Comparison of Parallel Verlet and Implicit Runge−Kutta Methods for Molecular Dynamics Integration

Roman Trobec

Jožef Stefan Institute, University of Ljubljana, Jamova 39, 61000 Ljubljana, Slovenia

Dušanka Janežič*

National Institute of Chemistry, Hajdrihova 19, 61115 Ljubljana, Slovenia

Received June 27, 1994[⊗]

Parallel implementation issues of the explicit Verlet-type (VT) and the implicit Runge−Kutta (RK) methods for molecular dynamics integration (MD) on parallel distributed memory processors with the ring topology are described and compared. Both methods are applied to a system of *N* particles interacting through the Lennard-Jones potential. The RK method is carried out for the two-stage fourth-order Gauss−Legendre scheme. The parallel algorithm, memory requirements, and the complexity estimation were performed for both methods and compared. Time results were measured on different computers for comparison.

1. INTRODUCTION

The extension of computer simulation methods to molecular systems of increasing size and complexity needs a conceptual improvement of the existing sequential or parallel algorithms. Well-known explicit molecular dynamics integration method of the Verlet-type[1,2] and a new implicit symplectic Runge−Kutta integration method[3,4] are two competitive schemes for Hamiltonian systems.

The major obstacle that reduces the efficiency of computer simulation methods of complex molecular systems lies in the inability to sample sufficient portions of the phase space. Most of the proposed molecular dynamics algorithms presently used for solving Hamiltonian systems exhibit instability unless the time step is small enough.[5] The enlarging of the time step in the molecular dynamics integration can be partially overcome by the use of implicit methods for the numerical solution of Hamilton equations.[6,7] Hamiltonian systems possess an important property, in that the flow in the phase space is symplectic, thus implying the existence of the phenomenon of the Poincaré recurrence.[8] Therefore, the numerical methods for solving these systems are expected to reproduce this property.[3,9] A numerical method is called symplectic if it generates numerical solutions that inherit the property of symplectivity. Specific Runge−Kutta (RK) methods appear to be, in this context, the most promising ones. Because of difficult implementation, particular RK methods have to be designed in a less demanding computational way while retaining symplectivity. It was shown that such methods do exist,[4] for example, the *s*-stage Gauss−Legendre Runge−Kutta (GLRK) method of order 2*s*.

The basic idea of computer simulation based on the sequential technique[1] was further devised and implemented in the parallel Verlet-type (VT) molecular dynamics program, described in detail in ref 10. The theoretical background of the implicit *s*-stage Gauss−Legendre Runge−Kutta methods of order 2s with *i* fixed-point iterations for solving the resulting nonlinear system of equations (*si*GLRK) for molecular dynamics (MD) integration, and the evaluation of the two-stage fourth-order GLRK method is given in ref 11. For brevity we will denote *si*GLRK as RK. Both parallel algorithms (VT and RK) have been tested on a distributed memory computer (DMC), more specifically, on a ring of 20 MHz INMOS transputers T800. Instead of the usual generic programming language *Occam*, the Parallel Computational Library (PCL)[12] in connection with 3L's *Parallel C* was used for programming.

In this paper we compare parallel Verlet and implicit Runge−Kutta methods for molecular dynamics integration. In section 2 the underlying theory of the Verlet-type method, and the implicit Runge−Kutta method is given. In section 3 the basic ideas of both parallel algorithms for molecular dynamics integration are analyzed. The memory requirements of the distributed memory computer are investigated; the theoretical time complexity and the optimal number of processors are devised. In section 4 the measured results for both methods on sequential and parallel type of computers are presented. In Conclusions, some possible improvements are pointed out.

2. THEORY

The Model. The evaluation model, a monoatomic system of *N* particles in a cube of side *L*, was chosen to give realistic merits of numeric MD methods efficiency. The lengths are expressed in units of σ (σ = 3.405 Å for argon), and the energies in units of ϵ (ϵ = 119.8° K for argon).[1,2] The periodic boundary conditions were imposed to conserve the number of particles. To perform the MD simulation of such a system the forces acting on each particle have to be calculated for each time step *h*. The equation of motion

$$m_i\ddot{\mathbf{r}}_i = \sum_{j\neq i}\mathbf{f}(\mathbf{r}_{ij}) \qquad (1)$$

for *N* particles with masses m_i interacting through the Lennard-Jones potential has to be numerically integrated. The pairwise forces at the distance r_{ij} in the *x* direction can be expressed as

* Author to whom correspondence should be addressed.

⊗ Abstract published in *Advance ACS Abstracts,* December 1, 1994.

$$\mathbf{f}_x(r_{ij}) = 48\frac{\epsilon}{\sigma^2}(x_i - x_j)\left[\left(\frac{\sigma}{r_{ij}}\right)^{14} - \frac{1}{2}\left(\frac{\sigma}{r_{ij}}\right)^{8}\right] \qquad (2)$$

and, similarly, for the y and z direction. All the equations in computer simulation were rendered dimensionless by scaling time and positions by $(m\sigma^2/48\epsilon)^{1/2}$ and σ, respectively (m is the mass of argon atom; m = 39.948 a.u.). The time unit for argon is 3×10^{-13} s. The time step was taken to be h = 0.064 or 2×10^{-14} s.

The number of all interactions is $N(N - 1)/2$. The reduction of the calculation time is possible by introducing the cutoff distance (r_{co}). The calculation is then performed as the distance between two particles is less than r_{co}. In the parallel implementation which we proposed in our previous paper[11] all distances have to be computed since there is no global knowledge of particle positions. Consequently, a nearly constant and minor speed-up of two for both methods has been measured (see ref 11). Since more advanced methods, as proposed in ref 11, imply a complicated parallelization and unbalanced load, and because the cutoff is not permitted in some complicated models, we do not consider such calculation. In the first *istop* steps (*istop* denotes the number of steps to reach equilibrium) in every *irep*th step the global kinetic energy (for a unit mass m_i = 1)

$$E_{kin} = \frac{1}{2}\sum_{i=1}^{N}\mathbf{v}_i^2 \qquad (3)$$

has to be calculated in order to scale velocities[2] by β

$$\beta = \left(\frac{T^*(N-1)}{16\sum_i \mathbf{v}_i^2}\right)^{1/2} \qquad (4)$$

Varlet-type Method. Integration of the equation of motion is performed at the beginning of each calculation step, when the sum of partial forces $\mathbf{f}(r_{ij})$ is already calculated. The basis for the integration is the Verlet-type algorithm

$$\mathbf{r}_i(t + h) = \mathbf{r}_i(t) + h\mathbf{v}_i(t) + \frac{h^2}{2}\sum_{j\neq i}\mathbf{f}(r_{ij}(t)) \qquad (5)$$

where h is the time-step.

At each time-step the new velocities are calculated by expressions

$$\mathbf{v}_i\left(t + \frac{h}{2}\right) = \mathbf{v}_i(t) + \frac{h}{2}\sum_{j\neq i}\mathbf{f}(r_{ij}(t)) \qquad (6)$$

$$\mathbf{v}_i(t + h) = \mathbf{v}_i\left(t + \frac{h}{2}\right) + \frac{h}{2}\sum_{j\neq i}\mathbf{f}(r_{ij}(t + h)) \qquad (7)$$

Implicit Runge–Kutta Method. Equation of motion can be, equivalently, written as a system of first-order differential equations

$$\dot{\mathbf{r}} = \mathbf{v} \qquad (8)$$

$$\dot{\mathbf{v}} = \mathbf{f}(\mathbf{r}) \qquad (9)$$

The system (8) and (9) is Hamiltonian since the force is the gradient of the potential $-V(\mathbf{r})$

$$\mathbf{f} = -\nabla V(\mathbf{x}) \qquad (10)$$

A general s-stage implicit RK method for the solution of a Hamiltonian system

$$\mathbf{q}' = \mathbf{F}(\mathbf{q}) \qquad \mathbf{q}(0) = \mathbf{q}_0 \qquad (11)$$

is of the form

$$\mathbf{Q}_i = \mathbf{q}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{F}(\mathbf{Q}_j) \qquad i = 1, ..., s \qquad (12)$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\sum_{i=1}^{s} b_i\mathbf{F}(\mathbf{Q}_i) \qquad (13)$$

where $\mathbf{q}$ is a d dimensional vector, determined by the Butcher's tableau[14]

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

of the $s \times s$ matrix $A = [a_{ij}]_{i,j=1}^{s}$, and the s-dimensional vectors $\mathbf{b} = [b_i]_{i=1}^{s}$ and $\mathbf{c} = [c_i]_{i=1}^{s}$ where $c_i = \sum_{j=1}^{s} a_{ij}$ and h the time step. Equation 12 implicitly defines intermediate stages $\mathbf{Q}_i$ which can be interpreted as an approximation to the exact solution $\mathbf{q}(t)$ at the intermediate point $t = t_n + c_i h$. Since the coefficients of the s-stage RK method of order $2s$ satisfy the relations

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \qquad 1 \le i, j \le s \qquad (14)$$

the method is symplectic.[3]

The application of the two-stage fourth-order RK to MD integration problem (eqs 8 and 9) leads to the following relations which implicitly define the intermediate stages (point values) $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{R}_1$, and $\mathbf{R}_2$

$$\mathbf{V}_1 = \mathbf{v}_n + h\,(a_{11}\mathbf{f}(\mathbf{R}_1) + a_{12}\mathbf{f}(\mathbf{R}_2))$$

$$\mathbf{V}_2 = \mathbf{v}_n + h\,(a_{21}\mathbf{f}(\mathbf{R}_1) + a_{22}\mathbf{f}(\mathbf{R}_2))$$

$$\mathbf{R}_1 = \mathbf{r}_n + h\,(a_{11}\mathbf{V}_1 + a_{12}\mathbf{V}_2)$$

$$\mathbf{R}_2 = \mathbf{r}_n + h\,(a_{21}\mathbf{V}_1 + a_{22}\mathbf{V}_2) \qquad (15)$$

where $\mathbf{V}_i$ and $\mathbf{R}_i$ correspond to $\mathbf{Q}_i$ from eq 12 and $\mathbf{f}$ to the force in eq 10.

The new step velocities and the new step positions are given finally in accordance to eq 13

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{h}{2}(\mathbf{f}(\mathbf{R}_1) + \mathbf{f}(\mathbf{R}_2))$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \frac{h}{2}(\mathbf{V}_1 + \mathbf{V}_2) \qquad (16)$$

The procedure is repeated for the desired number of steps.

3. COMPARISON OF PARALLELIZED METHODS

So far the sequential methods have been considered. A proven concept of parallelization has to lead to a balanced partition of calculations among interconnected processors. Moreover, if the computation time is much greater than the communication time, the parallel time complexity is reduced

```
par_VT()                          /* parallel VT MD algorithm */
{
 for (step=0; step<N_steps; step++){
   calc_new_position(N/p);             /* Eq. (5) */
   copy_pos(N/p);                      /* copy local step pos.*/
   calc_new_velocity(N/p);             /* Eq. (6) */
   clear_forces(N/p);                  /* clear force buffers */
   calc_all_Fij(N/p,p,pass);           /* Eq. (2) */
   calc_new_velocity(N/p);             /* Eq. (7) */
   calc_kin_energy(N/p,step,Tscale); /* Eq. (3,4) */
}}
/***********************************************************/

par_RK()                           /* parallel RK MD algorithm */
{
 for (step=0; step<N_steps; step++){
  set_predictor (N/p) ;               /* set predictor */
  for (iter=0; iter < i; iter++)  {
   for (point=0; point < s; point++) {
     calc_all_Fs(N/p,p);              /* Eq. (2) */
     calc_vel_pos(N/p);               /* Eq. (15) */
  }}
  calc_new_vel_pos(N/p);              /* Eq. (16) */
  calc_kin_energy(N/p,step,Tscale);   /* Eq. (3,4) */
}}
```

Figure 1. The C-like sketched code for the VT and the RK parallel molecular dynamics algorithm.

proportionally to the number of processors p. It was shown that for the system of N particles there exists a parallel implementation of both algorithms that is scalable with p and has the time requirements practically proportional to N^2/p if N/p is large enough.[10,11] The ring topology resolved the calculation and communication requirements in the same time. The particles have been divided evenly among the processors; local interactions have been calculated first. Then the copy of local particle data were sent simultaneously, in the whole ring, to neighboring processors where interactions among "traveling" and local particles were calculated, and so forth until local particles "meet" all other particles.

Algorithm. After the initialization of the ring topology, initial positions, velocities, and force buffers, for each VT algorithm step, new particle step positions **r** and intermediate velocities **v** are calculated. A local copy of the positions $\mathbf{r}^c$ is made, and then the local forces **f** among N/p local particles are calculated. During $\lceil p/2 \rceil$ shifts these forces are updated in order to comprise all interactions (see ref 10 for details). Finally, the new step velocities are calculated and scaled, if necessary.

Due to the ring philosophy, the parallel RK algorithm also becomes local. N/p particles are assigned to each processor, and all data buffers are set to initial values. The setting of initial positions and velocities and the initialization of data buffers are performed once in parallel for all steps. Predictors (forces) are calculated in each step; i iterations for the corrector are performed in s intermediate points which leads to si calculation of forces. For our particular example eight calculations of forces in each step have been implemented. Then new step velocities and positions are calculated. The rest of the code operates on computation of the global kinetic energy and eventual scaling (see ref 11 for details).

The sketched C-like code of the parallel VT algorithm and the parallel RK algorithm for each processor is given in Figure 1.

Memory Requirements. Each processor holds several data buffers for N/p local particles. Additionally, two bidirectional communication buffers are needed in force calculations. For the best performances, it is essential to keep the local variables in the high speed local memory (e.g., registers, internal cache memory); besides, all calculation of indices should be avoided with the use of pointer operations (usually, the compiler will do this job.). All calculations have been performed with single precision floating point numbers which occupy four bytes of memory. Let four bytes be a unit for our analysis, denoted by MU. From the implementation of the parallel algorithms in Figure 1, Table 1 can be devised. Note that the RK method is slightly more demanding with respect to memory requirements; it needs $3sN/p$ more MU units than the VT method does.

Table 1. Memory Requirements for the VT and RK Algorithms in MU Units

VT memory buffers	RK memory buffers
$r[N/p]$ (step position)	$r[N/p]$ (step position)
	$rn[N/p]$ (new step position)
$r^c[N/p]$ (copy of step position)	$r^c[N/p]$ (copy of step position)
$f^m[N/p]$ (mirror force)	$f^m[N/p]$ (mirror force)
$v[N/p]$ (step velocity)	$v[N/p]$ (step velocity)
	$r_j[N/p]\ j = 1...s$ (point position)
	$v_j[N/p]\ j = 1...s$ (point velocity)
$f[N/p]$ (step force)	$f_i[N/p]\ j = 1...s$ (point force)
$5 \times (N/p)$	$5 \times (N/p) + 3s \times (N/p)$

Time Requirements. The time requirements for one calculation step with N particles on p processors have to be compared. The time for communication and the time for processing have been considered. For the communication time an equal communication speed is supposed for all data channels on the ring. Data are transferred always to the nearest neighbors; consequently, the communication can be performed in parallel in the same time intervals. In the processing time, floating point operations (FLOPS), and data/program code manipulation are encompassed. Other internal processor operations (i.e., initialization, timers) represent only a minor part of required time and have been ignored.

The time requirements for one VT step can be expressed as the sum of the time required for communication (T_c), and the time for processing (T_p) (see ref 10 for details):

$$T^{VT} = T_c + T_p = T_{tran}\frac{3N}{p}(\lceil p/2 \rceil - 1)\left(1 + \frac{\lceil p/2 \rceil}{2}\right) + \frac{(T_f + T_i)N}{p}\left(\frac{(N/p) - 1}{2} + \frac{\lfloor p/2 \rfloor N}{p}\right) + \frac{T_{rv}N}{p} \quad (17)$$

where T_{tran} represents the time required for transferring one MU to the neighboring node, T_f represents the time required for calculation of a single force interaction between two particles, T_i represents the time for data/code manipulation in the inner loop of the algorithm, and T_{rv} the time for computation of new position, velocity, and kinetic energy for one particle. Considering the performances of the T800 given in the next section, the graphical presentation of eq 17 is shown in Figure 2 by the dotted curve. At first, the step time T^{VT} decreases rapidly by the increasing number of processors p. Then the optimal number of processor is reached with the minimal step time. As the number of processors is still augmented, the step time increases again because the communication load becomes dominant. As an example, the step time T^{VT} for one processor in the example of $N = 10^4$ particles would be about 2×10^3 s, in contrast to $T^{VT} = 8.7$ s (see Figure 2) which would be the minimal possible time reached by approximately 460 processors in parallel.

The step time for the RK algorithm is again the sum of the time required for communication and the time required for processing (see ref 11 for details)

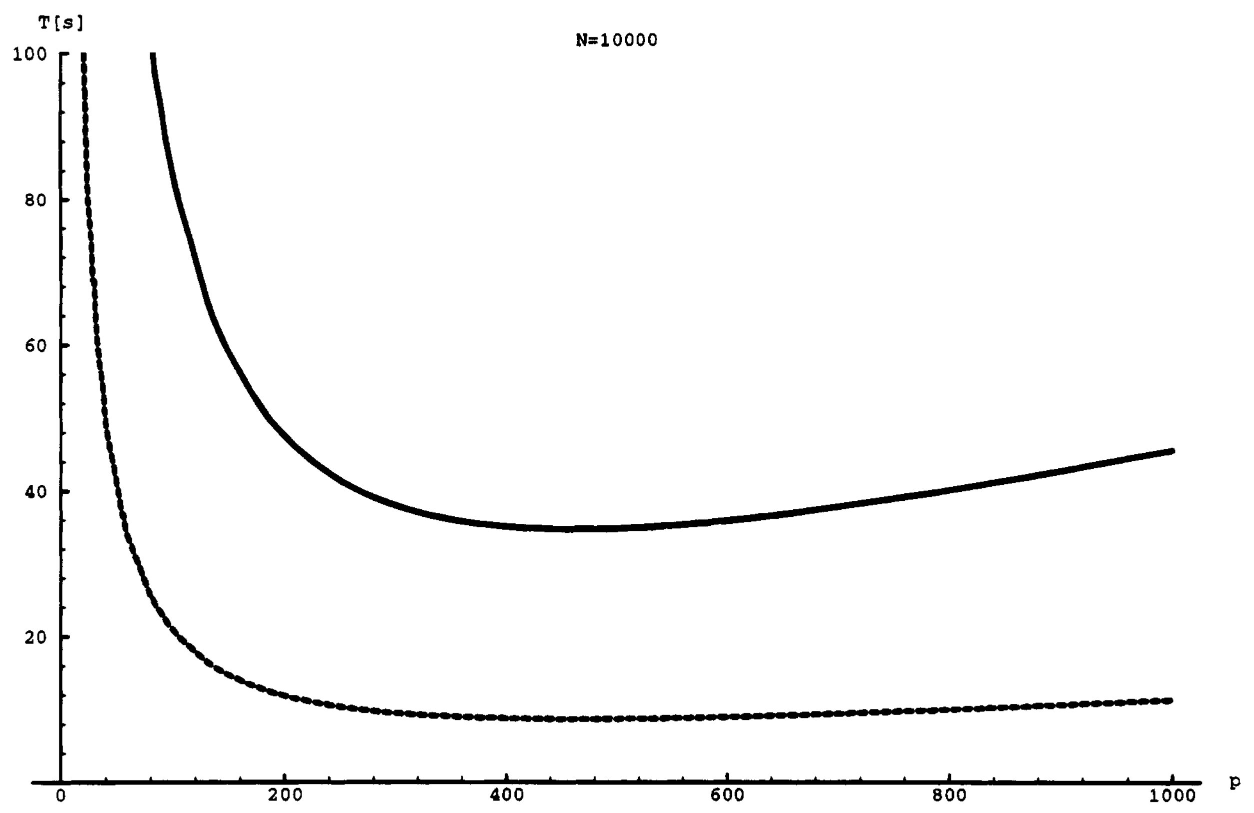

Figure 2. The normalized step time for $N = 10\,000$ particles for the VT (dotted curve) and the RK (solid curve) parallel molecular dynamics program.

$$T^{\mathrm{RK}} = T_{\mathrm{c}} + T_{\mathrm{p}} = T_{\mathrm{tran}}\left(\frac{3Nsi}{p}(\lceil p/2\rceil - 1)\left(1 + \frac{\lceil p/2\rceil}{2}\right) + \lfloor p/2\rfloor\right) + \frac{siN}{p}\left(\left(\frac{N/p - 1}{2} + \frac{\lfloor p/2\rfloor N}{p}\right)(T_{\mathrm{f}} + T_{\mathrm{i}}) + T_{\mathrm{rv}}\right) + \frac{N}{p}(T_{\mathrm{pr}} + T_{\mathrm{nrv}}) \quad (18)$$

where T_{pr} represents the time for calculation of predictor and T_{nrv} the time for calculation of new position, velocity, and kinetic energy for each particle. The other symbols meaning is the same as in eq 17.

According to the experiments given in ref 13, for the same accuracy, the RK method can use a two times larger time step h as VT method. Therefore, in the following the normalized time step $T'^{\mathrm{RK}} = T^{\mathrm{RK}}/2$ will be used for comparison. The graphical presentation of eq 18 for the RK method performed on the T800 ring is given in Figure 2 by the solid curve. The RK time requirement is approximately $si/2$ times greater (four times for our implementation) than in the VT algorithm and has a similar shape as in the VT algorithm.

Optimal Number of Processors. After differentiating the eqs 17 and 18 it is possible to find the minimal T'^{VT} and T'^{RK} as a function of the number of particles N. The number of processors p is said to be optimal for this time step, because the execution time gets the minimal possible value. Usually this solution would not be optimal according to the price/performance criterion. The optimal number of processors and the minimal time step are given on the same diagram. In Figure 3 the plots of the VT method are represented by dotted curves and the plots for the RK method by solid curves. The curves for the optimal number of processors are practically identical for both methods because force calculations are dominant in both methods. The curve representing the minimal step time of the RK method shows approximately four times larger value as the curve for VT step time which is in accordance with previous discussion.

4. MEASURED RESULTS

To make our theoretical results more convincing, some experimental measurements on different sequential and parallel computers have been performed. Table 2 summarizes the measured results in CPU seconds for the simulation of the same system, for the VT method, and the RK method, on sequential and parallel types of computers. In compilation process the maximal time optimization was always applied. The results in Table 2 are in good agreement with the theory for all computers considered. In particular, we measured the complete molecular dynamics program execution time for one step for the examples $N = 256$, $N = 512$, and $N = 1024$. The results for sequential computers obey the theoretical prediction that the time complexity, for both methods, is proportional to $N^2/2$ and that, for the same accuracy, the RK method is four times slower than the VT method. The available parallel computers were a 16 node and an 8 node 20 MHz T800 transputer ring. The comparison of step times for available parallel computers shows that the speed-up is proportional to the number of nodes p, in accord with the theory.

Finally, a test for an explicit check point for both parallel versions of the algorithm (eqs 17 and 18) has been made for $N = 1024$ particles and $p = 16$ nodes. All technical data of interest and timings were found in transputer data sheets. For example, the time required to transfer one MU is $T_{\mathrm{tran}} = 2.5\ \mu$s. A read or write (R/W) operation to the local memory for the same MU lasts 0.2 μs. An average scalar instruction execution time equals to 0.1 μs. The

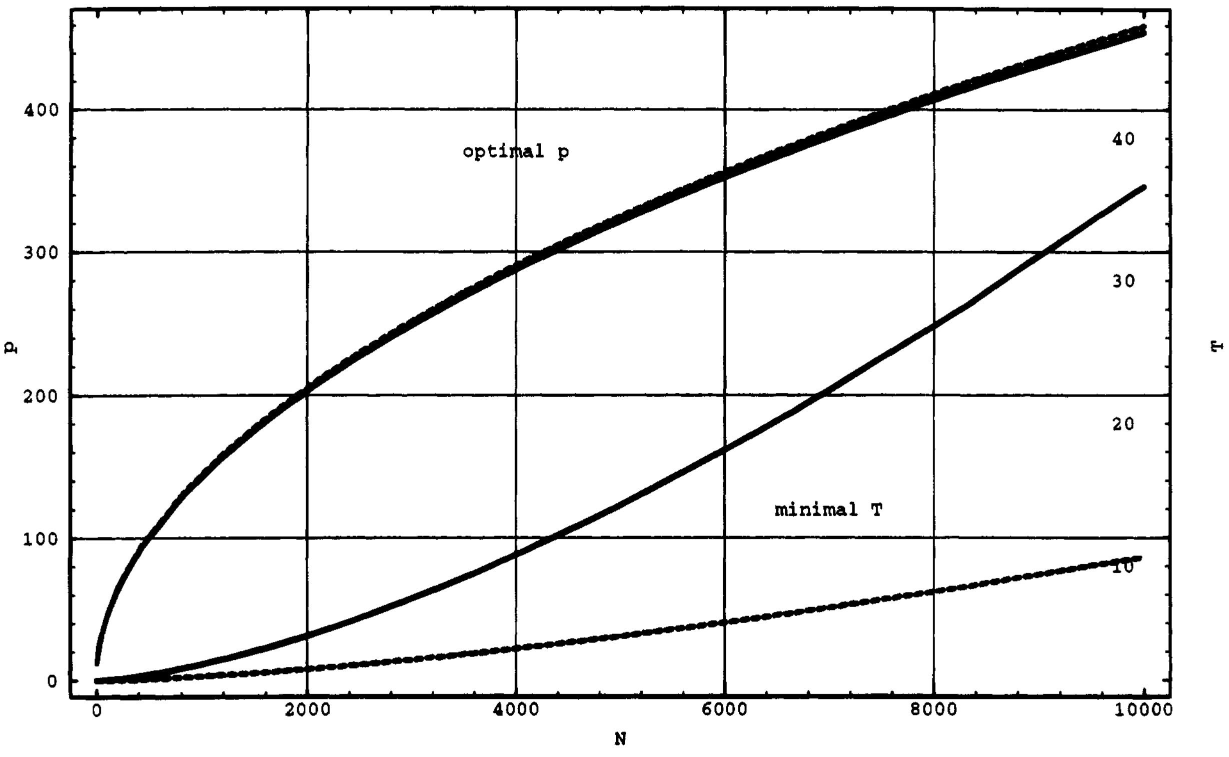

Figure 3. The optimal $p(N)$, minimal T^{VT} (dotted curves), and T^{RK} (solid curves).

Table 2. Ranked Computers for One Normalized Time Step of VT and RK Algorithm

target system	T^{VT} ($N = 256$)	T^{RK} ($N = 256$)	T^{VT} ($N = 512$)	T^{RK} ($N = 512$)	T^{VT} ($N = 1024$)	T^{RK} ($N = 1024$)
1 × T800/20 MHz	1.32	5.7	5.27	22.0	21.47	87.1
8 × T800/20 MHz	0.20	0.83	0.79	3.2	3.07	12.5
16 × T800/20 MHz	0.11	0.45	0.39	1.6	1.43	6.0
Convex C3860	0.07	0.3	0.28	1.13	1.14	4.4
HP 715/75 MHz	0.04	0.18	0.17	0.71	0.73	2.83
HP 735/100 MHz	0.03	0.13	0.13	0.53	0.53	2.11

timings for all FLOPS operations can also be found in the data sheets.

After analyzing the program code for the VT method (180 instruction fetches, 17 R/W of MUs, 11 multiplications, 1 division, etc.) the time for calculation of a single interaction was estimated to be $T_f = 18.4\ \mu s$ and $T_i = 21.4\ \mu s$. In the similar way the time for computation of position and velocity was $T_{rv} = 36.2\ \mu s$. From eq 17 we get the VT step time

$$T^{VT} = 63(192)(2.5\ \mu s) + 34784(18.4 + 21.4\ \mu s) + 64(36.2\ \mu s) = 17.3\ \text{ms} + 1384.4\ \text{ms} + 2.3\ \text{ms} = 1404\ \text{ms} \approx 1.4\ \text{s} \quad (19)$$

The result is in a good agreement with the measured value given in Table 2.

The values of T_f, T_i, and T_{vp} obtained from the analysis of the RK program code are the same as those obtained from VT algorithm, whereas $T_{pr} = 135\ \mu s$ and $T_{nrv} = 18.9\ \mu s$. By means of eq 18 the RK step time equals

$$T^{RK} = 53768(2.5\ \mu s) + 512(543.5(18.4 + 21.4) + 36.2\ \mu s) + 64(135 + 18.9\ \mu s) = 134.4\ \text{ms} + 11093.8\ \text{ms} + 9.8\ \text{ms} = 11238\ \text{ms} \approx 11.2\ \text{s} \quad (20)$$

which is close enough to be measured value from Table 2. Each RK step takes about 11.2 s, approximately eight times more than the VT step, or four times if the normalized step time is considered. In both examples the most of the computation time is spent on force calcualtions.

5. CONCLUSIONS

The present work describes a comparison of parallel Verlet-type and implicit Runge—Kutta symplectic integration methods for MD simulations.

The Verlet-type parallel algorithm for long range interactions is easy to implement and offers inherently a nearly ideal speed-up. In practice, where $N \gg p$ and $N/p \geq 50$ the time T^{VT} is practically proportional to $N^2/(2p)$ in comparison with $N^2/2$ for the sequential version with p-times greater time complexity. Further improvements could be achieved by the parallel implementation of a cutoff criterion.

The parallel two-stage fourth-order RK method offers a similar speed-up. For $N \gg p$ and $N/p \geq 50$ the time T^{RK} is practically proportional to $siN^2/(2p)$, while the time complexity of the sequential algorithm is p-times greater. The normalized enlargement of time complexity compared to the Verlet-type methods is then a constant $si/2$. It is expected that the further reduction of the time complexity could be achieved by a cutoff implementation especially of intermediate point calculations. Much work remains to be done in an improved implementation of the RK method, particularly to get good initial guesses for the solution of the nonlinear equations, and to choose an adequate iterative procedure.

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.

REFERENCES AND NOTES

(1) Verlet, L. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.* **1967**, *159*, 98–103.

(2) Heerman, D. W. *Computer Simulation Methods in Theoretical Physics*; Springer: New York, 1986.

(3) Sanz-Serna, J. M. Runge–Kutta Schemes for Hamiltonian Systems. *BIT* **1988**, *28*, 877–883.

(4) Sanz-Serna, J. M.; Symplectic Runge-Kutta and Related Methods: Recent Results. *Physica D* **1992**, *60*, 293–302.

(5) 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.

(6) Peskin, C. S.; Schlick, T. Molecular Dynamics by the Backward-Euler Method. *Comm. Appl. Math.* **1989**, *42*, 1001–1031.

(7) Fincham, D. Leapfrog Rotational Algorithms. *Mol. Sim.* **1992**, *8*, 165–178.

(8) Arnold, V. I. *Mathematical Methods of Classical Mechanics*; Springer: New York, 1978.

(9) Toxvaerd, S. J. Molecular Dynamics at Constant Temperature and Pressure. *Phy. Rew. E* **1993**, *47*, 343–350.

(10) Trobec, R.; Jerebic, I.; D.Janežič, D. Parallel Algorithm for Molecular Dynamics Integration. *Parallel Computing* **1993**, *19*, 1029–1039.

(11) Janežič, D.; Trobec, R. Parallelization of an Implicit Runge–Kutta Method for Molecular Dynamics Integration. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 641–646.

(12) Jerebic, I.; Slivnik, B.; Trobec, R. Library for Programming on Torus-connected Multiprocessors; Proceedings of the International Conference PACTA '92, IOS Press: Barcelona, 1992; pp 386–395.

(13) Janežič, D.; Orel, B. Implicit Runge–Kutta Method for Molecular Dynamics Integration. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 252–257.

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

CI940078X