Chapter 10
Parallel Approaches in Molecular Dynamics Simulations

Dušanka Janežič, Urban Borštnik and Matej Praprotnik

Abstract

In this contribution we will present the survey of our past and current endeavor on parallel approaches in molecular modeling algorithm development, for example, molecular dynamics (MD) simulation. In particular, we will describe the new split integration symplectic method for the numerical solution of molecular dynamics equations and methods for the determination of vibrational frequencies and normal modes of large systems, and the distributed diagonal force decomposition method, a parallel method for MD simulation.

Parallel computer programs are used to speed up the calculation of computationally demanding scientific problems such as MD simulations. Parallel MD methods distribute calculations to the processors of a parallel computer but the efficiency of parallel computation decreases due to inter processor communication. Calculating the interactions among all atoms of the simulated system is the most computationally demanding part of an MD simulation. Parallel methods differ in their distribution of these calculations among the processors, while the distribution dictates the method's communication requirements.

We have developed a computer program for molecular dynamics simulation that implements the split integration symplectic method and is designed to run on specialized parallel computers. The molecular dynamics integration is performed by the new integration method, which analytically treats high-frequency vibrational motion and thus enables the use of longer simulation time steps. The low-frequency motion

Dušanka Janežič
National Institute of Chemistry, Hajdrihova 19, 1000 Ljubljana, Slovenia,
e-mail: dusa@cmm.ki.si

Urban Borštnik
National Institute of Chemistry, Hajdrihova 19, 1000 Ljubljana, Slovenia,
e-mail: urban@cmm.ki.si

Matej Praprotnik
National Institute of Chemistry, Hajdrihova 19, 1000 Ljubljana, Slovenia,
e-mail: praprot@cmm.ki.si

R. Trobec et al. (eds.), *Parallel Computing*, DOI 10.1007/978-1-84882-409-6_10,

is treated numerically on specially designed parallel computers, which decreases the computational time of each simulation time step. We study the computational performance of simulation on specialized computers and provide a comparison to standard personal computers. The combination of the new integration method with two specialized parallel computers is an effective way to significantly increase the speed of molecular dynamics simulations.

We have also developed a parallel method for MD simulation, the distributed-diagonal force decomposition method. Compared to other methods its communication requirements are lower and it features dynamic load balancing, which increase the parallel efficiency. We have designed a cluster of personal computers featuring a topology based on the new method. Its lower communication time in comparison to standard topologies enables an even greater parallel efficiency.

10.1 Split Integration Symplectic Method

The standard integrators for solving the classical equations of motion are the second-order symplectic leap-frog Verlet (LFV) algorithm [1] and its variants. Their power lies in their simplicity since the only required information about the studied physical system are its interacting potential and the timescale of the fastest motion in the system, which determines the integration time step size. Therefore they are employed for solving dynamics problems in a variety of scientific fields, for example, molecular dynamics (MD) simulation [2, 3], celestial mechanics [4–6], and accelerator physics [7]. However, in the case of MD integration, the integration time step size is severely limited due to the numerical treatment of the high-frequency molecular vibrations, which represent the fastest motion in the system [8]. Therefore, a huge number of integration steps is usually required to accurately sample the phase space composed of all the coordinates and momenta of all the particles. This is a time-consuming task and is often too demanding for the capabilities of contemporary computers.

One way of overcoming the limitation of the standard methods' integration time step size is to analytically treat high-frequency molecular vibrations. This requires the standard theory of molecular vibrations [9] to be built into the integration method. In this way the fast degrees of freedom are rigorously treated and not removed, as in case of rigid-body dynamics [10–12], where small molecules are treated as rigid bodies. Such semi-analytical second-order symplectic integrators were developed by combining MD integration and the standard theory of molecular vibrations [13–16]. The unique feature of these MD integrators is that the standard theory of molecular vibrations, which is a very efficient tool to analyze the dynamics of the studied system from computed trajectories [17–23], is used not to analyze, but to compute trajectories of molecular systems. Information about the energy distribution of normal modes and the energy transfer between them is obtained without

additional calculations. The analytical description of coupled molecular vibrations can be employed only when using the normal coordinates [9, 13–15] and a translating and rotating internal coordinate system of each molecule [24,25]. The dynamics of an Eckart frame has to be adopted to be used within the second-order generalized leap-frog scheme [26, 27] for MD integration. This assures the time reversibility of the methods [13, 16]. In the following we shortly summarize technical details of the method.

In MD simulations for each atom of the system the Hamilton equations are solved

$$\frac{d\eta}{dt} = \{\eta, H\} = \hat{L}_H \eta \tag{10.1}$$

where $\widehat{L}_H$ is the Lie operator, $\{,\}$ is the Poisson bracket [28], and $\eta = (\mathbf{q}, \mathbf{p})$ is a vector of the coordinates of all the particles and their conjugate momenta.

The formal solution of the Hamiltonian system (10.1) can be written in terms of Lie operators as

$$\eta|_{t_k+\Delta t} = \exp(\Delta t \hat{L}_H)\eta|_{t_k} \tag{10.2}$$

and represents the exact time evolution of a trajectory in phase space composed of coordinates and momenta of all the particles from t_k to $t_k + \Delta t$, where Δt is the integration time step [28].

The first step in the development of a new symplectic integration method is to split the Hamiltonian H of a system into two parts [29, 30]

$$H = H_0 + H_r, \tag{10.3}$$

where H_0 is the part of the Hamiltonian that can be solved analytically and H_r is the remaining part.

Next, a second-order approximation for (10.2), known as the generalized leap-frog scheme [26, 27], is used

$$\eta|_{t_{k+1}} = \exp\left(\frac{\Delta t}{2}\hat{L}_{H_0}\right)\exp(\Delta t \hat{L}_{H_r})\exp\left(\frac{\Delta t}{2}\hat{L}_{H_0}\right)\eta|_{t_k} + O(\Delta t^3), \tag{10.4}$$

which defines the split integration symplectic method (SISM). The whole integration time step combines the analytical evolution of H_0 with a correction from the H_r resolved by numerical integration. The Eq. (10.4) on the operators level describes how to propagate from one point in phase space to another. First, the system is propagated for a half integration time step by H_0, then for a whole step by H_r, and finally for another half step by H_0. The whole integration time step thus combines the analytical evolution of H_0 with a correction arising from the H_r performed by numerical integration. This integration scheme was used as the basis for the development of the SISM.

The model Hamiltonian has the following form

$$H = \sum_i \frac{\mathbf{p}_i^2}{2m_i} + \frac{1}{2}\sum_{bonds} k_b(b-b_0)^2 + \frac{1}{2}\sum_{angles} k_\theta(\theta-\theta_0)^2 + \frac{1}{2}\sum_{torsions} V_0(\cos\phi - \cos\phi_0)^2 + \sum_{i>j}\frac{e_i e_j}{4\pi\varepsilon_0 r_{ij}} + \sum_{i>j} 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right], \quad (10.5)$$

where i and j run over all atoms, m_i is the mass of the i-th atom, $\mathbf{p}_i$ is the linear momentum of the i-th atom, b_0 and θ_0 are reference values for bond lengths and angles, respectively, k_b and k_θ are corresponding force constants, ϕ_0 are the reference values for the torsion angles, and V_0 are the corresponding barrier heights; e_i denotes the charge on the i-th atom, ε_0 is the dielectric constant in vacuum, r_{ij} is the distance between the i-th and j-th atoms, and ε_{ij} and σ_{ij} are the corresponding constants of the Lennard–Jones potential.

The Hamiltonian (10.5) is a typical MD Hamiltonian that describes a system of molecules with only one equilibrium configuration and no internal rotation. We assume that the height of the barrier of the torsional potential is large enough that the motion of atoms in the vicinity of the minimum of the torsional potential can be treated as a harmonic vibration around the equilibrium configuration. The vibrational potential energy is therefore the sum of vibrational potential energies of all the molecules in the system

$$V_{vib} = \sum_{j'=1}^{m} V_{vib_{j'}} = \frac{1}{2}\sum_{bonds} k_b(b-b_0)^2 + \frac{1}{2}\sum_{angles} k_\theta(\theta-\theta_0)^2 + \frac{1}{2}\sum_{torsions} V_0(\cos\phi - \cos\phi_0)^2, \quad (10.6)$$

where $V_{vib_{j'}}$ is the vibrational potential energy of the j'-th molecule.

The pure harmonic Hamiltonian H_0 in the splitting (10.3) is defined as the sum of vibrational energies of all the molecules in the system

$$H_0 = T + V_{harm} = \sum_{j'=1}^{m} (T_{j'} + V_{harm_{j'}}), \quad (10.7)$$

where $T = \sum_i \mathbf{p}_i^2/2m_i$ is the kinetic energy of all the atoms in the systems, $T_{j'}$ is the kinetic energy of the j'-th molecule, V_{harm} is the harmonic vibrational potential energy, which is for an individual molecule defined by Eq. (10.11), $V_{harm_{j'}}$ is the corresponding harmonic vibrational potential energy of the j'-th molecule, and m is the number of all the molecules in the system.

The remaining part of the Hamiltonian

$$H_r = H - H_0 = V_{nb} + V_{ah} \quad (10.8)$$

is then equal to the sum of the nonbonded potential energy

$$V_{nb} = \sum_{i>j} \frac{e_i e_j}{4\pi\varepsilon_0 r_{ij}} + \sum_{i>j} 4\varepsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] \tag{10.9}$$

and the anharmonic vibrational potential energy of higher terms (cubic, quartic, etc.) in terms of displacements of atoms from their equilibrium positions

$$V_{ah} = V_{vib} - V_{harm}. \tag{10.10}$$

The underlying principle to enable the SISM to permit longer integration time steps lies in the analytical treatment of high-frequency vibrations described by H_0. The propagation scheme (10.4) enables to treat the time evolution of the vibrational, rotational, and translational degrees of freedom of each molecule (described by $\exp\left((\Delta t/2)\hat{L}_{H_0}\right)$) independently of all other molecules in the system because the total intermolecular interactions are described by a separate term $\exp\left(\Delta t \hat{L}_{H_r}\right)$. Each molecule is treated as an isolated molecule when propagating by $\exp\left((\Delta t/2)\hat{L}_{H_0}\right)$. Propagation by $\exp\left((\Delta t/2)\hat{L}_{H_0}\right)$ can therefore be solved analytically using normal-mode analysis. In the latter, only quadratic terms are kept in the expansion of the vibrational potential energy V_{vib} and all higher terms are neglected [9]

$$\begin{aligned} V_{vib} \approx V_{harm} = \frac{1}{2} \sum_{i,j=1}^{3N} \left(\frac{\partial^2 V_{vib}}{\partial \Delta q_i \partial \Delta q_j} \right)_0 \Delta q_i \Delta q_j = \\ \frac{1}{2} \sum_{i,j=1}^{3N} \left(\frac{\partial^2 V_{harm}}{\partial \Delta q_i \partial \Delta q_j} \right)_0 \Delta q_i \Delta q_j \\ = \frac{1}{2} \sum_{i,j=1}^{3N} H_{ij} \Delta q_i \Delta q_j = \frac{1}{2} \Delta \mathbf{q} \cdot \mathbf{H} \cdot \Delta \mathbf{q}. \end{aligned} \tag{10.11}$$

Here $\Delta\mathbf{q} = (\Delta x_1, \Delta y_1, \Delta z_1, \ldots, \Delta x_N, \Delta y_N, \Delta z_N)$ is a vector of the relative Cartesian displacement coordinates and their corresponding momenta are $\Delta\mathbf{p} = (m_1 \Delta v_{1_x}, m_1 \Delta v_{1_y}, m_1 \Delta v_{1_z}, \ldots, m_N \Delta v_{N_x}, m_N \Delta v_{N_y}, m_N \Delta v_{N_z})$, where subscripts $_x$, $_y$, $_x$ denote x, y, z components of the internal coordinate system, respectively (see Fig. 10.1).

The Hessian $\mathbf{H} \in \mathbb{R}^{3N \times 3N}$ is a symmetric matrix of the second derivatives of the vibrational potential energy with the elements

$$H_{ij} = H_{ji} = \left(\frac{\partial^2 V_{vib}}{\partial \Delta q_i \partial \Delta q_j} \right)_0 = \left(\frac{\partial^2 V_{harm}}{\partial \Delta q_i \partial \Delta q_j} \right)_0. \tag{10.12}$$

To determine the vibrational motions of the system, the eigenvalues and eigenvectors of the mass-weighted Hessian $\mathbf{M}^{-1/2} \cdot \mathbf{H} \cdot \mathbf{M}^{-1/2}$ have to be calculated [9, 20–22]. This leads to solving a secular equation

$$\det(\mathbf{M}^{-1/2} \cdot \mathbf{H} \cdot \mathbf{M}^{-1/2} - \lambda \mathbf{I}) = 0, \tag{10.13}$$

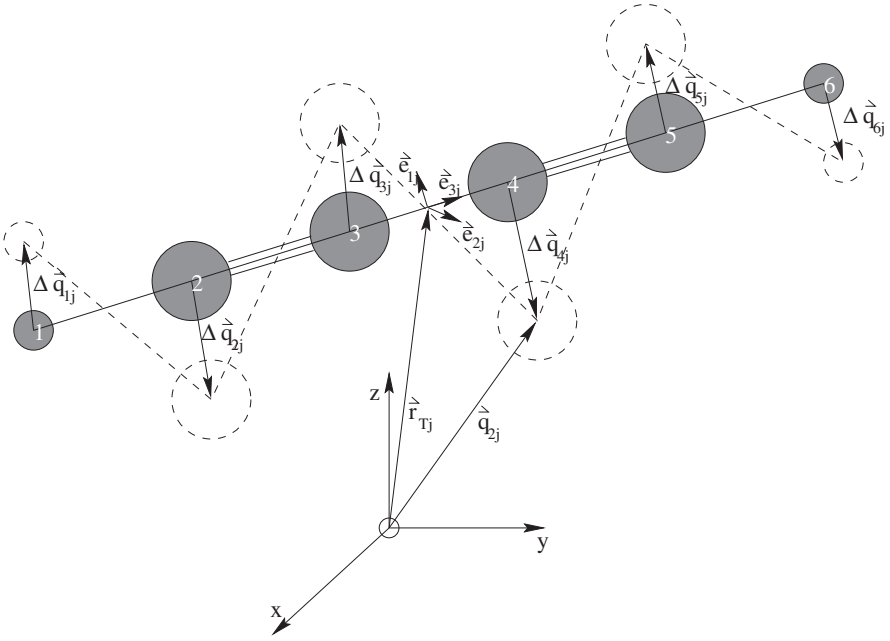


Fig. 10.1 Atom displacement in the Cartesian and the internal coordinate system.

where $\mathbf{M} \in \mathbb{R}^{3N\times 3N}$ is a diagonal mass matrix. The diagonal elements are $M_{11} = m_1$, $M_{22} = m_1$, $M_{33} = m_1$,...,$M_{3N-2,3N-2} = m_N$, $M_{3N-1,3N-1} = m_N$ and $M_{3N,3N} = m_N$. For a nonlinear molecule composed of N atoms, Eq. (10.13) has $3N-6$ nonzero eigenvalues $\omega_i = \sqrt{\lambda_i}$ describing molecular vibrations. The corresponding dynamics is described in the standard theory of molecular vibration by normal coordinates Q_i, $i = 1,2,\ldots,3N-6$ [28]. Six of $3N$ roots in Eq. (10.13) are zero. They correspond to three translations and three rotations of a molecule as a whole while their dynamics is not described in terms of the normal coordinates [9, 13].

An alternative approach to standard theory's description of molecules' rotation and translation [9] is to describe rotation and translation of a molecule in terms of the normal coordinates. To do so the whole atom velocity needs to be expressed in terms of the relative Cartesian displacement coordinates. It has been shown in full detail that the dynamics of the internal coordinate system in this case differs from the dynamics of the Eckart frame, which is employed in the standard theory of molecular vibrations [13].

The equations of motion for the normal coordinates take the Hamiltonian form as [13]

$$\frac{d}{dt}P_i = -\omega_i^2 Q_i; \qquad \frac{d}{dt}Q_i = P_i, \qquad i = 1,2,\ldots,3N \tag{10.14}$$

where P_i is the conjugate momentum to the normal coordinate Q_i [28].

The particular solution of the system (10.14) can be written as [13]

$$\begin{bmatrix} P_i(\frac{\Delta t}{2}) \\ Q_i(\frac{\Delta t}{2}) \end{bmatrix} = \begin{bmatrix} \cos(\omega_i \frac{\Delta t}{2}) & -\omega_i \sin(\omega_i \frac{\Delta t}{2}) \\ \frac{1}{\omega_i} \sin(\omega_i \frac{\Delta t}{2}) & \cos(\omega_i \frac{\Delta t}{2}) \end{bmatrix} \begin{bmatrix} P_i(0) \\ Q_i(0) \end{bmatrix}. \tag{10.15}$$

Equation (10.15) describes vibrational motion corresponding to the normal mode i with $\omega_i > 0$.

The equations of motion for the translation and rotation of a molecule in terms of the normal coordinates, obtained from Eq. (10.15) for the normal coordinates with $\omega_i = 0$ and using $\lim_{x\to 0} \frac{\sin x}{x} = 1$, are [13]

$$P_i\left(\frac{\Delta t}{2}\right) = P_i(0), \tag{10.16}$$

$$Q_i\left(\frac{\Delta t}{2}\right) = P_i(0)\frac{\Delta t}{2} + Q_i(0). \tag{10.17}$$

The expressions for the transformations between Cartesian, relative Cartesian displacement, and normal coordinates are obtained in a straightforward way [13].

The SISM then explicitly reads as follows:

- **Preparatory step**: at the outset of calculation, vibrational frequencies and normal modes of H_0, represented by the normal coordinates P, Q, are determined. The initial normal coordinates P_i^0, Q_i^0, $i = 1,...,3N$, are obtained from the initial atoms' velocities and the initial displacements of the atoms from their equilibrium positions by means of the transformational matrix $\mathbf{A}$. The columns of $\mathbf{A}$ are the eigenvectors of the root-mass-weighted second-derivative matrix $\mathbf{M}^{-1/2} \cdot \mathbf{H} \cdot \mathbf{M}^{-1/2}$ and N is the number of atoms in each molecule.
- **Analytical solution** $\exp\left(\frac{\Delta t}{2}\hat{L}_{H_0}\right)$: the normal coordinates, P_i^0, Q_i^0, are rotated in phase space by the corresponding vibrational frequency ω_i for $\frac{\Delta t}{2}$:

$$\begin{bmatrix} P_i' \\ Q_i' \end{bmatrix} = \mathbf{R} \begin{bmatrix} P_i^0 \\ Q_i^0 \end{bmatrix} \tag{10.18}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\omega_i \frac{\Delta t}{2}) & -\omega_i \sin(\omega_i \frac{\Delta t}{2}) \\ (1/\omega_i) \sin(\omega_i \frac{\Delta t}{2}) & \cos(\omega_i \frac{\Delta t}{2}) \end{bmatrix} \tag{10.19}$$

$\omega_i \neq 0$ defines the vibrations of atoms in each molecule
$\omega_i = 0$ defines translations and rotations of molecules
The normal coordinates of the normal modes with frequency zero
($\lim_{x\to 0} \frac{\sin x}{x} = 1$ for $\omega_i = 0$) evolve as

$$P_i^{'} = P_i^0 \tag{10.20}$$

$$Q_i^{'} = P_i^0 \frac{\Delta t}{2} + Q_i^0 \tag{10.21}$$

Coordinate transformation: the normal coordinates P_k', Q_k' are transformed to the Cartesian displacement coordinates $\Delta p_i'$, $\Delta q_i'$ ($m_1 = m_2 = m_3$,...,$m_{3N-2} = m_{3N-1} = m_{3N}$, where m_i, $i = 1,...,3N$ are the atoms' masses):

$$\Delta p_i^{'} = \sqrt{m_i} \sum_k A_{ik} P_k^{'} \tag{10.22}$$

$$\Delta q_i^{'} = \frac{1}{\sqrt{m_i}} \sum_k A_{ik} Q_k^{'} \tag{10.23}$$

- **Numerical solution**, $\exp(\Delta t \hat{L}_{H_r})$: momenta in the Cartesian coordinates are numerically integrated:

$$p_i'' = p_i' - \Delta t \left(\frac{\partial H_r}{\partial q} \right) \tag{10.24}$$

$$q_i'' = q_i' + \Delta t \left(\frac{\partial H_r}{\partial p} \right) = q_i' \tag{10.25}$$

  Only one force calculation per integration step must be performed. Since $H_r = H_r(q)$ and $\left(\frac{\partial H_r}{\partial p} \right) = 0$, only momenta change at this step.

  Back-transformation: the Cartesian displacement coordinates $\Delta p_k''$, $\Delta q_k''$ are back-transformed to the normal coordinates P_i'', Q_i'':

$$P_i'' = \sum_k \frac{1}{\sqrt{m_k}} A_{ik}^T \Delta p_k'' \tag{10.26}$$

$$Q_i'' = \sum_k \sqrt{m_k} A_{ik}^T \Delta q_k'' \tag{10.27}$$

- **Analytical solution**, $\exp\left(\frac{\Delta t}{2} \hat{L}_{H_0}\right)$: the normal coordinates are again rotated in phase space for $\frac{\Delta t}{2}$:

$$\begin{bmatrix} P_i \\ Q_i \end{bmatrix} = \mathbf{R} \begin{bmatrix} P_i'' \\ Q_i'' \end{bmatrix} \tag{10.28}$$

This concludes one full SISM integration step, which is repeated until the desired number of integration steps is reached.

One time step of SISM is schematically presented in Fig. 10.2.

10.1.1 Calculation of Infrared Spectra

The vibrational and rotational motions of molecules are those which involve energies that produce the spectra in the infrared region. Therefore, the SISM is particularly suitable for computing the IR spectra because rotational, translational, and vibrational motions are resolved analytically, independently of the MD integration time step.

Figure 10.3(a) demonstrates that the IR spectra of bulk water at ambient conditions calculated by SISM and LFV using a 0.5 fs integration time step are in good agreement. These IR spectra were taken as a reference for comparison with calculated IR spectra using longer integration time steps. When using a 1.0 fs integration

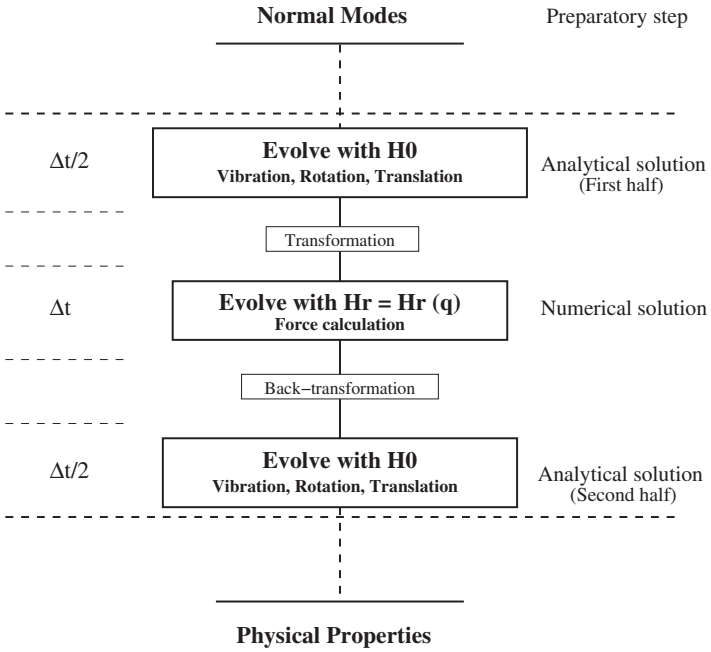
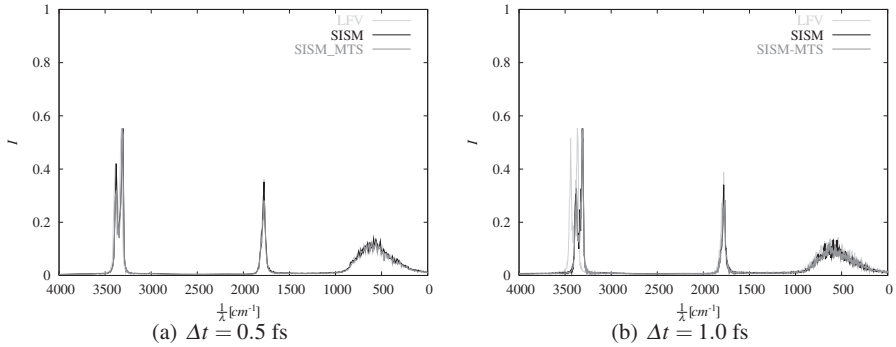


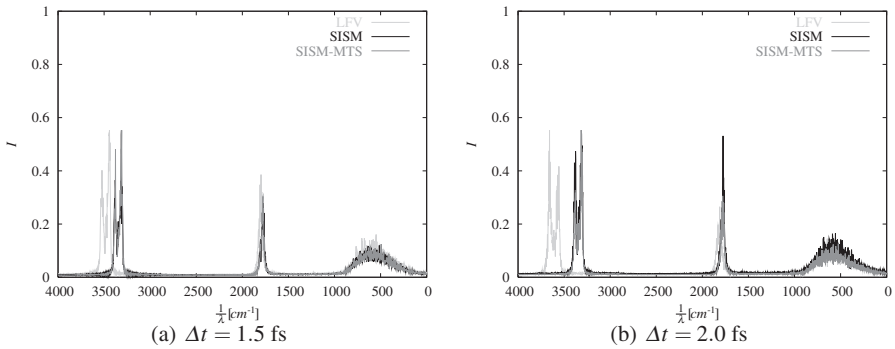
Fig. 10.2 Solution scheme for SISM.



(a) $\Delta t = 0.5$ fs (b) $\Delta t = 1.0$ fs

Fig. 10.3 Calculated (LFV, SISM) IR spectrum of bulk water for $\Delta t = 0.5$ fs and $\Delta t = 1.0$ fs.

time step, the high-frequency double peak at 3300 cm^{-1} in the IR spectrum calculated by the LFV already shifts to the higher frequencies as shown in Fig. 10.3(b). The observed blue shift suggests that when using a 1.0 fs integration time step, the LFV can no longer accurately describe the high-frequency vibrational motions of atoms in a water molecule. This phenomenon is even more evident in Fig. 10.4 for the cases of 1.5 fs and 2.0 fs integration time steps, where the peak at 1775 cm^{-1} also starts shifting toward higher frequencies. Peaks in corresponding IR spectra,



(a) $\Delta t = 1.5$ fs (b) $\Delta t = 2.0$ fs

Fig. 10.4 Calculated (LFV, SISM) IR spectrum of bulk water for $\Delta t = 1.5$ fs and $\Delta t = 2.0$ fs.

which are calculated by the SISM, however remain at the same positions as corresponding peaks in the reference IR spectra calculated using the integration time step of 0.5 fs. This proves that owing to the analytical description of high-frequency molecular vibrations, the latter are accurately described by the SISM also using a 2.0 fs integration time step [15].

10.1.2 Enlarging the Integrational Time Step

The actual speedup of an integrational method is determined by measuring the required CPU time per integration step. Our results show that the computational cost per integration step is slightly larger for the SISM than the LFV for systems smaller than 1000 atoms. However, for larger systems consisting of more than 1000 atoms the computational cost per integration step becomes approximately the same for all of the methods due to the time-consuming $O(N^2)$ numerical calculation of nonbonded forces, which is performed by all three methods in the same way and prevails over the additional calculations in the iterative SISM, which scale linearly with N. Therefore, the speedup of the SISM over the LFV is determined mainly by the significant difference in the integration time step size owing to the analytical treatment of high-frequency motions by SISM [16, 31].

10.2 Parallel Computers

Computers are an essential tool used to solve computational problems in science today. The speed of computer processors is continually increasing, enabling its use to approach ever more complex computational problems [32, 33]. However, many existing problems would be well served by an increase in computational capacity

today. For these problems, parallel computers provide a solution [34]. Many scientific problems can be effectively parallelized to run on a parallel computer.

10.2.1 Parallel Computing

In parallel computing, a problem is split into several subproblems that are solved concurrently on parallel processors in a shorter time. A parallel program is written to be executed on many processors at once and they must correctly share and exchange data to solve the problem. Generally, the processors must communicate throughout the computation since the results from one processor are needed by others. The manner in which the initial problem is divided among the processors – the data distribution and the distribution of computation – greatly affects how the parallel program is written and the time that is spent for communication. Generally, time that is spent for communication cannot be used for communication, since the processor is waiting for input for its next calculation.

10.2.1.1 Parallel Efficiency

Since it is the goal of parallel computing to reduce the total time required to solve a problem, the time spent for communication must be minimized. If it takes time T to solve the problem, an ideal parallel computation on P processors would take only T/p; however, due to the time lost to communication and other factors, the time T_p required by any processor is usually greater: $T_p > T/p$. We can now define the *speedup*

$$S = \frac{T}{T_p} \tag{10.29}$$

as the factor specifying how much faster the parallel computation is compared to a single processor computation. Ideally, the speedup S would equal the number of processors P, $S = P$, which is true if $T_p = T/P$. In several rare cases such a *linear* speedup is possible or even exceeded due to hardware effects [35]. We can define the *parallel efficiency*

$$E = \frac{S}{P} = \frac{T}{PT_p} \tag{10.30}$$

to measure the performance of the parallel computation relative to the ideal time. In optimizing a parallel program, we strive to obtain the highest parallel efficiency since it directly translates to increasing the speedup offered by the program. A higher parallel efficiency is obtained by bounding the communication time and by ensuring that all of the processors have an equal computational load. If processors have unequal computational loads, then whenever the processors communicate globally, the ones with the lowest load must wait for the most loaded processor to finish its computation and begin communication. Load balancing the computation

attempts to keep an equal computional load among all processors, which minimizes waiting time and achieves a higher parallel efficiency.

10.2.2 Parallel Computer Types

Parallel computers may be divided into two broad categories depending on the way processors access memory. The type of memory access greatly influences the way in which a parallel program must be written.

Shared Memory

In shared memory computers, all processors may access all memory directly (i.e., a processor may read from or write to any memory location, as if the memory were local). The two common types of shared memory computers are symmetric multiprocessing (SMP) computers, in which all memory is local to all of the processors. In effect, all memory accesses require the same access time. In nonuniform memory access computers (NUMA), processors have local memory, which provides the fastest access times; however, they can still directly access remote memory (i.e., another processor's local memory), albeit with a higher access time.

Distributed Memory

In distributed memory computers, processors can access only their local memory, but they cannot directly access remote memory. All data exchange between the processors must occur by explicit *message passing* that involves both processors exchanging messages over a processor interconnect, which provides the connection among the processors. Current interconnect technologies range from standard Ethernet to higher-performance Myrinet [36], Infiniband [37], and others.

Libraries, such as the Parallel Virtual Machine [38] (PVM) or the Message Passing Interface [39,40] (MPI) are used to abstract the implementation details of a given computer's message-passing hardware, providing a standard interface to the programmer. Since distributed memory computers are more specific than shared memory computers, parallel programs targeted for distributed memory computers can run on shared memory ones as well. Specific implementations of message-passing libraries on shared memory computers are often optimized to take advantage of the shared memory.

Modern parallel computers, such as clusters of personal computers, are increasingly hybrids of both shared memory and distributed memory computers: the parallel computer is composed of a number of shared-memory nodes (such as multiprocessor, multi-core personal computers), which are in turn connected by the interconnect. While the processors in one node share memory, the overall parallel

computer is still characterized by its distributed memory. The programmer must still use a message-passing library as the overall data exchange mechanism.

10.2.2.1 Topologies of Clusters

Clusters are traditionally built using switching technologies. Indeed, the first clusters used the fastest Ethernet switches then available [41, 42]. However, switches often have limited number of connections, limiting the cluster size, and often have a limited amount of bandwidth that must be shared among all nodes connected to it, which is especially true if multiple levels of switches are used [43].

Many parallel computers have therefore been designed around point-to-point connections between individual processors. A point-to-point processor interconnect can be described by a mathematical graph. The vertices of a graph correspond to the processors while the edges correspond to the interconnect's connections between the processors. The topology of the interconnect is then described by the graph's topology. While it is virtually impossible to provide full direct connectivity among any processor pair for larger numbers of processors, the topology can be chosen to have desirable attributes from both a performance standpoint as well as from an ease of programming perspective. Generally, successful topologies used for MD simulation have been rings, meshes [44], and hypercubes [45].

10.2.3 Reducing Computational Complexity in Molecular Dynamics Simulations

The number of nonbonding interactions in a molecular system greatly outnumbers the number of bonding interactions. A system of N atoms has $O(N^2)$ nonbonding interactions arising from the $N^2/2$ atomic pairs. Since any atom can have at most a few bonds, the number of bonding interactions is $O(N)$. The calculation of the nonbonding interactions is the principal limiting factor in computer simulations, limiting not only the attainable simulation lengths but also the system sizes that can be feasibly simulated.

Several approaches are used to reduce the computational complexity of nonbonding interactions below $O(N^2)$. Among these are employing an interaction cutoff distance, the Barnes–Hut tree method [46], and the fast multipole methods [47, 48].

Cutoff Distance

Employing a cutoff distance is among the principal means of reducing the computational complexity of computing nonbonding interactions [49]. A characteristic of nonbonding interactions is their decreasing magnitude with increasing distance. Both commonly-employed potentials in classical MD simulations behave this way.

The Lennard–Jones potential used to describe van der Waals interactions between atomic pairs, decays as r^{-6} with increasing distance r and the Coulomb potential, which describes the electrostatic interaction between atomic pairs, decays as r^{-1} with increasing distance r. The limit at infinite distance for these interactions is 0. The potential can be changed or redefined to be 0 beyond a certain cutoff distance. Various methods are used to achieve this while retaining an accurate simulation despite the changed functional form [50, 51].

The gain is that only interactions with the cutoff distance need to be calculated. Since interactions among atoms farther apart than the cutoff distance is defined to be zero, their calculation can be ignored. Instead of calculating $O(N)$ interactions for each of the N atoms (yielding $O(N^2)$ interaction calculations), only a finite subset of interactions for each of the N atoms must be calculated. The size of the subset depends on the system density and the cutoff radius, but is independent of the system size. The computational complexity is therefore reduced to $O(N)$.

Tree and Fast Multipole Methods

Tree-based methods and fast multipole-based methods provide a means to account for all the pairwise interactions in a molecular system with a computational complexity less than $O(N^2)$. Both involve clustering spatially close atoms into clusters and using representative values of these clusters instead of individual atoms to calculate distant interactions. In the Barnes and Hut tree method, interactions are calculated individually for each atom. For close by atoms, the interaction is calculated directly. Beyond a certain distance, the interactions are calculated between the atom and the cluster. The computational complexity of the tree-based methods is $O(N \log N)$. In the fast multipole methods, several orders of multipoles are calculated for each atomic cluster. Atomic interactions are derived from the interactions of their representative multipoles. For most distant clusters, individual atoms are not even considered.

As seen later in Sect. 10.3.1, the tree- and multipole-based methods are especially well suited to parallelization techniques in which the presence of atoms on individual processors is limited.

10.3 Parallel Molecular Dynamics Computer Simulations

In parallel calculations of molecular dynamics simulations, processors are used in parallel to calculate the two parts of every MD integration step: the force calculation and the coordinate update [52]. MD simulation time steps are inherently sequential: the newest coordinates are needed to correctly calculate the forces and coordinates can be updated only when the latest forces have been calculated. While the force calculation and the coordinate update are calculated in parallel, the processors must exchange force and atomic coordinates between these two calculations in a global

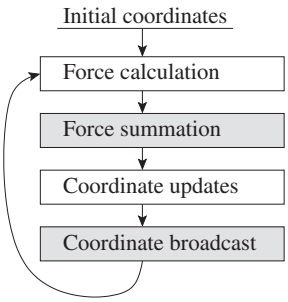


Fig. 10.5 The parallel main loop in molecular dynamics. It consists of two computation phases indicated in white boxes (the force calculation and coordinate updates) and two communication phases indicated in grayed boxes (the force summation and coordinate broadcast). The global operations performed in the communication phases are detailed in Sect. 10.3.3.

operation step. The parallel MD loop is shown in Fig. 10.5 and the global operations are detailed in Sect. 10.3.3.

10.3.1 Methods for Parallel Molecular Dynamics Simulations

Three main classes of parallel methods have been developed for MD simulations: replicated data [53, 54], spatial decomposition [55], and force decomposition [54, 56, 57]. Several advanced methods combine both the spatial and force decomposition approaches [58–61]. The methods differ in how interaction calculations are distributed among the processors. Since a processor needs coordinate data to calculate interactions, the distribution of interaction calculation determines the data distribution among the processors. The data distribution in turn governs the data that must be transferred among processors in each global operation. In addition, the atomic distribution maps atoms to processors for coordinate updates and other calculations that do not depend on interactions with other atoms.

Replicated Data

The replicated data method [53, 54] is the most straightforward parallelization method yet with the highest communication cost. As its name implies, all atomic data are replicated among all processors. As such, each global operation step entails the transfer of all N atomic data among all P processors, which has a higher communication cost than other methods. The global communication can easily be performed using a single collective operation routine. Any processor can calculate any interaction and perform any of the force updates, which simplifies load balancing. The atomic distribution is therefore very fluid.

Spatial Decomposition

In the spatial decomposition method, the space of the molecular system is divided into separate regions, nominally one per processor. The processors are then responsible for calculating the interactions among atoms in their region of space; for this, they need to communicate with at least their 27 neighboring processors resulting in a data transfer volume of $(N/P)^{2/3}$. The spatial decomposition method is well suited to simulations with a short cutoff distance. Since the transferred data volume is limited and the communication due to the global operations is also limited to nearby processors, it is straightforward to map processors onto common interconnect topologies such as a mesh. If no cutoff would be used, the communication would degenerate to data replication. If the molecular system does not have uniform density, the load balancing is nontrivial. The atomic distribution generally mirrors the spatial decomposition, that is, a processor updates coordinates of the atoms in its assigned spatial region.

Force Decomposition

The force decomposition [54,56,57] method divides the N^2 force matrix (representing the N^2 interactions among N atomic pairs) into P disjoint sets called blocks, where P is related to the number of processors employed for the calculation. Such a division of the force matrix implies that the set of N atoms is divided into $N/\sqrt{P}$ subsets. Each processor calculates the interactions in its region, that is, among the atoms in two blocks. Only $O(N/\sqrt{P})$ data is exchanged and a processor communicates only with $\sqrt{P}$ other processors that are in the same processor row or column. The atomic distribution is a refinement of the distribution of atoms into blocks. Atoms in a block are assigned to one of the $\sqrt{P}$ processors associated with the block for coordinate updates since its data are already present on the processor.

10.3.2 Specialized Processors

Specialized processors are processors that are designed for only a certain type of calculation. While they are much faster than general-purpose processors, they are more difficult to use. They are usually coprocessors, located in the host computer, and software must be specially written to effectively use them. A common example are the graphics processing units (GPU) found in modern personal computers. These processors optimized for calculating the linear algebra operations that are commonly used for computer graphics but are not as suited for other general purpose calculations as general-purpose processors [62, 63].

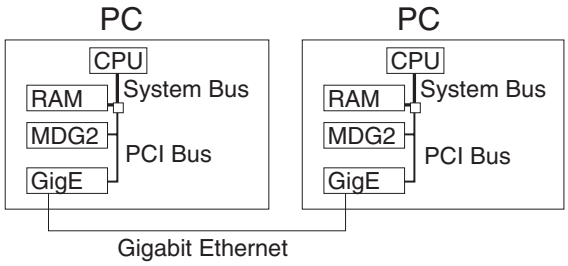


Fig. 10.6 The use of parallel MDGRAPE-II processors. Shown are two personal computers (PCs), each with one MDGRAPE-II processor (labeled MDG2). The PCs are directly connected with a gigabit Ethernet point-to-point connection.

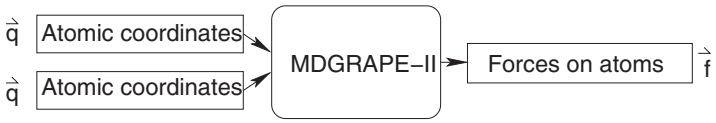


Fig. 10.7 The calculation of forces by the MDGRAPE-II. The atomic position vectors **q** are input, and the MDGRAPE-II returns a vector of forces **f** exerted on the atoms.

MDGRAPE

The MDGRAPE (MD Gravity Pipeline) processor is a specialized processor for calculating MD simulations [64–67]. Specifically, it is used for the fast evaluation of pairwise interactions, which is precisely the most demanding part of MD simulations. Due to its specialization, it can be effectively used to calculate only the nonbonding interactions. Other calculations, including bonding interactions, are calculated on the general-purpose processor of the host computer. An example of two MDGRAPE-II processors placed in two PCs is shown in Fig. 10.6. Using the MDGRAPE-II processor achieves an eightfold speedup in the evaluation of pairwise interactions compared to standard contemporary processors [68].

In MD applications on specialized processors, the input data are the atomic coordinates and atomic types, while the output data are the interactions, for example the forces acting on the atoms or the energies of individual atoms. As an example for the MDGRAPE-II processor, the interaction to be calculated (i.e., the Coulomb and the Lennard–Jones potential) is defined as a function and uploaded to the processor. Coordinates are then sent to the processor in a vector, and the return value is the vector containing forces or the atomic energies. The process of calculating interactions is depicted in Fig. 10.7. The calculation on other specialized processors proceeds in a similar manner.

10.3.3 Global Communication in Parallel Molecular Dynamics Simulations

Global operations entail a communication operation in which all processors participate. A simple example is the broadcast of data by one processor to all others. The collective operations that are present in many message-passing libraries often include basic collective operations such as a broadcast-to-all and all-to-all data exchanges; however, more complex global operations must still be programmed by hand to be efficient [45, 69]. The two main operations found in parallel MD are the global sum and global broadcast [45, 69]. The role of these two global operations is illustrated in Fig. 10.5.

Global Sum

The global sum operation in MD is used after the calculation of interactions, for example, forces acting on atoms. After the calculation, many processors may have a partial force acting upon an atom, their sum being the total force, which is the same as if it were calculated by a single processor. The global sum operation therefore sums all of the partial forces to obtain the total forces. In addition, the force is needed only by the processor that updates coordinates. Therefore, an efficient implementation of the global sum operation leaves the total forces only on the processors performing the coordinate updates of the respective atoms. The global operation can be implemented using the `MPI_reduce_scatter` MPI routine in a parallel MD program using the replicated data parallelization method [69] in which any processor may have a force acting on any atom.

Global Broadcast

The global broadcast is used in MD simulations to broadcast updated coordinates to processors. After processors perform force updates for their respective atoms, other processors must receive the updated coordinates to correctly calculate the next interactions. The global broadcast operation performs this broadcast. In a replicated data parallel MD program, the `MPI_allgatherv` MPI routine may be used since every processor may need coordinates of any atom.

The global sum and broadcast operations for parallel MD not using the replicated data parallelization method tend to be more complex. In spatial decomposition, the global sum needs to sum interactions from neighboring processors only (assuming the cutoff distance is small enough) and the broadcast has a similarly small locality. In the force decomposition method, the communication in the global sum and global broadcast operations is limited to blocks. Only the processors that share a block communicate. Since data within a block is replicated, the processors within a block

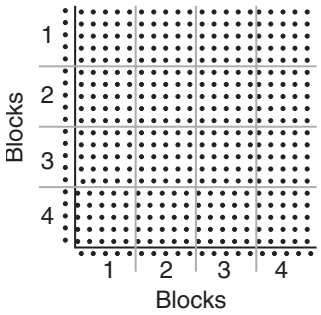


Fig. 10.8 The decomposition of the force matrix used for our parallel SISM MD program. An example for 20 atoms and 16 processors is shown. The atoms are divided into 4 blocks and one processor is assigned to calculate the interactions among each of the $4 \times 4 = 16$ block pairs.

perform a "block-limited" version of the global operation used in replicated data parallel MD.

10.4 Parallelization of SISM

Because the SISM method focuses on speeding up the calculation of bonding interactions and parallelization focuses on speeding up the calculation of non-bonding interactions, it is natural to complement the two approaches.

To showcase the complementarity of the SISM method, parallelization, and the use of specialized processors, we have developed a parallel program for MD simulation implementing the SISM method [68]. It supports the use of multiple MDGRAPE-II processors in many host computers. We opted to use the force decomposition approach to parallelization and do not rely on any special interconnect topology. The method is available for distributed memory parallel computers

The decomposition of the force matrix that we used in our program is depicted in Fig. 10.8. Molecules are never split into different blocks. The molecules in every block are also assigned to individual processors, forming an atomic distribution. A processor applies the SISM to the molecules assigned to it, including coordinate updates of its constituent atoms. The processor is also responsible for calculating interactions among the atoms in two of its associated blocks. If the MDGRAPE-II board is present, the calculations are performed on the board as shown in Fig. 10.9, otherwise the host processor calculates the interactions.

10.4.1 The Distributed Diagonal Force Decomposition Method

To enable calculations of the SISM method on larger, general parallel computers that do not rely on specialized processors, we have implemented the distributed diagonal force decomposition (DDFD) method [70,71]. The DDFD method is an extension of the general force decomposition method. It uses a minimal number of processors for

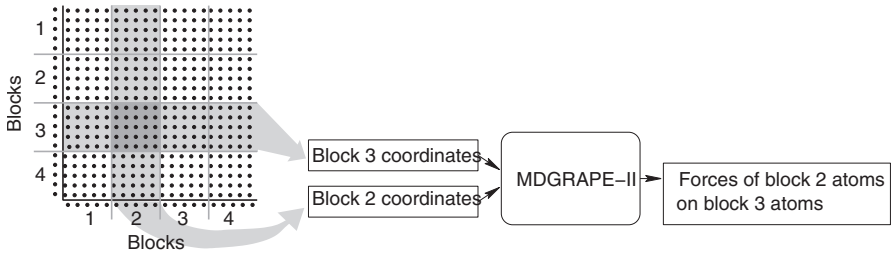


Fig. 10.9 Force calculation on the MDGRAPE-II processor using force decomposition. Shown is the force calculation of the interactions among the atoms in blocks 2 and 3, specifically the forces exerted by block 2 atoms on block 3 atoms. The blocks are highlighted with a light gray background; the dark gray square represents the interactions among the atoms of these two blocks. A separate calculation is used to calculate the equal but opposite forces of block 3 atoms on block 2 atoms.

the number of blocks used to decompose the force matrix. Since a larger number of blocks are smaller, the communication requirements are lower, resulting in a higher parallel efficiency.

In the DDFD method, the diagonal of the force matrix is distributed. As seen in Fig. 10.10(a), there are three types of interactions among the atomic blocks: a block product (interactions among two atomic blocks) lies either above, on, or below the diagonal. The interactions in the block products above the diagonal are opposite but equal to the interactions in the block products below the diagonal, so they do not have to be explicitly calculated. The interactions in block products on the diagonal are only among atoms in the same block. Any processor that has atomic data for these atoms can calculate any of the intra-block interactions for this block. As seen in Fig. 10.10(b), these interactions are distributed for calculation to processors below the diagonal; Fig. 10.10(c) shows the final state. The number of processors needed is equal to only the number of block products below the diagonal.

A side effect of the diagonal distribution process in the DDFD method is the straightforward implementation of load balancing. The distribution of interactions from a diagonal block product to processors holding the block data can easily be altered, assigning specific processors more or less interaction calculations. By altering the diagonal distribution in this way, the computational load of the processors is changed [70, 71]. Load balancing is especially crucial when using an interaction cutoff distance, since the computational load inherently varies among processors. In addition, due to atomic motion during the MD simulation, the atoms included with one atom's cutoff range varies throughout a simulation. Since the load balancing in the DDFD method is dynamic, the load balancing is dynamically tuned during the entire MD simulation, resulting in a higher parallel efficiency.

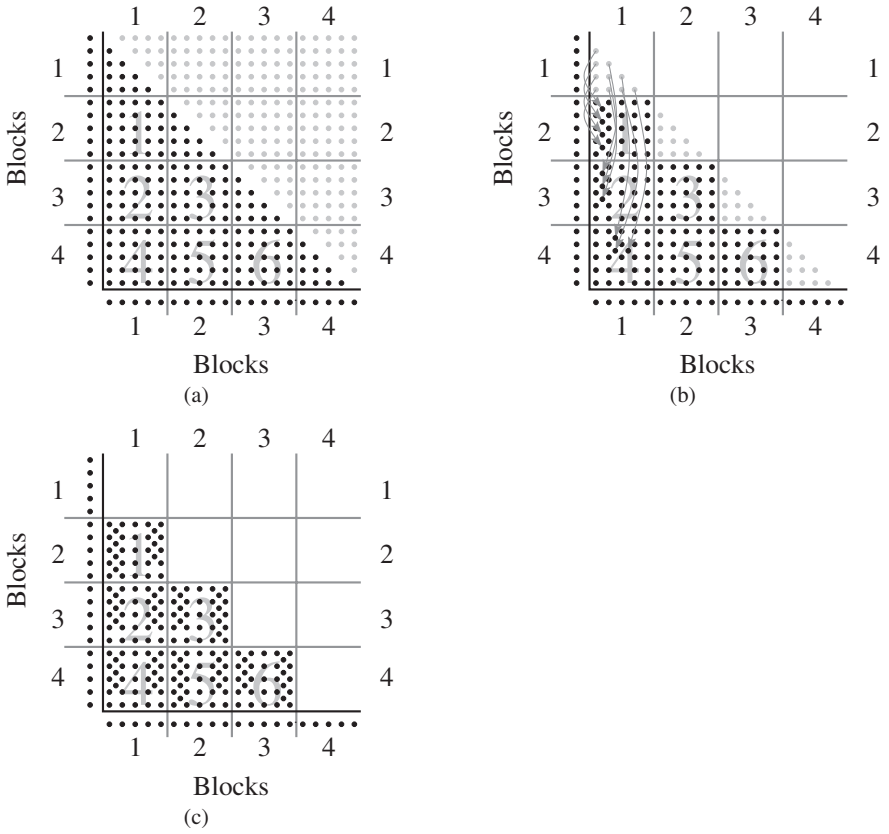


Fig. 10.10 The distributed diagonal force decomposition (DDFD) method. In **(a)** the interactions between the atoms are shown. The self-interactions (diagonal interactions) are 0 and not considered. The grayed interactions above the diagonal are equal but opposite to the ones below the diagonal and are therefore obtained from those. One processor is assigned to every block product of two different blocks. For example, processor 1 is assigned to the product of blocks 1 and 2, while no processor is assigned to the product of block 1 with itself; as shown in **(b)**, these interactions are rather assigned to processors calculating other interactions with block 1 (i.e., processors 1, 2, and 4). The final state when this diagonal distribution is performed for all 4 blocks is shown in **(c)**.

10.5 Conclusions

We have presented our research on parallel approaches to MD simulation. We have shown the complement between algorithmic approaches and parallelization in the quest to speed up the calculation of simulations.

The SISM, based on the standard theory of molecular vibrations, enables the use of much larger integration time steps than are possible with standard MD methods. Since the computational cost of an MD time step remains virtually constant, the computational time of an MD simulation is drastically reduced.

While the SISM allows larger integration time steps to be used, other methods must be used to reduce the computational time of the time steps themselves. Calculating nonbonding interactions dominates an MD time step, so focusing on reducing the time of calculating nonbonding forces is crucial. Specialized hardware can be effectively used to reduce the computational time of nonbonding interactions. We have shown the use of multiple MDGRAPE-II processors to speed up the calculation of nonbonding interactions.

As the algorithmic approaches and specialized hardware reduce the computational cost of individual MD time steps, efficient parallelization becomes even more important to achieving faster MD simulations, since the communication time increases relative to the computational time. The SISM is readily parallelized, including its implementation with multiple specialized processors. Used in combination with the force decomposition method, the communication between the distributed memory computers is guaranteed to be limited even for systems where no distance cutoff is employed. The DDFD method further reduces communication requirements among processors and enables a greater number of processors to be used. In addition, it intrinsically supports dynamic load balancing, which allows effective load balancing, which leads to higher parallel efficiencies and greater speedups of MD simulations.

Acknowledgments The authors would like to acknowledge the financial support of the Slovenian Research Agency under grant No. P1-0002.

References

1. L. Verlet, Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, Phys. Rev. 159 (1967) 98–103.
2. J. M. Sanz-Serna, M. P. Calvo, Numerical Hamiltonian Problems, Chapman & Hall, London (1994).
3. B. J. Leimkuhler, S. Reich, R. D. Skeel, Integration methods for molecular dynamics, IMA (1994) 1–26.
4. J. Wisdom, M. Holman, Symplectic maps for the N-body problem, Astron. J. 102 (1991) 1528–1538.
5. J. Wisdom, M. Holman, J. Touma, Symplectic correctors, Field Inst. Commun. 10 (1996) 217–244.
6. J. Laskar, P. Robutel, High order symplectic integrators for perturbed Hamiltonian systems, Celestial Mech. 80 (2001) 39–62.
7. L. Nadolski, J. Laskar, Application of a new class of symplectic integrators to accelator tracking, Proceedings of EPAC 2002 (2002) 1276–1278.
8. T. Schlick, E. Barth, M. Mandziuk, Biomolecular dynamics at long timesteps: Bridging the timescale gap between simulation and experimentation, Annu. Rev. Biophys. Biomol. Struct. 26 (1997) 181–222.
9. E. B. Wilson, J. C. Decius, P. C. Cross, Molecular Vibrations, McGraw-Hill Book Company, Inc., New York (1955).
10. N. Matubayasi, M. Nakahara, Reversible molecular dynamics for rigid bodies and hybrid Monte Carlo, J. Chem. Phys. 110 (1999) 3291–3301.

11. T. F. Miller III, M. Eleftheriou, P. Pattnaik, A. Ndirango, D. Newns, G. J. Martyna, Symplectic quaternion scheme for biophysical molecular dynamics, J. Chem. Phys. 116 (2002) 8649–8659.
12. M. Ikegutchi, Partial rigid-body dynamics in NPT, NPAT and NPγT ensembles for proteins and membranes, J. Comput. Chem. 25 (2004) 529–541.
13. D. Janežič, M. Praprotnik, F. Merzel, Molecular dynamics integration and molecular vibrational theory: I. New symplectic integrators, J. Chem. Phys. 122 (2005) 174101.
14. M. Praprotnik, D. Janežič, Molecular dynamics integration and molecular vibrational theory: II. Simulation of non-linear molecules, J. Chem. Phys. 122 (2005) 174102.
15. M. Praprotnik, D. Janežič, Molecular dynamics integration and molecular vibrational theory: III. The infrared spectrum of water, J. Chem. Phys. 122 (2005) 174103.
16. M. Praprotnik, D. Janežič, Molecular dynamics integration meets standard theory of molecular vibrations, J. Chem. Inf. Model 45 (2005) 1571–1579.
17. R. Rey, Vibrational energy of HOD in liquid D_2O, J. Chem. Phys. 104 (1996) 2356–2368.
18. R. Rey, Transformation from internal coordinates to Cartesian displacements in the Eckart frame for a triatomic molecule, Chem. Phys. 229 (1998) 217–222.
19. R. Rey, Vibrational phase and energy relaxation of CN^{-1} in water, J. Chem. Phys. 108 (1998) 142–153.
20. B. R. Brooks, D. Janežič, M. Karplus, Harmonic analysis of large systems: I. Methodology, J. Comput. Chem. 16 (12) (1995) 1522–1542.
21. D. Janežič, B. R. Brooks, Harmonic analysis of large systems: II. Comparison of different protein models, J. Comput. Chem. 16 (12) (1995) 1543–1553.
22. D. Janežič, R. M. Venable, B. R. Brooks, Harmonic analysis of large systems: III. Comparison with molecular dynamics, J. Comput. Chem. 16 (12) (1995) 1554–1566.
23. M. Praprotnik, D. Janežič, J. Mavri, Temperature dependence of water vibrational spectrum: a molecular dynamics simulation study, J. Phys. Chem. A 108 (2004) 11056–11062.
24. C. Eckart, Some studies concerning rotating axes and polyatomic molecules, Phys. Rev. 47 (1935) 552–558.
25. J. D. Louck, H. W. Galbraith, Eckart vectors, Eckart frames, and polyatomic molecules, Rev. Mod. Phys. 48 (1) (1976) 69–106.
26. H. F. Trotter, On the product of semi-groups of operators,, Proc. Am. Math. Soc. 10 (1959) 545–551.
27. G. Strang, On the construction and comparison of difference schemes, SIAM J. Numer. Anal. 5 (1968) 506–517.
28. H. Goldstein, Classical Mechanics, 2nd Edition, Addison-Wesley Publishing Company (1980).
29. D. Janežič, F. Merzel, An efficient symplectic integration algorithm for molecular dynamics simulations, J. Chem. Inf. Comput. Sci. 35 (1995) 321–326.
30. D. Janežič, F. Merzel, Split integration symplectic method for molecular dynamics integration, J. Chem. Inf. Comput. Sci. 37 (1997) 1048–1054.
31. D. Janežič, M. Praprotnik, Molecular dynamics integration time step dependence of the split integration symplectic method on system density, J. Chem. Inf. Comput. Sci. 43 (6) (2003) 1922–1927.
32. U. Borštnik, M. Hodošček, D. Janežič, Fast parallel molecular simulations, Croat. Chem. Acta 78 (2) (2005) 211–216.
33. W. F. van Gunsteren, H. J. C. Berendsen, Computer simulation of molecular dynamics: Methodology, applications, and perspectives in chemistry, Angew. Chem. Int. Ed 29 (9) (1990) 992–1023.
34. D. W. Heermann, A. N. Burkitt, Parallel Algorithms in Computational Science, Springer-Verlag, Berlin (1991).
35. R. Trobec, M. Šterk, M. Praprotnik, D. Janežič, Implementation and evaluation of MPI-based parallel MD program, Int. J. Quant. Chem. 84 (1) (2001) 23–31.
36. N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, W.-K. Su, Myrinet: A gigabit-per-second local area network, IEEE Micro 15 (1) (1995) 29–36.

37. J. Liu, J. Wu, D. K. Panda, High performance RDMA-based MPI implementation over InfiniBand, Int. J. Parallel Programm. 32 (3) (2004) 167–198.
38. V. S. Sunderam, PVM: A framework for parallel distributed computing, Concurr. Pract. Exper. 2 (4) (1990) 315–339.
39. G. Burns, R. Daoud, J. Vaigl, LAM: An open cluster environment for MPI, in: Proceedings of Supercomputing Symposium, Vol. 94 (1994) pp. 379–386.
URL `http://www.lam-mpi.org/download/files/lam-papers.tar.gz`
40. W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Comput. 22 (6) (1996) 789–828.
41. T. Sterling, D. J. Becker, D. Savarese, Beowulf: A parallel workstation for scientific computation, in: Proceedings, 24th International Conference on Parallel Processing, Vol. 1 (1995) pp. 11–14.
42. D. H. M. Spector, Building Linux Clusters: Scaling Linux for Scientific and Enterprise Applications, O'Reilly & Associates, Sebastopol, CA (2000).
43. H. G. Dietz, T.I.Mattox, KLAT2's flat neighborhood network, in: Extreme Linux track of the 4th Annual Linux Showcase (2000).
44. R. Trobec, Two-dimensional regular d-meshes, Parallel Comput. 26 (13) (2000) 1945–1953.
45. U. Borštnik, M. Hodošček, D. Janežič, Improving the performance of molecular dynamics simulations on parallel clusters, J. Chem. Inf. Comput. Sci. 44 (2) (2004) 359–364.
46. J. Barnes, P. Hut, A hierarchical O(N log N) force-calculation algorithm, Nature 324 (4) (1986) 446–449.
47. J. A. Board, Jr., C. W. Humphres, C. G. Lambert, W. T. Rankin, A. Y. Toukmaji, Ewald and multipole methods for periodic N-body problems, in: P. Deuflhard, et al. (Eds.), Lecture Notes in Computational Science and Engineering, Springer-Verlag (1998).
48. J. Board, L. Schulten, The fast multipole algorithm, Comput. Sci. Eng. 2 (1) (2000) 76–79.
49. A. R. Leach, Molecular Modeling: Principles and Applications, Addison Wesley Longman Limited, Essex (1996).
50. R. Loncharich, B. Brooks, The effects of truncating long-range forces on protein dynamics, Proteins: Struct. Funct. Genet 6 (1989) 32–45.
51. S. Feller, R. Pastor, A. Rojnuckarin, S. Bogusz, B. Brooks, Effect of electrostatic force truncation on interfacial and transport properties of water, J. Phys. Chem. 100 (1996) 17011–17020.
52. R. Trobec, I. Jerebic, D. Janežič, Parallel algorithm for molecular dynamics integration, Parallel Comput. 19 (9) (1993) 1029–1039.
53. B. R. Brooks, M. Hodošček, Parallelization of CHARMm for MIMD machines, Chemical Design Auto. News 7 (1992) 16–22.
54. S. Plimpton, B. Hendrickson, Parallel molecular dynamics algorithms for simulation of molecular systems, in: T. G. Mattson (Ed.), Parallel Computing in Computational Chemistry, American Chemical Society (1995) pp. 114–132.
55. T. G. Mattson (Ed.), Parallel Computing in Computational Chemistry, American Chemical Society (1995).
56. S. J. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Chem. Phys. 117 (1) (1995) 1–19.
57. S. J. Plimpton, B. A. Hendrickson, A new parallel method for molecular-dynamics simulation of macromolecular systems, J. Comp. Chem. 17 (1996) 326–337.
58. M. Snir, A note on N-body computation with cutoffs, Tech. rep., IBM T. J. Watson Research Center (2001).
59. M. Snir, A note on n-body computations with cutoffs, Theory Comput. Systems 37 (2004) 295–318.
60. K. Bowers, R. Dror, D. Shaw, The midpoint method for parallelization of particle simulations, J. Chem. Phys. 124 (18) (2006) 184109–184109.
61. K. Bowers, R. Dror, D. Shaw, Overview of neutral territory methods for the parallel evaluation of pairwise particle interactions, J. Phys. Conf. Ser. 16 (2005) 300–304.
62. K. Moreland, E. Angel, The FFT on a GPU, in: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, ACM (2003).

63. J. Krueger, R. Westermann, Linear algebra operators for GPU implementation of numerical algorithms, ACM Trans. Graphics 22 (3) (2003) 908–916.
64. T. Narumi, R. Susukita, T. Ebisuzaki, G. McNiven, B. Elmegreen, Molecular dynamics machine: Special-purpose computer for molecular dynamics simulations, Mol. Sim. 21 (1999) 401–415.
65. T. Narumi, Special-purpose computer for molecular dynamics simulations, Doctor's thesis, University of Tokyo (1998).
66. T. Narumi, A. Kawai, T. Koishi, An 8.61 Tflop/s molecular dynamics simulation for NaCl with a special-purpose computer: MDM, in: Proceedings of SuperComputing 2001, ACM, Denver (2001).
67. M. Taiji, T. Narumi, Y. Ohno, N. Futatsugi, A. Suenaga, N. Takada, A. Konagaya, Protein explorer: A Petaflops special-purpose computer system for molecular dynamics simulations, in: Proceedings of SuperComputing 2003, ACM, Phoenix (2003).
68. U. Borštnik, D. Janežič, Symplectic molecular dynamics simulations on specially designed parallel computers, J. Chem. Inf. Model. 45 (6) (2005) 1600–1604.
69. K. Kutnar, U. Borštnik, D. Marušič, D. Janežič, Interconnection networks for parallel molecular dynamics simulation based on hamiltonian cubic symmetric topology, J. Math. Chem. 45(2) (2009) 372–385.
70. U. Borštnik, Parallel computer simulations on clusters of personal computers, Ph.D. thesis, University of Ljubljana (2007).
71. U. Borštnik, B. R. Brooks, D. Janežič, The distributed diagonal force decomposition method. I. Description of the method, submitted for publication (2008).