure 7 show that if the cutoff is implemented, the complexity of SISM increases in comparison with LFV. The optimal number of processors

Table 1. LFV and SISM CPU Time per Step (s) for MD Simulations of System of $n = 6$, $N = 128$, and $N = 256$ with No Cutoff Performed on Different Computers

target system	T^{LFV} $N = 128$	T^{SISM} $N = 128$	T^{LFV} $N = 256$	T^{SISM} $N = 256$
Pentium/133	3.13	3.47	12.98	13.02
SGI R8000/75	0.38	0.39	1.52	1.54
HP C180/180	0.35	0.36	1.39	1.42
HP 735/100	0.49	0.50	1.99	1.99
SUN UltraSparc	0.40	0.40	1.62	1.62
T3E/275/1PE	0.28	0.28	1.15	1.15
T3E/275/16PE	0.018	0.019	0.076	0.076
T800/20/1PE	11.88	11.97	47.43	47.43
T800/20/16PE[a]	0.81	0.81	3.27	3.28

[a] Denotes sixteen transputers running on a 20 MHz clock and connected in a ring network.

for LFV and SISM algorithms decreases and is approximately 30 for the chosen number of particles. Apparently, when the cutoff is implemented, the computational complexity decreases more than the communication complexity of the algorithms. However, proportions among the curves on the right picture of Figure 7 are similar to those of Figure 6 except an increased advantage is gained if LFV is applied to smaller systems and a smaller number of processors.

4. MEASURED RESULTS

In the compilation process the maximal time optimization was always applied. Parallel programs were coded on the single program multiple data principle, based on the algorithm described in ref 9. The parallelization was performed on a ring connected message passing computer with p processor nodes of the same processing power and the same memory capacity. Particles have been divided evenly among the available processors. The fastest possible way for the interprocessor communication was always applied.

Table 1 summarizes the measured results of the simulation time per step of the same system in CPU seconds for LFV and SISM on sequential and parallel types of computers. In particular, we measured the complete molecular dynamics program execution time for a 50 ps simulation time for $N = 128$ and $N = 256$ molecules consisting of 6 atoms, resulting in 768 and 1536 atoms, respectively. The results obey the theoretical prediction in that the time step is proportional to $(Nn)^2/(2p)$.

5. CONCLUSIONS

The present work examined three different integration methods for MD simulation in parallel. Since the same molecular system was employed and the same potential function was used in all of the calculations, a direct comparison of all the methods was possible. Through the methods presented, insight can be gained into different parallel MD algorithms.

The parallel LFV algorithm, the second-order MD algorithm, for long-range interactions is simple to implement and gives inherently a nearly ideal speed-up. In practice, if $Nn \gg p$ and $Nn/p \geq 50$, the time T^{LFV} is practically proportional to $(Nn)^2/(2p)$ in comparison with $(Nn)^2/2$ for the sequential version of the LFV algorithm. Also the parallel implementation of the spherical cutoff method is discussed.

The parallel GLRK, the fourth-order MD algorithm, offers a similar speed-up. For $Nn \gg p$ and $Nn/p \geq 50$ the time T^{GLRK} is practically proportional to $8(Nn)^2/(2p)$, while the time complexity of the sequential algorithm is p-times greater. The normalized enlargement of time complexity compared with the LFV method is then a constant equal to 4. The reduction of the time complexity using the spherical cutoff implementation is estimated.

The parallelization of the SISM, the second-order MD algorithm, is following the same principles as those for the previous methods. The algorithm is more powerful in that it employs substantially larger time steps than can be used by other methods of the same order. The calculation complexity and memory requirements are modestly larger as for LFV, in particular for systems consisting of 500 atoms and less. It was shown that SISM offers better performances in comparison with LFV and GLRK for both sequential and parallel implementation.

C-coded source programs are available from the authors upon request.

ACKNOWLEDGMENT

The authors thank Professor D. Hadži for critical reading of the manuscript. This work was supported by the Ministry of Science and Technology of Slovenia under Grants No. J2-5092-106-97 and J1-7346-104-97.

REFERENCES AND NOTES

(1) Heerman, D. W. *Computer Simulation Methods in Theoretical Physics*; Springer: New York, 1986.
(2) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Clarendon Press: Oxford, U.K., 1987.
(3) Hwang, Y. S.; Das, R.; Saltz, J. H.; Hodošček, M.; Brooks, B. R. Parallelizing Molecular Dynamics Programs for Distributed Memory Machines. *IEEE: Comput. Sci. Eng.* **1995**, *2*, 18−29.
(4) Schlick, T.; Barth, E.; Mandziuk, M. Biomolecular Dynamics at Long Timesteps: Bridging the Timescale Gap Between Simulation and Experimentation; Stroud, R. M., Ed. *Annu. Rev. Biophys. Biomol. Struct.* **1997**, *26*, and references cited therein.
(5) Leimkuhler, B. J.; Reich, S.; Skeel, R. D. Integration Methods for Molecular Dynamics. In *IMA Volumes in Mathematics and its Applications*; Mesirov, J., Schulten, K., Eds.; Springer-Verlag: Berlin, 1995; Vol. 82.
(6) Verlet, L. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.* **1967**, *159*, 98−103.
(7) Pastor, R. W.; Brooks, B. R.; Szabo, A. An Analysis of the Accuracy of Langevin and Molecular Dynamics Algorithms. *Mol. Phys.* **1988**, *65*, 1409−1419.
(8) Watanabe, M.; Karplus, M. Dynamics of Molecules with Internal Degrees of Freedom by Multiple Time-Step Methods. *J. Chem. Phys.* **1993**, *99*, 8063−8074.
(9) Trobec, R.; Jerebic, I.; D. Janežič, D. Parallel Algorithm for Molecular Dynamics Integration. *Parallel Comput.* **1993**, *19*, 1029−1039.
(10) Peskin, C. S.; Schlick, T. Molecular Dynamics by the Backward Euler Method. *Comm. Pure. Appl. Math.* **1989**, *42*, 1001−1031.
(11) Nyberg, A. M.; Schlick, T. On Increasing the Time Step in Molecular Dynamics. *Chem. Phys. Lett.* **1992**, *98*, 538.
(12) Derreumaux, P.; Zhang, G.; Schlick, T.; Brooks, B. R. A Truncated Newton Minimizer Adapted for CHARMM and Biomolecular Applications. *J. Comput. Chem.* **1994**, *15*, 532−555.
(13) Sanz-Serna, J. M. Runge-Kutta Schemes for Hamiltonian Systems. *BIT* **1988**, *28*, 877−883.
(14) Arnold, V. I. *Mathematical Methods of Classical Mechanics*; Springer: New York, 1978.
(15) Janežič, D.; Orel, B. Implicit Runge−Kutta Method for Molecular Dynamics Integration. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 252−257.
(16) Janežič, D.; Orel, B. Improvement of Methods for Molecular Dynamics Integration. *Int. J. Quantum. Chem.* **1994**, *51*, 407−415.
(17) Janežič, D.; Trobec, R. Parallelization of an Implicit Runge−Kutta Method for Molecular Dynamics Integration. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 641−646.
(18) Trobec, R.; Janežič, D. Comparison of Parallel Verlet and Implicit Runge−Kutta Methods for Molecular Dynamics Integration. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 100−105.

(19) Janežič, D.; Merzel, F. An Efficient Symplectic Integration Algorithm for Molecular Dynamics Simulations. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 321−326.

(20) Janežič, D.; Merzel, F. Molecular Dynamics Simulations of Macromolecules: Improvements on Computational Efficiency. In *Proceedings of the International Workshop PARALLEL NUMERICS '96*; Trobec, R., Vajteršič, M., Zinterhof, P., Šilc, J., Robič, B., Eds.; Institute Jožef Stefan: Ljubljana, Slovenia, **1996**; pp 17−27 (ISBN 86-80023-25-6).

(21) Janežič, D.; Merzel, F. Simulations of Macromolecules: Improvements of MD Efficiency. *Prog. Biophys. Mol. Biol.* **1996**, *65* (S1), 48.

(22) van Gunsteren, W. I.; Berendsen, H. J. C. *Groningen Molecular Simulation (GROMOS) Library Manual*; Biomos: Groningen, 1987.

(23) Koller, J.; Merzel, F.; Hadži, D. Charge Populations of and Water Binding at the Oxygen Atoms of Some Esters. *J. Mol. Struct.* **1993**, *300*, 233−238.

(24) Sanz-Serna, J. M.; Calvo, M. P. *Numerical Hamiltonian Problems*; Chapman and Hall: London, 1994.

(25) Sanz-Serna, J. M. Symplectic Integrators for Hamiltonian Problems: An Overview. *Acta Numerica* **1991**, 243−286.

(26) Butcher, J. *The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta Methods and General Linear Methods*; Wiley: Chichester, U.K., 1987.

(27) Goldstein, H. *Classical Mechanics*; Addison-Wesley: Reading, MA, 1965.

(28) Yoshida, H. Recent Progress in the Theory and Application of Symplectic Integrators. *Celestial Mechanics and Dynamical Astronomy* **1993**, *56*, 27−43.

(29) Brooks, B. R.; Janežič, D.; Karplus, M. Harmonic Analysis of Large Systems: I. Methodology. *J. Comput. Chem.* **1995**, *16*, 1522−1542.

(30) Janežič, D.; Brooks, B. R. Harmonic Analysis of Large Systems: II. Comparison of Different Protein Models. *J. Comput. Chem.* **1995**, *16*, 1543−1553.

(31) Janežič, D.; Venable, R. M.; Brooks, B. R. Harmonic Analysis of Large Systems. III. Comparison with Molecular Dynamics. *J. Comput. Chem.* **1995**, *16*, 1554−1566.

(32) Janežič, D.; Merzel, F. Split Integration Symplectic Method for Molecular Dynamics Integration. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 1048−1054.

(33) Bernard, E. P.; Plateau, B.; Trystram, D. Using Threads for Developing Parallel Applications: Molecular Dynamics as a Case Study. In *Proceedings of the International Workshop PARALLEL NUMERICS '96*; Trobec, R., Vajteršič, M., Zinterhof, P., Šilc, J., Robič, B., Eds.; Institute Jožef Stefan: Ljubljana, Slovenia, 1996; pp 3−16 (ISBN 86-80023-25-6).

(34) Brooks, B. R.; Hodošček, M. Parallelization of CHARMM for MIMD Machines. *Chem. Des. Autom. News* **1992**, *7* (12), 16−22.

(35) Steinbach, P. J.; Brooks, B. R. New Spherical-Cutoff Methods for Long-Range Forces in Macromolecular Simulation. *J. Comput. Chem.* **1994**, *15*, 667−683.

CI970226U