Parallelization of an Implicit Runge–Kutta Method for Molecular Dynamics Integration

Dušanka Janežič*

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

Roman Trobec

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

Received July 21, 1993®

A parallelized algorithm of an implicit Runge–Kutta integration scheme, the s-stage Gauss–Legendre Runge–Kutta (GLRK) method of order $2s$ with i fixed-point iterations for solving the resulting nonlinear system of equations, is presented. The algorithm is used for numerical solution of molecular dynamics equations on the distributed memory computers in the ring topology. It is designed for the two-stage fourth-order GLRK method for $i = 4$ and applied to a system of N particles interacting through the Lennard–Jones potential. The theoretical time complexity estimation is performed, and time results are also measured on different computers for comparison. The proposed parallel algorithm is scalable with the number of processors p, and its time requirement is practically proportional to $siN^2/2p$ if N/p is large enough.

I. INTRODUCTION

Demands for the extension of computer simulation methods to molecular systems of increasing size and complexity cannot be met solely by developing better hardware and consuming more computer time; needs for conceptual improvement of the existing algorithms are becoming more and more obvious. Enlargement of the scope of the existing methods of molecular dynamics (MD) of complex molecular systems may be achieved by introducing the implicit symplectic integration schemes for Hamiltonian systems and their parallel implementations on the ring topology for distributed memory computers.

The major obstacle that reduces the efficiency of computer simulation methods of complex molecular systems lays in the inability to sample sufficient portions of phase space. A number of MD algorithms for solving Hamiltonian systems have been proposed.[1–5] Most of the methods presently used exhibit instability unless the time step is small enough.[6] But, for the studies of dynamics of large molecular systems, larger time steps in the MD integration procedure are needed. The problem of increasing the time step in the MD integration can be overcome by the use of implicit methods for the numerical solution of Hamilton equations.[7,8] Hamiltonian systems possess an important property, in that the flow in the phase space is symplectic;[9] hence, the numerical methods for solving these systems are expected to reproduce this property.[10] Specific Runge–Kutta (RK) methods appear to be, in this context, the most promising ones.[11] Such methods are difficult to implement; therefore, special implicit RK methods have to be designed in a less computationally demanding way, while retaining symplecticity. There exist unique s-stage ($s \geq 1$) implicit RK methods of order $2s$, the Gauss–Legendre (GL) schemes, which are symplectic.[12]

Application of the s-stage Gauss–Legendre Runge–Kutta (GLRK) method of order $2s$ to the MD integration leads to the nonlinear system of algebraic equations of dimension $s \times d$ (s being the number of stages of the GLRK method and d being the number of degrees of freedom of the system). For solving such a system of equations the known alternatives are the Newton–Raphson method and the fixed-point iteration. In both cases an initial guess for the starting value of the fixed-point iteration has to be made.

The basic principles for the parallel solution of long-range interaction problems are given in ref 13. The idea based on the fundamental sequential technique[1] was further developed and implemented in the parallel Verlet MD program described in detail in ref 14. The theoretical background of the implicit s-stage Gauss–Legendre Runge–Kutta method of order $2s$ with i fixed-point iterations for solving the resulting nonlinear system of equations (siGLRK) for molecular dynamics (MD) integration and the evaluation of the sequential version of the two-stage fourth-order GLRK method is given in ref 15. The main drawback of the method in comparison to the classical Verlet-type solution for the same reliability was the approximate time amplification for $si/2$. The reason for this lies primarily in the need for multiple force calculations.

In this paper we describe the parallelized siGLRK method for MD integration applied to a complex system of N particles interacting through the Lennard–Jones potential. The method is scalable with the number of processors p and reduces the time requirements practically proportional to $siN^2/2p$ if N/p is large enough; in addition, the time step h can be doubled for the same reliability. Consequently, with a constant greater number of processors $(si/2)p$ the required time for siGLRK calculation is in the same range as that for the fast Verlet-type algorithms executed on p processors.

The parallel algorithm for the two-stage fourth-order GLRK for $i = 4$ (24GLRK) method has been tested on a transputer ring; however, it is generally applicable to other massively parallel computers of MIMD type. Instead of the usual generic programming language *Occam*, the parallel computational library (PCL),[16] in connection with 3L's *Parallel C* was used for programming. PCL gives the user, who is typically not an expert in writing parallel programs, a standard set of high-level primitives for mesh/torus-connected processors.

In section II the physical background, the details of the GLRK method, and the parallelization principle are given. In section III the time complexity of the parallel siGLRK method is analyzed and some theoretical merits are shown. In section IV the measured results, obtained for systems consisting of 256, 512, and 1024 particles on different computer systems are shown and compared. In the Conclusions section the

* Author to whom correspondence should be addressed.

® Abstract published in *Advance ACS Abstracts*, March 15, 1994.

method is analyzed and possible future improvements are discussed.

II. THEORY

II.1. 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).[1,3] 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

$$m_i\ddot{\mathbf{r}}_i = \sum_{j\neq i}\mathbf{f}(r_{ij}) \qquad \text{(II.1.1)}$$

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

$$\mathbf{f}_x(r_{ij}) = (x_i - x_j)(r_{ij}^{-14} - (1/2)r_{ij}^{-8}) \qquad \text{(II.1.2)}$$

and correspondingly for the other directions. 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 interactions are negligible since the distance between two particles is greater than r_{co}. In the example of r_{co} = 2.5, the number of interactions can be reduced to

$$(2\pi r_{co}^3/3L^3)N^2 \approx 0.11N^2 \qquad \text{(II.1.3)}$$

as compared to $0.5N^2$ for the calculation without cutoff. The cutoff tables and other improvements employing the cutoff criterion[1,13] were not considered in the time complexity analysis since they imply complicated parallelization and unbalanced load. Only a simple cutoff threshold was applied for the results given in Table 3; therefore, distances r_{ij} have to be calculated for all particles, and the speedup is in general not so distinct as hinted by eq II.1.3.

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)

$$E_{kin} = \frac{1}{h^2}\sum_{i=1}^{N}\mathbf{v}_i^2 \qquad \text{(II.1.4)}$$

has to be calculated in order to scale the velocities[3] by β

$$\beta = \left(\frac{T_{ref}(N-1)}{16\sum_i \mathbf{v}_i^2}\right)^{1/2} \qquad \text{(II.1.5)}$$

II.2. *si*GLRK Method. The system to be considered here is described by a set of Hamilton equations. The energy $H = V + T$ is the sum of potential energy V and kinetic energy T, where $V = V(r_1, r_2, ..., r_d)$ depends only on the positions of the different particles and where the kinetic energy $T = (1/2)\sum_{i=1}^{d}(m_i\dot{r}_i^2)$ depends on particle interactions. Using Newton's law, the equation of motion (eq II.1.1) reads

$$\ddot{\mathbf{r}} = \mathbf{f}(\mathbf{r}) \qquad \text{(II.2.6)}$$

for a unit mass (m_i = 1). $\mathbf{r}(t)$ is the position of the particle at time t, $\ddot{\mathbf{r}}$ its acceleration, and $\mathbf{f}$ the force acting on it.

The equation of motion can be, equivalently, written as a system of the first-order differential equations

$$\dot{\mathbf{r}} = \mathbf{v} \qquad \text{(II.2.7)}$$

$$\dot{\mathbf{v}} = \mathbf{f}(\mathbf{r}) \qquad \text{(II.2.8)}$$

The system of eqs II.2.7 and II.2.8 is Hamiltonian since the force is the gradient of the potential $-V(\mathbf{r})$

$$\mathbf{f} = -\nabla V(\mathbf{r}) \qquad \text{(II.2.9)}$$

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

$$\mathbf{q}' = \mathbf{F}(\mathbf{q}) \qquad \mathbf{q}(0) = \mathbf{q}_0 \qquad \text{(II.2.10)}$$

is of the form

$$\mathbf{Q}_i = \mathbf{q}_n + h\sum_{j=1}^{s}a_{ij}\mathbf{F}(\mathbf{Q}_j) \qquad i = 1, ..., s \qquad \text{(II.2.11)}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\sum_{i=1}^{s}b_i\mathbf{F}(\mathbf{Q}_i) \qquad \text{(II.2.12)}$$

where $\mathbf{q}$ is a d dimensional vector, determined by Butcher's tableau[17]

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

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 II.2.11 implicitly defines the intermediate stages $\mathbf{Q}_i$, which can be interpreted as an approximation to the exact solution $\mathbf{q}(t)$ at the intermediate point $t = t_n + c_ih$.

Since the coefficients of the s-stage GLRK method of order $2s$ satisfy the relations

$$b_ia_{ij} + b_ja_{ji} - b_ib_j = 0, \quad 1 \le i, j \le s \qquad \text{(II.2.13)}$$

the method is symplectic.[11] Butcher's tableau for the two-stage fourth-order GLRK method is

$$\begin{array}{c|cc} c_1 = \frac{3-\sqrt{3}}{6} & a_{11} = \frac{1}{4} & a_{12} = \frac{3-2\sqrt{3}}{12} \\ c_2 = \frac{3+\sqrt{3}}{6} & a_{21} = \frac{3+2\sqrt{3}}{12} & a_{22} = \frac{1}{4} \\ \hline & b_1 = \frac{1}{2} & b_2 = \frac{1}{2} \end{array} \qquad \text{(II.2.14)}$$

The initial values for the intermediate stages in a fixed-point iteration are determined by means of an adequately chosen extrapolation function. Several ways to determine this function are possible. The extrapolation function $z(t)$, imposed by the character of motion of the system considered, is chosen to be of the form

$$z(t) = s_1 + s_2 \sin \omega(t - t_{n-1}) + s_3 \cos \omega(t - t_{n-1}) \qquad \text{(II.2.15)}$$

where s_1, s_2, and s_3 are free parameters and ω is the typical frequency of a particle. The system of particles with the same typical frequency and the same mass will be taken into account.

II.2.1. Algorithm. Starting values for the fixed-point iteration in each computational step can be written as

$$\mathbf{R}_1 = s_1 + s_2 \sin h\omega(1 + c_1) + s_3 \cos h\omega(1 + c_1) \qquad \text{(II.2.16)}$$

$$\mathbf{R}_2 = s_1 + s_2 \sin h\omega(1 + c_2) + s_3 \cos h\omega(1 + c_2) \qquad \text{(II.2.17)}$$

These values are based on the information from the previous step. At each step of the solution procedure the values of $\mathbf{r}_{n-1}$, $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{r}_n$, which are approximate values of the solution at points t_{n-1}, $t_{n-1} + c_1h$, $t_{n-1} + c_2h$, and $t_{n-1} + h$, are known. The free parameters of eq II.2.15 are obtained by means of the least squares procedure. Therefore, the following constants have to be calculated:

$$S = \sin \omega hc_1 + \sin \omega hc_2 + \sin \omega h$$

$$C = 1 + \cos \omega hc_1 + \cos \omega hc_2 + \cos \omega h$$

$$S_2 = \sin^2 \omega hc_1 + \sin^2 \omega hc_2 + \sin^2 \omega h$$

$$C_2 = 1 + \cos^2 \omega hc_1 + \cos^2 \omega hc_2 + \cos^2 \omega h$$

$$S_C = \sin \omega hc_1 \cos \omega hc_1 + \sin \omega hc_2 \cos \omega hc_2 + \sin \omega h \cos \omega h \qquad \text{(II.2.18)}$$

The free parameters s_1, s_2, and s_3 for the *step*th step are the solution of the following system of linear equations:

$$4s_1 + Ss_2 + Cs_3 = Z_1$$

$$Ss_1 + S_2s_2 + S_Cs_3 = Z_2 \qquad \text{(II.2.19)}$$

$$Cs_1 + S_Cs_2 + C_2s_3 = Z_3$$

where the right-hand side quantities are equal to

$$Z_1 = \mathbf{r}_{n-1} + \mathbf{R}_1 + \mathbf{R}_2 + \mathbf{r}_n$$

$$Z_2 = \mathbf{R}_1 \sin \omega hc_1 + \mathbf{R}_2 \sin \omega hc_2 + \mathbf{r}_n \sin \omega h$$

$$Z_3 = \mathbf{r}_{n-1} + \mathbf{R}_1 \cos \omega hc_1 + \mathbf{R}_2 \cos \omega hc_2 + \mathbf{r}_n \cos \omega h \qquad \text{(II.2.20)}$$

The application of the two-stage fourth-order GLRK method to MD integration problem (eqs II.2.7 and II.2.8) leads to the following relations which implicitly define the intermediate stages (point values) $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{R}_1$, and $\mathbf{R}_2$:

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

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

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

$$\mathbf{R}_2 = \mathbf{r}_n + h(a_{21}\mathbf{V}_1 + a_{22}\mathbf{V}_2) \qquad \text{(II.2.21)}$$

where $\mathbf{V}_i$ and $\mathbf{R}_i$ correspond to $\mathbf{Q}_i$ from eq II.2.11 and $\mathbf{f}$ to the force in eq II.2.9.

The new step velocities and the new step positions are given finally in accordance to eq II.2.12:

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

$$\mathbf{r}_{n+1} = \mathbf{r}_n + (h/2)(\mathbf{V}_1 + \mathbf{V}_2) \qquad \text{(II.2.22)}$$

The procedure is repeated for the desired number of steps.

II.3. Parallelized *si*GLRK Method. The intuitive concept of parallelization leads to a balanced partition of particles among interconnected processors. Moreover, assuming that the computation time is much greater than the communication time, the parallel time complexity can be reduced with the number of processors p.

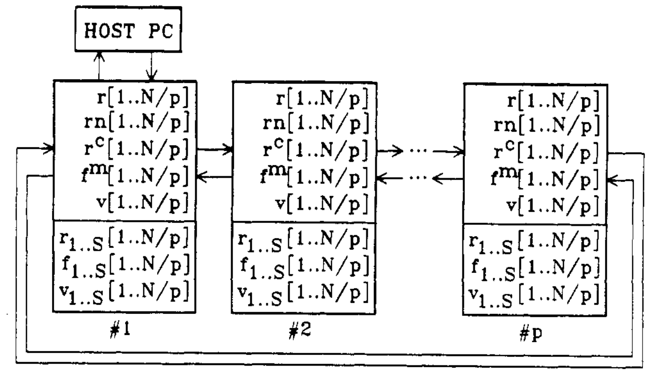

Figure 1. Ring topology and appropriate data buffers for *si*GLRK.

Suppose that the p processors are connected in a ring topology as shown in Figure 1. Each processor has the following common data buffers for N/p local particles: $r[N/p]$ (step position); $rn[N/p]$ (new step position); $r^c[N/p]$ (copy of the step position); $f^m[N/p]$ (mirror force); $v[N/p]$ (step velocity). Additionally, s sets of point buffers have to be allocated ($j = 1, ..., s$): $r_j[N/p]$ (point position); $v_j[N/p]$ (point velocity); $f_j[N/p]$ (point force).

The main part of the *si*GLRK algorithm becomes local due to the ring philosophy. In the main program (see sketched code in Figure 2) N/p particles are assigned to each processor and all data buffers are set to initial values. The constants defined by eqs II.2.18 are then calculated. The main loop is repeated for the requested number of steps, *N_steps*. First, the predictor is calculated according to eqs II.2.16 and II.2.17 in each step. Then i iterations of the corrector (eq II.2.21) are performed in s intermediate points. Finally, new step velocities and positions are calculated by means of eqs II.2.22. The rest of the code operates on the computation of global kinetic energy and eventual scaling, employing eqs II.1.4 and II.1.5.

The setting of initial positions and velocities and the initialization of data buffers are performed in parallel once for all steps.

In each algorithm step a new predictor for every $N_{pr} = N/p$ local particles has to be provided.

Subsequently, *si* calculations of intermediate point forces are carried out. The calculation of forces is the most demanding task, because $N(N-1)/2$ interactions have to be considered. The details of the parallel calculation of forces can be found in ref 14. Hence, only a brief description is presented here.

First, a local copy of the *point*th point positions $\mathbf{r}^c_{point}$ is accomplished (the subscript *point* is omitted in the following text since the procedure is the same for all points). The further calculation proceeds in $\lfloor p/2 \rfloor + 1$ passes. In the first pass $1/2(N/p)(N/p - 1)$ local interactions are performed due to self-interactions $f_{ii} = 0$ and symmetry of the pairwise forces $f_{ij} = -f_{ji}$. The copy of point particle positions $\mathbf{r}^c$ is then transferred to the clockwise neighbor in the ring. The obtained position buffers will be designated with $\mathbf{r}^{c,pass}$ (the superscript *pass* will be omitted since the procedure is the same for all copies). In the next $\lfloor p/2 \rfloor$ passes the full calculation of $(N/p)^2$ partial forces among particles in $\mathbf{r}$ and in $\mathbf{r}^c$ is performed.

The partial pass forces are also summed into the so-called mirror force buffer $\mathbf{f}^m$, which is transferred at the end of each pass to the counterclockwise neighbors on the distance (*pass* − 1). Namely, the calculation in the kth pass is symmetrical to the calculation in the $(p - k + 2)$th pass. The rotated

```
par_GLRK()
{
 charac_init( N/p,p,p_th,Tscale);     /* set the initial positions and
                                         velocities */
 init_d_buffers (N/p);                /* initialize other data buffers */

 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);              /* calculate forces for intermediate
                                         points, and update buffers */
     calc_vel_pos(N/p);               /* calculate velocities & positions in
                                         intermediate points (corrector) */
  }}
  calc_new_vel_pos(N/p);              /* calculate step velocities &
                                         step positions */
  calc_kin_energy(N/p,step,Tscale);   /* calculate local and global
                                         step kinetic energy  */
}}
```

Figure 2. C-like sketched code of the main program for the parallel *si*GLRK.

mirror force buffer $\mathbf{f}^m$ is added to the local force buffer. For p even, the summation and transfer of mirror forces need not to be performed in the last pass.

The name mirror force originates in the way the partial pass forces are summed into the mirror force buffer. When calculating the pass force, summation proceeds with the varying of the second index

$$f_{index(i)} = \sum_j f(\mathbf{r}_i, \mathbf{r}_j^c)$$

which gives the interaction of all particles in $\mathbf{r}^c$ with a single particle from $\mathbf{r}$. The notation *index*(i) denotes the global index of the particle, described by $\mathbf{r}_i$. The summation for the mirror force j on a processor q varies the first index

$$f_j^{m(q)} = \sum_i -f(\mathbf{r}_i, \mathbf{r}_j^c)$$

which gives the interaction of all particles in $\mathbf{r}$ with a single particle in $\mathbf{r}^c$. The sign of partial forces changes due to the pairwise forces symmetry.

The total number of force calculations at each step is

$$N_f = si[(N/2p)(N/p - 1) + \lfloor p/2 \rfloor (N/p)^2]$$
$$= \frac{siN}{p}\left(\frac{N/p - 1}{2} + \frac{\lfloor p/2 \rfloor N}{p}\right) \qquad \text{(II.3.23)}$$

Moreover, the $si(\lceil p/2 \rceil - 1)$ transfers of position buffers in the clockwise direction are needed, and also $(si/2)\lceil p/2 \rceil(\lceil p/2 \rceil - 1)$ transfers of mirror force buffers in the opposite direction are required. The number of buffer transfers in each step is

$$N_{fc} = si(\lceil p/2 \rceil - 1)\left(1 + \frac{\lceil p/2 \rceil}{2}\right) \qquad \text{(II.3.24)}$$

The $N_{vp} = siN/p$ calculations of corrector (intermediate point velocities and intermediate point positions) are produced in parallel for N/p particles.

Then the calculation of the new velocities and the new positions for N/p local particles $N_{nvp} = N/p$ is made.

At the end the calculation of kinetic energy is performed once per each step with N/p particles, $N_{ke} = N/p$. To obtain the global kinetic energy $N_{ke} = \lfloor p/2 \rfloor$, communication steps are required to perform the *combine* function which performs the global summation.

III. TIME COMPLEXITY

In this section we shall estimate the time requirements of one *si*GLRK calculation step for N particles on p processors with regard to the consideration of the communication time and the processing time. In the processing time floating point operations (FLOPS) and data/program code manipulation are included.

III.1. Communication Time. The communication time consists of the time needed for position and mirror force buffer transfers (eq II.3.24) and for the *combine* of kinetic energies. Each buffer is of the size $3(N/p)$ since it includes data for all three dimensions. Therefore, the accumulative communication time is equal to

$$T_c = T_T\left(\frac{3N}{p}N_{fc} + N_{kc}\right)$$
$$= T_T\left(\frac{3Nsi}{p}(\lceil p/2 \rceil - 1)\left(1 + \frac{\lceil p/2 \rceil}{2}\right) + \lfloor p/2 \rfloor\right) \qquad \text{(III.1.25)}$$

where T_T is the time required to transfer a single precision floating point number (4 bytes).

III.2. Processing Time. The FLOPS (addition, A; comparison, C; subtraction, S; multiplication, M; and division, D) can be measured in the number of processor cycles or directly by the time units for a specific processor hardware.

According to the discussion in the previous section, the initialization time of buffers is negligible. The predictor is set once per step for all N/p local particles for all three dimensions $\mathbf{r}_i = (x_i, y_i, z_i)$. The calculation of forces is performed N_f times in accordance to eq II.3.23. Subsequently, si calculations are performed to determine the intermediate point velocities and positions for N/p particles. Finally, new velocities and new positions as well as kinetic energy for N/p local particles are computed, and the *combine* function for the global kinetic energy is performed.

In Table 1 are given the required number of FLOPS for one particle in all three dimensions for the predictor, the pairwise force and the corrector, new velocity, and new position, as well as kinetic energy. In the last column the actual values for the T800 processor are given ($T_A = T_S = T_C = 0.45$ μs, $T_M = 0.75$ μs, and $T_D = 1.1$ μs). T_A represents the time required for one addition of two floating point numbers and

Table 1. Required Number of FLOPS Operations for One Particle in All Three Dimensions and the Actual CPU time for the T800 Processor

type of calculation	required FLOPS operations	T800 time (μs)
predictor	111M, 9D, 27S, 66A	$T_{pr} = 135$
point force	11M, 1D, 4S, 8A, 8C	$T_f = 18.35$
corrector	36M, 8A,	$T_{vp} = 30.6$
new velocity and new position	6M, 21A, 3C	$T_{nvp} = 15.3$
kinetic energy	3M, 3A	$T_{ke} = 3.6$

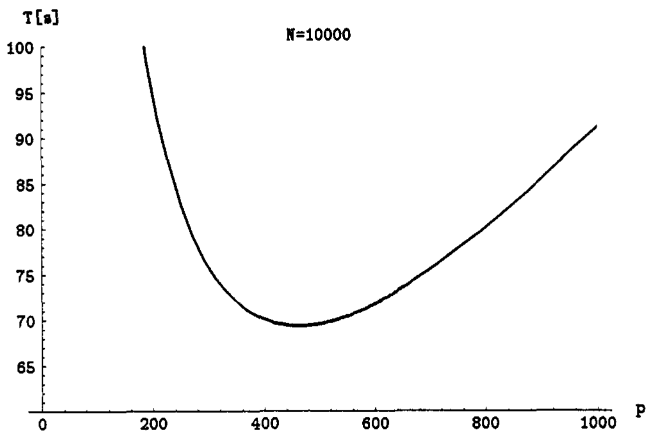

Figure 3. Estimated time requirements for the parallel *24*GLRK ($N = 10^4$, $s = 2$, $i = 4$) implemented on $p \times$ T800 transputers.

similarly for other operations. The time T_F to perform all FLOPS presented in Table 1 for one step is equal to

$$T_F = N_{pr}T_{pr} + N_fT_f + N_{vp}T_{vp} + N_{nvp}N_{nvp} + N_{ke}T_{ke}$$
$$= \frac{siN}{p}\left(\left(\frac{N/p-1}{2} + \frac{\lfloor p/2\rfloor N}{p}\right)T_f + T_{vp}\right) + \frac{N}{p}(T_{pr} + T_{nvp} + T_{ke}) \quad \text{(III.2.26)}$$

The time for data and program code manipulation in a single step is also estimated. The program runs in the main loop N_f times. The rest of the time required for the program execution is negligible. Analyzing the code in a single run, 11 floating point memory reads (T_{MR} denotes a memory read time) and 6 memory writes (T_{MW} denotes a memory write time) are found. It was figured out that approximately 180 instructions are fetched in every inner loop (T_{IE} denotes the average instruction execution time). The total time for data and program code manipulation T_I is then

$$T_I = 11T_{MR} + 6T_{MW} + 180T_{IE}$$

Consequently, the processing time for one step is

$$T_p = T_F + N_fT_I \quad \text{(III.2.27)}$$

III.3. Step Time. The time requirements for the complete program are the sum of the time required for communication and the time required for processing:

$$T^s = T_c + T_p = T_T\left(\frac{3Nsi}{p}(\lceil p/2\rceil - 1)\left(1 + \frac{\lceil p/2\rceil}{2}\right) + \lfloor p/2\rfloor\right) + \frac{siN}{p}\left(\left(\frac{N/p-1}{2} + \frac{\lfloor p/2\rfloor N}{p}\right)(T_f + T_I) + T_{vp}\right) + \frac{N}{p}(T_{pr} + T_{nvp} + T_{ke}) \quad \text{(III.3.28)}$$

The graphical presentations of eq III.3.28 for the *24*GLRK method in the example of a T800 ring are given in Figures 3 and 4.

Figure 3 shows that the step time T^s decreases at first practically inversely proportional to the number of processors

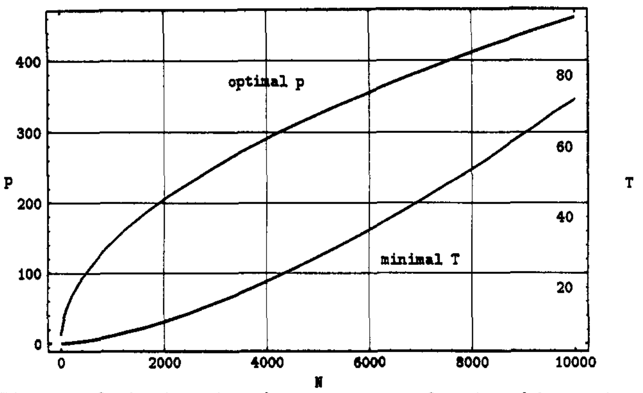

Figure 4. Optimal number of processors p as a function of the number of particles N versus the minimal step time T^s.

Table 2. Classification of Measured Results in CPU Seconds for Various Types of Computers for One 24GLRK Step Time without Cutoff (x() Denotes Informative Expected Values)

target system	$N = 256$	$N = 512$	$N = 1024$
VAX 8650	15.9	62.8	237.7
1 × T800/20 MHz	11.3	44.1	174.1
HP710/50 MHz	5.7	23.0	92.6
PC 486/50 MHz	5.0	x(20)	x(80)
HP730/66 MHz	4.3	17.0	68.1
8 × T800/20 MHz	1.66	6.4	25.0
16 × T800/20 MHz	0.91	3.2	12.1
Convex C3860	0.59	1.98	8.57

p. Then the optimal number of processors is reached. If the number of processors increases above the optimal value, the step time increases also since the communication time tends to dominate. Considering the performances of T800, the step time T^s for one processor in the case of $N = 10^4$ particles would be about 16×10^3 s. The optimal number of processors is approximately $p = 460$, attributing 22 particles to each processor; now the minimal step time is $T^s = 69$ s.

Figure 4 displays the minimal step time T^s for the chosen number of particles N, estimated with regard to the optimal number of processors p.

III.4. Measured Results for the *24*GLRK Method. In the previous paper[15] the *24*GLRK method for MD integration has been shown as a competitive example of implicit RK methods. Our theoretical results are checked by means of an experimental measurement on a 20-MHz T800 transputer ring ($N = 512$, $p = 8$, $T_T = 2.5$ μs, $T_{MR} = T_{MW} = 0.2$ μs, $T_{IE} = 0.1$ μs, and $T_I = 21.4$ μs).

From eq III.3.28 and Table 1 the step time equals

$$T^s = 13828(2.5)\,\mu\text{s} + 512(11442.5 + 30.6)\mu\text{s} + 64(135 + 15.3 + 3.6)\mu\text{s} = 34.6\text{ ms} + 5874.2\text{ ms} + 9.8\text{ ms} = 5918.6\text{ ms} \approx 5.9\text{ s} \quad \text{(III.4.29)}$$

Each step takes about 5.9 s and is composed of the communication time, the time for the force calculation, and the time for the rest of the transactions. However, most of the time is spent on the force calculation.

Table 2 gives the measured results in CPU seconds for the same MD program executed on different types of computers. The computers are classified according to the length of the step time. The number of particles taken into account was $N = 256$, $N = 512$, and $N = 1204$. The maximal time optimization in the compilation process was always applied. In some cases x() designates no available reuslts at the time of the writing of this paper; however, informative expected values are given in parentheses.

The theoretical time complexity given by eq III.4.29 agrees well with the measured values presented in Table 2 for the 8

Table 3. Classification of Measured Results in CPU Seconds for Various Types of Computers for One 24GLRK Step Time with r_{co} = 2.5 (x() Denotes Informative Expected Values)

target system	N = 256	N = 512	N = 1024
1 × T800/20 MHz	7.3	25.6	101.8
VAX 8650	5.0	18.2	51.5
PC 486/50 MHz	3.5	x(14)	x(56)
HP710/50 MHz	1.7	5.5	16.8
HP730/66 MHz	1.2	3.9	11.2
8 × T800/20 MHz	1.1	3.8	14.5
16 × T800/20 MHz	0.59	1.92	7.0
Convex C3860	0.34	1.18	4.07

× T800 system. Table 3 summarizes the results for the same program code as r_{co} = 2.5. The time complexity is in accordance with eq II.1.3 for HP workstations only, owing to their innovative superscalar architecture and cache memory philosophy. Other computers, including the supercomputer Convex 3860 (global memory, six vector processors), produce inferior results since they cannot calculate the distances concurrently with the program code fetching.

IV. CONCLUSIONS

The present work is a first attempt to develop a parallel algorithm for the implicit Runge–Kutta integration scheme, the s-stage Gauss–Lengendre Runge–Kutta (GLRK) method of order $2s$ with i fixed-point iterations for solving the resulting nonlinear system of equations. The algorithm is used for numerical solution of molecular dynamics equations on the distributed memory computers in the ring topology. It is designed and analyzed for the two-stage fourth-order GLRK method for $i = 4$ and applied to a complex system of N particles interacting through the Lennard–Jones potential. The presented parallel algorithm offers inherently a nearly ideal speedup. In practice, where $N \gg p$ and N/p is large enough (e.g., $N = 10^4$, $N/p > 50$ as on Figure 3), the time T^s is practically proportional to $siN^2/2p$ while the time complexity of the sequential algorithm is p times greater.

It has been shown[15] that the step time can be doubled in the *24*GLRK method for the same performances as given by Verlet-type methods. The net enlargement of time complexity compared to the Verlet-type methods is then only $si/2$. It seems reasonable to expect that a further reduction could be achieved with cutoff implementation, especially for intermediate point calculations. The parallel algorithm could compensate for the computational demands of the accurate and stable *si*GRLK methods.

It is important to note that the GLRK methods are symplectic, which means that the symplectic structure in the phase space is preserved and the time reversibility is ensured. In practice, it comes close to conserving the total energy of the system. Much work remains to be done in the practical implementation of the GLRK methods, particularly how to get good initial guesses for the solution of the resulting nonlinear equations, which iterative procedure to use, and how to implement the cutoff criterion, especially in corrector calculations.

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 the Republic 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) Borštnik, B.; Pumpernik, D.; Janežič, D.; Ažman, A. In *Molecular Interactions: Molecular Dynamics Studies of Molecular Interactions*; Ratajczak, H., Orville-Thomas, W. J.; Eds.; John Wiley and Sons: New York, 1980.
(3) Heerman, D. W. *Computer Simulation Methods in Theoretical Physics*; Springer: New York, 1986.
(4) Berendsen, H. J. C.; Gunsteren, W. F. van. In *Molecular Dynamics of Statistical Mechanical Systems: Practical Algorithms for Dynamic Simulation*; Ciccotti, G., Hoover, W. G., Eds.; Proceedings of the International School of Physics, North Holland: Amsterdam, 1986.
(5) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Clarendon Press: Oxford, 1987.
(6) 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.
(7) Peskin, C. S.; Schlick, T. Molecular Dynamics by the Backward-Euler Method. *Commun. Pure Appl. Math.* **1989**, *42*, 1001–1031.
(8) Fincham, D. Leapfrog Rotational Algorithms. *Mol. Simul.* **1992**, *8*, 165–178.
(9) Arnold, V. I. *Mathematical Methods of Classical Mechanics*; Springer: New York, 1978.
(10) Toxvaerd, S. J. Molecular dynamics at constant temperature and pressure. *Phy. Rev. E* **1993**, *47*, 343–350.
(11) Sanz-Serna, J. M. Runge–Kutta Schemes for Hamiltonian Systems. *BIT* **1988**, *28*, 877–883.
(12) Sanz-Serna, J. M. Symplectic Runge–Kutta and related methods: recent results. *Physica D* **1992**, *60*, 293–302.
(13) Fox, G. C.; Johnson, M. A.; Lyzenga, G. A.; Otto, S. W.; Salmon, J. K.; Walker, D. W. In *Solving Problems on Concurrent Processors*; Prentice-Hall: New Jersey, 1988; Vol. 1, Chapter 9, pp 155–165; Chapter 16, pp 291–306.
(14) Trobec, R.; Jerebic, I.; Janežič, D. Parallel Algorithm for Molecular Dynamics Integration. *Parallel Comput.* **1993**, *19*, 1029–1039.
(15) Janežič, D.; Orel, B. Implicit Runge–Kutta Method for Molecular Dynamics Integration. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 252–257.
(16) Jerebic, I.; Slivnik, B.; Trobec, R. In *Library for programming on torus-connected multiprocessors*; Valero, M., Oñate, E., Janes, M., Larriba, J. L., Suárez, B., Eds.; Proceedings of the International Conference PACTA '92, IOS Press: Barcelona, 1992, pp 386–395.
(17) Butcher, J. *The Numerical Analysis of Ordinary Differential Equations, Runge–Kutta Methods and General Linear Methods*; Wiley: Chichester, 1987.