John von Neumann Institute for Computing

Classical Molecular Dynamics

Godehard Sutmann

published in

Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms, Lecture Notes,
J. Grotendorst, D. Marx, A. Muramatsu (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **10**, ISBN 3-00-009057-6, pp. 211-254, 2002.

http://www.fz-juelich.de/nic-series/volume10

Classical Molecular Dynamics

Godehard Sutmann

John von Neumann Institute for Computing
Central Institute for Applied Mathematics
Research Centre Jülich, 52425 Jülich, Germany
E-mail: g.sutmann@fz-juelich.de

An introduction to classical molecular dynamics simulation is presented. In addition to some historical notes, an overview is given over particle models, integrators and different ensemble techniques. In the end, methods are presented for parallisation of short range interaction potentials. The efficiency and scalability of the algorithms on massively parallel computers is discussed with an extended version of Amdahl's law.

1 Introduction

Computer simulation methods have become a very powerful tool to attack the many-body problem in statistical physics, physical chemistry and biophysics. Although the theoretical description of complex systems in the framework of statistical physics is rather well developed and the experimental techniques for detailed microscopic information are rather sophisticated, it is often only possible to study specific aspects of those systems in great detail via the simulation. On the other hand, simulations need specific input parameters that characterize the system in question, and which come either from theoretical considerations or are provided by experimental data. Having characterized a physical system in terms of model parameters, simulations are often used both to solve theoretical models beyond certain approximations and to provide a hint to experimentalists for further investigations. In the case of big experimental facilities it is even often required to prove the potential outcome of an experiment by computer simulations. In that way one can say that the field of computer simulations has developed into a very important branch of science, which on the one hand helps theorists and experimentalists to go beyond their *inherent limitations* and on the other hand is a scientific field on its own.

The traditional simulation methods for many-body systems can be divided into two classes of stochastic and deterministic simulations, which are largely covered by the Monte Carlo (MC) method and the molecular dynamics (MD) method, respectively. Monte Carlo simulations probe the configuration space by trial moves of particles. Within the so-called Metropolis algorithm, the energy change from step n to $n+1$ is used as a trigger to accept or reject the new configuration. Paths towards lower energy are always accepted, those to higher energy are accepted with a probability governed by Boltzmann statistics. In that way, properties of the system can be calculated by averaging over all Monte Carlo moves (where one move means that every degree of freedom is probed once on average). By contrast, MD methods are governed by the system's Hamiltonian and consequently Hamilton's equations of motion

$$\dot{p}_i = -\frac{\partial \mathcal{H}}{\partial q_i} \qquad , \qquad \dot{q}_i = \frac{\partial \mathcal{H}}{\partial p_i} \tag{1}$$

are integrated to move particles to new positions and to get new velocities at these new positions. This is an advantage of MD simulations with respect to MC, since not only is the configuration space probed but the whole phase space which gives additional information about the dynamics of the system. Both methods are complementary in nature but they lead to the same averages of static quantities, given that the system under consideration is ergodic and the same statistical ensemble is used.

Although there are different methods to obtain information about complex systems, particle simulations always require a model for the interaction between system constituants. This model has to be tested against experimental results, i.e. it should reproduce or approximate experimental findings like distribution functions or phase diagrams, and theoretical constraints, i.e. it should obey certain fundamental or limiting laws like energy conservation.

Concerning MD simulations the ingredients for a program are basically threefold:
(i) As already mentioned, a model for the interaction between system constituants (atoms, molecules, surfaces etc.) is needed. Often, it is assumed that particles interact only pairwise, which is exact e.g. for particles with fixed partial charges. This assumption greatly reduces the computational effort and the work to implement the model into the program.
(ii) An integrator is needed, which propagates particle positions and velocities from time t to $t + \delta t$. It is a finite difference scheme which moves trajectories discretely in time. The time step δt has properly to be chosen to guarantee stability of the integrator, i.e. there should be no drift in the system's energy.
(iii) A statistical ensemble has to be chosen, where thermodynamic quantities like pressure, temperature or the number of particles are controlled. The natural choice of an ensemble in MD simulations is the microcanonical ensemble (NVE), since the system's Hamiltonian without external potentials is a conserved quantity. Nevertheless, there are extensions to the Hamiltonian which also allow to simulate different statistical ensembles.

These steps essentially define an MD simulation. Having this tool at hand, it is possible to obtain *exact* results within numerical precision. Results are only correct with respect to the model which enters into the simulation and they have to be tested against theoretical predictions and experimental findings. If the simulation results differ from the *real system* properties or are incompatible with *solid* theoretical manifestations, the model has to be refined. This procedure can be understood as an adaptive refinement which leads in the end to an approximation of a model of the *real world* at least for certain properties. The model itself may be constructed from plausible considerations, where parameters are chosen from neutron diffraction or NMR measurements. It may also result from first principle investigations, like quantum *ab initio* calculations. Although the electronic distribution of the particles is calculated very accurately, this type of model building contains also some approximations, since many-body interactions are mostly neglected (this would increase the parameter space in the model calculation enormously). However, it often provides a good starting point for a realistic model.

An important issue of simulation studies is the accessible time- and length-scale coverable by microscopic simulations. Fig.1 shows a schematic representation for different types of simulations in a *length-time-diagram*. It is clear that the more detailed a simulation technique operates, the smaller is the accessibility of long times and large length scales. Therefore quantum simulations, where fast motions of electrons are taken into account, are located in the lower left corner of the diagram and typical length and time

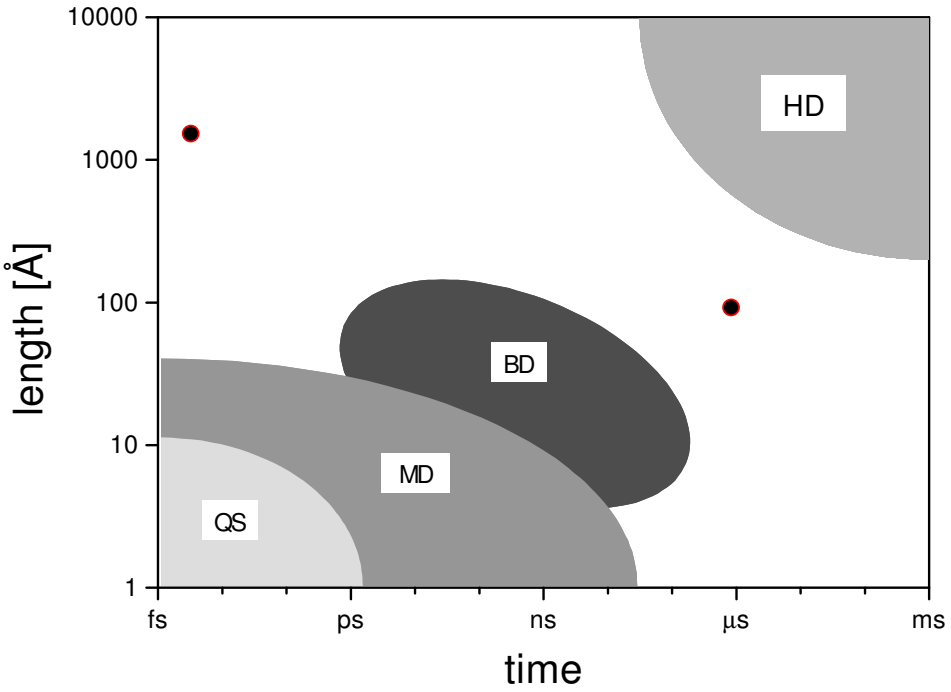


Figure 1. Schematic comparison of time- and length-scales, accessible to different types of simulation techniques (quantum simulations (QM), molecular dynamics (MD), Brownian dynamics (BD) and hydrodynamics/fluid dynamics (HD)). The black dots mark the longest ($\approx$ 1 μs) and the biggest ($N > 5 \times 10^9, L \approx 0.4 \mu m$ molecular dynamics simulations by Duan & Kollman[1] and Roth[2] respectively.)

scales are of order of Å and ps. Classical molecular dynamics approximates electronic distributions in a rather coarse-grained fashion by putting either fixed partial charges on interaction sites or by adding an approximate model for polarization effects. In both cases, the time scale of the system is not dominated by the motion of electrons, but the time of intermolecular collision events, rotational motions or intramolecular vibrations, which are orders of magnitude slower than those of electron motions. Consequently, the time step of integration is larger and trajectory lengths are of order ns and accessible lengths of order $10 - 100$ Å. If one considers tracer particles in a solvent medium, where one is not interested in a detailed description of the solvent, one can apply Brownian dynamics, where the effect of the solvent is hidden in average quantities. Since collision times between tracer particles is very long, one may apply larger timesteps. Furthermore, since the solvent is not simulated explicitly, the lengthscales may be increased considerably. Finally, if one is interested not in a microscopic picture of the simulated system but in macroscopic quantities, the concepts of hydrodynamics may be applied, where the system properties are hidden in effective numbers, e.g. density, viscosity, sound velocity.

It is clear that the performance of particle dynamics simulations strongly depends on the computer facilities at hand. The first studies using MD simulation techniques were performed in 1957 by B. J. Alder and T. E. Wainright[3] who simulated the phase transition

of a system of hard spheres. The general method, however, was presented two years later[4]. In this early simulation, which was run an an IBM-704, up to 500 particles could be simulated, for which 500 collisions per hour could be calculated. Taking into account 200000 collisions for a production run, these simulations lasted for more than two weeks. The propagation of hard spheres in a simulation is determined by the collision events between two particles. Therefore, the propagation is not based on an integration of the equations of motion, but rather the calculation of the time of the next collision, which results in a variable time step in the calculations.

The first MD simulation which was applied to atoms interacting via a continuous potential was performed by A. Rahman in 1964. In this case, a model system for Argon was simulated and not only binary collisions were taken into account but the interactions were modeled by a Lennard-Jones potential and the equations of motion were integrated with a finite difference scheme. This work may be considered as seminal for dynamical calculations. It was the first work where an exact method (within numerical precision) was used to calculate dynamical quantities like autocorrelation functions and transport coefficients like the diffusion coefficient for a realistic system. Also more involved topics like the dynamic van Hove function and non-Gaussian corrections to diffusion were evaluated. The calculations were performed for 864 particles on a CDC 3600, where the propagation of all particles for one time step took $\approx 45\ s$. The calculation of 50000 timesteps then took more than three weeks! [a]

With the development of faster and bigger massively parallel architectures the accessible time and length scales are increasing. In the case of classical MD simulations it was demonstrated by J. Roth in 1999 on the CRAY T3E-1200 in Jülich that it is possible to simulate more than 5×10^9 particles, corresponding to a length scale of several 1000 Å. This was possible with the highly memory optimised MD program IMD[5,2], which used the 512 nodes with 256 MB memory each, quite efficiently. However, the limits of such a demonstration became rather obvious, since for a usual production run of 10000 time steps a simulation time of a quarter of a year would be required (given that the whole machine is dedicated to one user). In another demonstration run Y. Duan and P. A. Kollman extended the time scale of an all atom MD simulation to $1\ \mu s$, where they simulated the folding process of the subdomain HP-36 from the villin headpiece[6,1]. The protein was modelled with a 596 interaction site model dissolved in a system of 3000 water molecules. Using a timestep of integration of $2 \times 10^{-15} s$, the program was run for 5×10^8 steps. In order to perform this type of calculation, it was necessary to run the program several months on a CRAY T3D and CRAY T3E with 256 processors. It is clear that such kind of simulation is exceptional due to the large amount of computer resources needed, but is nonetheless a kind of milestone pointing to future simulation practices.

Classical molecular dynamics methods are nowadays applied to a huge class of problems, e.g. properties of liquids, defects in solids, fracture, surface properties, friction, molecular clusters, polyelectrolytes and biomolecules. Due to the large area of applicability, simulation codes for molecular dynamics were developed by many groups. On the internet homepage of the Collaborative Computational Project No.5 (CCP5)[7] a lot of computer codes are assembled for condensed phase dynamics. During the last years several programs were designed for parallel computers. Among them, which are partly avail-

[a] On a standard PC this calculation may be done within one hour nowadays!

able free of charge, are, e.g., Amber/Sander[8], CHARMM[9], NAMD[10], NWCHEM[11] and LAMMPS[12].

2 Models for Particle Interactions

A system is completely determined through it's Hamiltoian $\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_1$, where $\mathcal{H}_0$ is the *internal* part of the Hamiltonian, given as

$$\mathcal{H}_0 = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + \sum_{i<j}^{N} u(\mathbf{r}_i, \mathbf{r}_j) + \sum_{i<j}^{N} u^{(3)}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \dots \tag{2}$$

where $\mathbf{p}$ is the momentum, m the mass of the particles and u and $u^{(3)}$ are pair and three-body interaction potentials. $\mathcal{H}_1$ is an external part, which can include time dependent effects or external sources for a force. All simulated objects are defined within a model description. Often a precise knowledge of the interaction between atoms, molecules or surfaces are not known and the model is constructed in order to describe the main features of some observables. Besides boundary conditions, which are imposed, it is the model which completely determines the system from the physical point of view. In classical simulations the *objects* are most often described by point-like centers which interact through pair- or multibody interaction potentials. In that way the highly complex description of electron dynamics is abandoned and an effective picture is adopted where the main features like the hard core of a particle, electric multipoles or internal degrees of freedom of a molecules are modeled by a set of parameters and (most often) analytical functions which depend on the mutual position of particles in the configuration. Since the parameters and functions give a complete information of the system's energy as well as the force acting on each particle through $\mathbf{F} = -\nabla U$, the combination of parameters and functions is also called a *force field*. Different types of force field were developed during the last ten years. Among them are e.g. MM3[13], MM4[14], Dreiding[15], SHARP[16], VALBON[17], UFF[18], CFF95[19], AMBER[20] CHARMM[21], OPLS[22] and MMFF[23]. Typical examples for force field functions are summerized in Fig. 2.

There are major differences to be noticed for the potential forms. The first distinction is to be made between pair- and multibody potentials. In systems with no constraints, the interaction is most often described by pair potentials, which is simple to implement into a program. In the case where multibody potentials come into play, the counting of interaction partners becomes increasingly more complex and dramatically slows down the execution of the program. Only for the case where interaction partners are known in advance, e.g. in the case of torsional or bending motions of a molecule can the interaction be calculated efficiently by using neighbor lists or by an intelligent way of indexing the molecular sites.

A second important difference between interactions is the spatial extent of the potential, classifying it into short and long range interactions. If the potential drops down to zero faster than r^{-d}, where r is the separation between two particles and d the dimension of the problem, it is called short ranged, otherwise it is long ranged. This becomes clear by considering the integral

$$I = \int \frac{dr^d}{r^n} = \begin{cases} \infty & : n \leq d \\ \text{finite} & : n > d \end{cases} \tag{3}$$

i.e. a particles' potential energy gets contributions from *all particles of the universe* if $n \leq d$, otherwise the interaction is bound to a certain region, which is often modeled by a spherical interaction range. The long range nature of the interaction becomes most important for potentials which only have potential parameters of the same sign, like the gravitational potential where no screening can occur. For Coulomb energies, where positive and negative charges may compensate each other, long range effects may be of minor importance in some systems like molten salts. In the following two examples shall illustrate the different treatment of short- and long range interactions.

2.1 Short Range Interactions

Short range interactions offer the possibility to take into account only neigbored particles up to a certain distance for the calculation of interactions. In that way a cutoff radius is introduced beyond of which mutual interactions between particles are neglected. As an approximation one may introduce *long range corrections* to the potential in order to compensate for the neglect of explicit calculations. The whole short range potential may then be written as

$$U = \sum_{i<j}^{N} u(r_{ij}|r_{ij} < R_c) + U_{lrc} \tag{4}$$

The long-range correction is thereby given as

$$U_{lrc} = 2\pi N \rho_0 \int_{R_c}^{\infty} dr\, r^2 g(r) u(r) \tag{5}$$

where ρ_0 is the number density of the particles in the system and $g(r) = \rho(r)/\rho_0$ is the radial distribution function. For computational reasons, $g(r)$ is most often only calculated up to R_c, so that in practice it is assumed that $g(r) = 1$ for $r > R_c$, which makes it possible for many types of potentials to calculate U_{lrc} analytically.

Besides internal degrees of freedom of molecules, which may be modeled with short range interaction potentials (c.f. Fig.2), it is first of all the excluded volume of a particle which is of importance. A finite diameter of a particle may be represented by a steep repulsive potential acting at short distances. This is either described by an exponential function or an algebraic form, $\propto r^{-n}$, where $n \geq 9$. Another source of short range interaction is the van der Waals interaction. For neutral particles these are the London forces arising from induced dipole interactions. Fluctuations of the electron distribution of a particle give rise to fluctuating dipole moments, which on average compensate to zero. But the instantaneous created dipoles induce also dipoles on neighbored particles which attract each other $\propto r^{-6}$. Two common forms of the resulting interactions are the Buckingham potential

$$u^B_{\alpha\beta}(r_{ij}) = A_{\alpha\beta} e^{-B_{\alpha\beta} r_{ij}} - \frac{D_{\alpha\beta}}{r^6_{ij}} \tag{6}$$

and the Lennard-Jones potential

$$u^{LJ}_{\alpha\beta}(r_{ij}) = 4\epsilon_{\alpha\beta} \left(\left(\frac{\sigma_{\alpha\beta}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{\alpha\beta}}{r_{ij}} \right)^{6} \right) \tag{7}$$

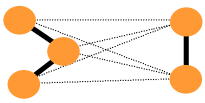
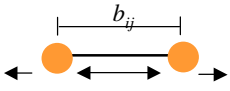
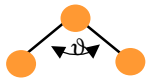
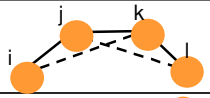
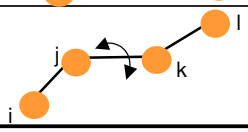
6-9 van der Waals 6-12 van der Waals Electrostatic		$u_{ij}(r_{ij}) = 4\varepsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^9 - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^6\right)$ $u_{ij}(r_{ij}) = 4\varepsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^6\right)$ $u_{ij}(r_{ij}) = \frac{q_i q_j}{r_{ij}}$
Quadratic bond-stretching Morse bond-stretching		$u_{ij}^s(r_{ij}) = \frac{1}{2}k_{ij}(r_{ij} - b_{ij})^2$ $u_{ij}(r_{ij}) = k\left(1 - e^{-a(r_{ij}-r_0)}\right)^2$
Bond-bending		$u_{ij}^b(\vartheta_{ijk}) = \frac{1}{2}k_{ijk}(\vartheta_{ijk} - \vartheta_{ijk}^0)^2$
Improper dihedrals		$u_{ij}^{id}(\xi_{ijkl}) = \frac{1}{2}k_{ijkl}(\xi_{ijkl} - \xi_0)^2$
Proper dihedrals		$u_{ij}^{pd}(\varphi_{ijkl}) = k_\varphi(1 + \cos(n\varphi_{ijkl} - \varphi_0))$

Figure 2. Typical examples for potential terms as used in common force-fields.

which are compared in Fig.3. In Eqs.6,7 the indices α, β indicate the species of the particles, i.e. there are parameters A, B, D in Eq.6 and ϵ, σ in Eq.7 for intra-species interactions ($\alpha = \beta$) and cross species interactions ($\alpha \neq \beta$). For the Lennard-Jones potential the parameters have a simple physical interpretation: ϵ is the minimum potential energy, located at $r = 2^{1/6}\sigma$ and σ is the diameter of the particle, since for $r < \sigma$ the potential becomes repulsive. Often the Lennard-Jones potential gives a reasonable approximation of a *true* potential. However, from exact quantum ab initio calculations an exponential type repulsive potential is often more appropriate. Especially for dense systems the too steep repulsive part often leeds to an overestimation of the pressure in the system. Since computationally the Lennard-Jones interaction is quite attractive the repulsive part is sometimes replaced by a weaker repulsive term, like $\propto r^{-9}$. The Lennard-Jones potential has another advantage over the Buckingham potential, since there are combining rules for the parameters. A common choice are the Lorentz-Berelot combining rules

$$\sigma_{\alpha\beta} = \frac{\sigma_{\alpha\alpha} + \sigma_{\beta\beta}}{2} \quad , \quad \epsilon_{\alpha\beta} = \sqrt{\epsilon_{\alpha\alpha}\epsilon_{\beta\beta}} \tag{8}$$

This combining rule is, however, known to overestimate the well depth parameter. Two

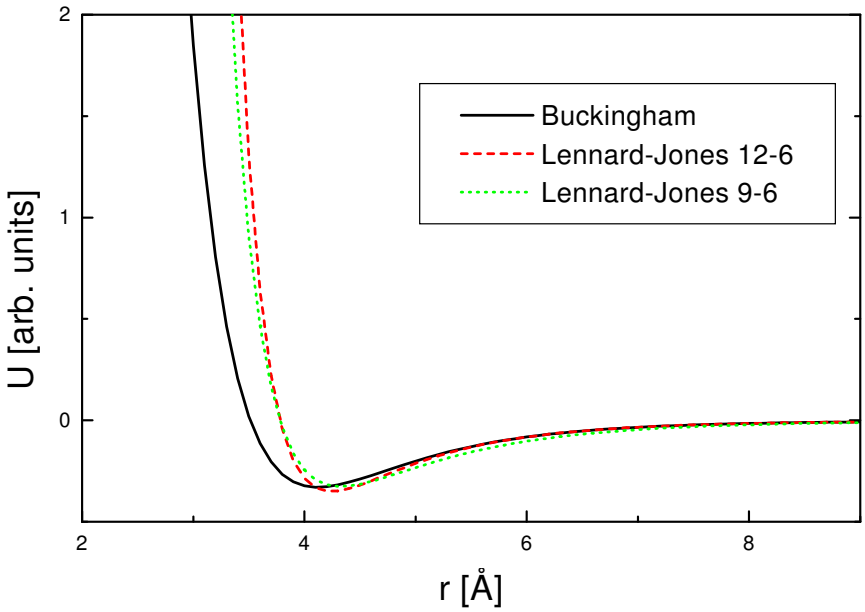


Figure 3. Comparison between a Buckingham-, Lennard-Jones (12-6) and Lennard-Jones (9-6) potential.

other commonly known combining rules try to correct this effect, which are Kong[24] rules

$$\sigma_{\alpha\beta} = \left[\frac{1}{2^{13}} \frac{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{12}}{\sqrt{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{6}\epsilon_{\beta\beta}\sigma_{\beta\beta}^{6}}} \left(1 + \left(\frac{\epsilon_{\beta\beta}\sigma_{\beta\beta}^{12}}{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{12}} \right)^{1/13} \right)^{13} \right]^{1/6} \tag{9}$$

$$\epsilon_{\alpha\beta} = \frac{\sqrt{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{6}\epsilon_{\beta\beta}\sigma_{\beta\beta}^{6}}}{\sigma_{\alpha\beta}^{6}} \tag{10}$$

and the Waldman-Kagler[25] rule

$$\sigma_{\alpha\beta} = \left(\frac{\sigma_{\alpha\alpha}^{6} + \sigma_{\beta\beta}^{6}}{2} \right)^{1/6} \quad , \quad \epsilon_{\alpha\beta} = \frac{\sqrt{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{6}\epsilon_{\beta\beta}\sigma_{\beta\beta}^{6}}}{\sigma_{\alpha\beta}^{6}} \tag{11}$$

In a recent study[26] of Ar-Kr and Ar-Ne mixtures, these combining rules were tested and it was found that the Kong rules give the best agreement between simulated and experimental pressure-density curves. An illustration of the different combining rules is shown in Fig.4 for the case of an Ar-Ne mixture.

2.2 Long Range Interactions

In the case of long range potentials, like the Coulomb potential, interactions between all particles in the system must be taken into account, if treated without any approximation.

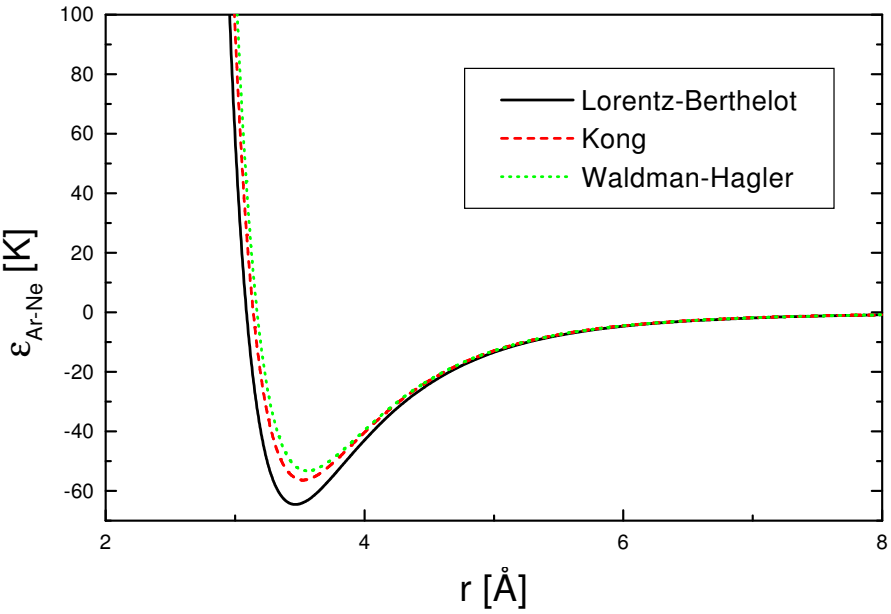


Figure 4. Resulting cross-terms of the Lennard-Jones potential for an Ar-Ne mixture. Shown is the effect of different combining rules (Eqs.8-11). Parameters used are $\sigma_{Ar} = 3.406$ Å, $\epsilon_{Ar} = 119.4\ K$ and $\sigma_{Ne} = 2.75$ Å, $\epsilon_{Ne} = 35.7\ K$.

This leads to an $\mathcal{O}(N^2)$ problem, which increases considerably the execution time of a program for larger systems. For systems with open boundary conditions (like liquid droplets), this method is straightforwardly implemented and reduces to a double sum over all pairs of particles. In the case when periodic boundary conditions are applied, not only the interactions with particles in the *central cell* but also with all periodic images must be taken into account and formally a lattice sum has to be evaluated

$$U = \frac{1}{2} \sum_{i,j=1}^{N} \sum_{\mathbf{n}}{}' \frac{q_i q_j}{|\mathbf{r}_{ij} - \mathbf{n}L|} \tag{12}$$

where $\mathbf{n}$ is a lattice vector and $\sum_{\mathbf{n}}{}'$ means that for $\mathbf{n} = 0$ it is $i \neq j$. It is, however, a well known problem that this type of lattice sum is conditionally convergent, i.e. the result depends on the sequence of evaluating the sum (see e.g.[27]). A method to overcome this limitation was invented by Ewald[28]. The idea is to introduce a convergence factor into the sum of Eq.12 which depends on a parameter s, evaluate the sum and put $s \to 0$ in the end. A characterization of the convergence factors was given in Ref.[29,30]. A form which leads to the Ewald sum is an exponential $e^{-s\mathbf{n}^2}$, transforming Eq.12 into

$$U(s) = \frac{1}{2} \sum_{i,j=1}^{N} \sum_{\mathbf{n}}{}' \frac{q_i q_j}{|\mathbf{r}_{ij} - \mathbf{n}L|}\, e^{-s\mathbf{n}^2} \tag{13}$$

The evaluation and manipulation of this equation proceeds now by using the definition for the Γ-function and the Jacobi imaginary transform. A very instructive way of the derivation

of the Ewald sum may be found in Ref.[29,30]; a heuristic derivation is given Ref.[31]. For brevity, only the final form of the sum is given here

$$U = \frac{1}{2}\Bigg\{ \underbrace{\sum_{i,j=1}^{N} \sum_{\mathbf{n}}{}' \frac{q_i q_j \mathrm{erfc}(\alpha|\mathbf{r}_{ij} - \mathbf{n}L|/L)}{|\mathbf{r}_{ij} - \mathbf{n}L|}}_{U_{real}} + \underbrace{\frac{4\pi q_i q_j}{L^3} \sum_{\mathbf{k}} \frac{1}{k^2} e^{i\mathbf{k}\mathbf{r}_{ij}} e^{-k^2/4\alpha^2}}_{U_{reciprocal}} + \underbrace{\frac{1}{L}\left[\sum_{\mathbf{n}\neq\mathbf{0}} \frac{\mathrm{erfc}(\alpha\mathbf{n})}{|\mathbf{n}|} + \frac{e^{-\pi^2\mathbf{n}^2/\alpha^2}}{\pi\mathbf{n}^2} - \frac{2\alpha}{\sqrt{\pi}}\right] \sum_{i=1}^{N} q_i^2}_{U_{self}} + \underbrace{\frac{4\pi}{L^3}\left|\sum_{i=1}^{N} q_i\right|^2}_{U_{surface}} \Bigg\} \tag{14}$$

The evaluation of the potential thus splits into four different terms, where the so called *self-* and *surface-terms* are constant and may be calculated in the beginning of a simulation. The first two sums, however, depend on the interparticle separations $\mathbf{r}_{ij}$, which need to be evaluated in each time step. It is seen that the lattice sum is essentially split into a sum which is evaluated in real space and a sum over reciprocal space-vectors, $\mathbf{k} = 2\pi\mathbf{n}/L$. The parameter α appears formally in the derivation as a result of an integral splitting but it has a very intuitive physical meaning. The first sum gives the potential of a set of point charges which are screened by an opposite charge of the same magnitude but with a Gaussian form factor where the width of the Gaussian is given by α. The second sum subtracts this screening charge, but the sum is evaluated in reciprocal space. Since $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$ decays as e^{-x^2} for large x, the first sum contains mainly short range contributions. On the other side, the second sum decays strongly for large k-vectors and thus contains mainly long range contributions. Most often, the parameter α is chosen in way to reduce the evaluation of the real space sum to the central simulation cell. Often, a spherical cutoff is then applied for this term, i.e. contributions of particle pairs, separated in a distance $|\mathbf{r}_{ij}| > R_c$ are neglected. On the other hand, the reciprocal space sum is conventionally truncated after a maximum wave-vector $\mathbf{k}_{max}$. All three parameters $\alpha, R_c, \mathbf{k}_{max}$ may be chosen in an optimal way to balance the truncation error in each sum and the number of operations. This balancing even leads to the effect that the Ewald sum may be tuned[32,33] to scale with $\mathcal{O}(N^{3/2})$ (for fast methods which have better scaling characteristics, see Ref.[31]). A detailed analysis of the individual errors occuring in the different sums was given in Ref.[34]. An alternative derivation of the Ewald sum starts directly by assuming a Gaussian form factor for the screening charge. This gives the opportunity to investigate also form factors, differing from a Gaussian. In these cases the convergence function is in general not known but it is assumed to exist. Different form factors were studied systematically in Ref.[35].

The present form of the Ewald sum gives an exact representation of the potential energy of point like charges in a system with periodic boundary conditions. Sometimes the charge distribution in a molecule is approximated by a point dipole or higher multipole moments. A more general form of the Ewald sum, taking into account arbitrary point multipoles was given in Ref.[36]. The case, where also electronic polarizabilities are considered is given in Ref.[37].

In certain systems, like in molten salts or electrolyte solutions, the interaction between charged species may approximated by a screened Coulomb potential, which has a Yukawa-

like form

$$U = \frac{1}{2} \sum_{i,j=1}^{N} q_i q_j \frac{e^{-\kappa |\mathbf{r}_{ij}|}}{|\mathbf{r}_{ij}|} \tag{15}$$

The parameter κ is the inverse Debye length, which gives a measure of screening strength in the system. If $\kappa < 1/L$ the potential is short ranged and usual cut-off methods may be used. Instead, if $\kappa > 1/L$, or generally if $u(r = L/2)$ is larger than the prescribed uncertainties in the energy, the minimum image convention in combination with truncation methods fails and the potential must be treated in a more rigorous way, which was proposed in Ref.[38], where an extension of the Ewald sum for such Yukawa type potentials was developed.

3 The Integrator

For a given potential model which characterizes the physical system, it is the integrator which is responsible for the accuracy of the simulation results. If the integrator would work without any error the simulation would provide *exact model results* within the errors occuring due to a finite number representation. However, any finite difference integrator is naturally an approximation for a system developing continuously in time. The requirements for the integrator are therefore to be

- accurate, in the sense that it approximates the *true* trajectory very well (this may be checked with simple model systems for which analytical solutions exist)
- stable, in the sense that it conserves energy and that small perurbations do not lead to instabilities
- robust, in the sense that it allows for large time steps in order to propagate the system efficiently through phase space

In the following different types of integration schemes are presented. First, simple integrators based on Taylor expansions are shown. Later on integrators based on an operator splitting method are discussed which provide the possibility to introduce in an elegant way the integrations of motion on different time scales. Finally, attention is given to more complex situations where molecules with orientational degrees of freedom are considered.

3.1 Expansion Based Integrators

The simplest and most straight forward way to construct an integrator is by expanding the positions and velocities in a Taylor series. The class of integrators which may be obtained in that way are called Verlet-Störmer integrators[39,40]. For a small enough time step δt the expansion gives

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \mathbf{v}(t)\ \delta t + \frac{1}{2}\mathbf{a}(t)\delta t^2 + \frac{1}{6}\mathbf{b}(t)\delta t^3 + \ldots \tag{16}$$

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \mathbf{a}(t)\ \delta t + \frac{1}{2}\mathbf{b}(t)\delta t^2 + \frac{1}{6}\mathbf{c}(t)\delta t^3 + \ldots \tag{17}$$

where $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are the 2nd, 3rd and 4th time derivative of the coordinates. In the same way the expansion may be performed for $\delta t \rightarrow -\delta t$, which gives

$$\mathbf{r}(t-\delta t) = \mathbf{r}(t) - \mathbf{v}(t)\ \delta t + \frac{1}{2}\mathbf{a}(t)\delta t^2 - \frac{1}{6}\mathbf{b}(t)\delta t^3 \pm \ldots \tag{18}$$

$$\mathbf{v}(t-\delta t) = \mathbf{v}(t) - \mathbf{a}(t)\ \delta t + \frac{1}{2}\mathbf{b}(t)\delta t^2 - \frac{1}{6}\mathbf{c}(t)\delta t^3 \pm \ldots \tag{19}$$

Adding up Eqs.16,18 and Eqs.17,19 gives for the new positions and velocities

$$\mathbf{r}(t+\delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\delta t) + \mathbf{a}(t)\delta t^2 + \mathcal{O}(\delta t^4) \tag{20}$$

$$\mathbf{v}(t+\delta t) = 2\mathbf{v}(t) - \mathbf{v}(t-\delta t) + \mathbf{b}(t)\delta t^2 + \mathcal{O}(\delta t^4) \tag{21}$$

A method whose truncation varies as $\delta t^{(n+1)}$ is called an n-th order method. Eqs.20,21 are therefore of 3rd order.The drawback of Eq.21 is, however, that it requires the 3rd derivative of the coordinates with respect with to time which is not routinely calculated in MD simulations and thus introduces some additional computational and storage overhead. To overcome this drawback one can simply substract Eq.18 from Eq.16, giving the central difference scheme for the velocity

$$\mathbf{v}(t) = \frac{1}{2\delta t}(\mathbf{r}(t+\delta t) - \mathbf{r}(t-\delta t)) + \mathcal{O}(\delta t^3) \tag{22}$$

This is, however, one order less in accuracy than Eq.21 and also provides velocities at timestep t, not at $t+\delta t$. Since this information i not required by Eq.20 to calculate accurately the positions, one may take Eq.22 as an estimate for the velocities from which the kinetic energy of the system is calculated.

From the point of view of storage requirements, Egs.20,22 are not optimal, since information is required form positions not only at time t but also at time $t-\delta t$. An equivalent algorithm, which stores only information from one timestep is the so called *velocity Verlet* algorithm, whic reads

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \mathbf{v}(t)\ \delta t + \frac{1}{2}\mathbf{a}(t)\delta t^2 \tag{23}$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \frac{1}{2}\delta t(\mathbf{a}(t) + \mathbf{a}(t+\delta t)) \tag{24}$$

This scheme, however, requires the knowledge of the accelerations, $\mathbf{a}$, at timestep $t+\delta t$. One may therefore decompose Eq.24 into two steps. First calculate

$$\mathbf{v}(t+\delta t/2) = \mathbf{v}(t) + \frac{1}{2}\delta t\mathbf{a}(t) \tag{25}$$

then compute the actual forces on the particles at time $t+\delta t$ and finish the velocity calculation with

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t+\delta t/2) + \frac{1}{2}\mathbf{a}(t+\delta t)\delta t \tag{26}$$

At this point the kinetic energy may be calculated without a time delay of δt, as it was in Eq.22. Several other schemes have been proposed in literature, such as the leap-frog[41]

scheme or Beeman's[42] algorithm. They all have the same accuracy and should produce identical trajectories in coordinate space [b]

3.2 Operator Splitting Methods

A more rigorous derivation, which in addition leads to the possibility of splitting the propagator of the phase space trajectory into several time scales, is based on the phase space description of a classical system. The time evolution of a point in the $6N$ dimensional phase space is given by the Liouville equation

$$\mathbf{\Gamma}(t) = e^{i\mathcal{L}t}\,\mathbf{\Gamma}(0) \tag{27}$$

where $\mathbf{\Gamma} = (\mathbf{q}, \mathbf{p})$ is the $6N$ dimensional vector of generalized coordinates, $\mathbf{q} = \mathbf{q}_1, \ldots, \mathbf{q}_N$, and momenta, $\mathbf{p} = \mathbf{p}_1, \ldots, \mathbf{p}_N$. The Liouville operator, $\mathcal{L}$, is defined as

$$i\mathcal{L} = \{\ldots, \mathcal{H}\} = \sum_{j=1}^{N} \left(\frac{\partial \mathbf{q}_j}{\partial t}\frac{\partial}{\partial \mathbf{q}_j} + \frac{\partial \mathbf{p}_j}{\partial t}\frac{\partial}{\partial \mathbf{p}_j} \right) \tag{28}$$

In order to construct a discrete timestep integrator, the Liouville operator is split into two parts, $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$, and a Trotter expansion[43] is performed

$$e^{i\mathcal{L}\delta t} = e^{i(\mathcal{L}_1+\mathcal{L}_2)\delta t} \tag{29}$$

$$= e^{i\mathcal{L}_1\delta t/2} e^{i\mathcal{L}_2\delta t} e^{i\mathcal{L}_1\delta t/2} + \mathcal{O}(\delta t^3) \tag{30}$$

The partial operators can be chosen to act only on positions and momenta. Assuming usual cartesian coordinates for a system of N free particles, this can be written as

$$i\mathcal{L}_1 = \sum_{j=1}^{N} \mathbf{F}_j \frac{\partial}{\partial \mathbf{p}_j} \tag{31}$$

$$i\mathcal{L}_2 = \sum_{j=1}^{N} \mathbf{v}_j \frac{\partial}{\partial \mathbf{r}_j} \tag{32}$$

Applying Eq.29 to the phase space vector $\mathbf{\Gamma}$ and using the property $e^{a\partial/\partial x} f(x) = f(x+a)$ for any function f, where a is independent of x, gives

$$\mathbf{v}_i(t + \delta t/2) = \mathbf{v}(t) + \frac{\mathbf{F}_i(t)}{m_i}\frac{\delta t}{2} \tag{33}$$

$$\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t + \delta t/2)\delta t \tag{34}$$

$$\mathbf{v}_i(t + \delta t) = \mathbf{v}_i(t + \delta t/2) + \frac{\mathbf{F}_i(t + \delta t)}{m_i}\frac{\delta t}{2} \tag{35}$$

which is the velocity Verlet algorithm, Eqs.23,25,26.

[b] This statement is derived from the point of view of accuracy. Since numerical operations are in general not associative a differnt implementation of an algorithm will have different round off errors and therefore the accumulation of the roundoff error will accumulate which will lead in practice to a deviation from the above statement.

In the same spirit, another algorithm may be derived by simply changing the definitions for $\mathcal{L}_1 \rightarrow \mathcal{L}_2$ and $\mathcal{L}_2 \rightarrow \mathcal{L}_1$. This gives the so called *position Verlet algorithm*

$$\mathbf{r}_i(t + \delta t/2) = \mathbf{r}_i(t) + \mathbf{v}(t)\frac{\delta t}{2} \tag{36}$$

$$\mathbf{v}_i(t + \delta t) = \mathbf{v}(t) + \frac{\mathbf{F}_i(t + \delta t/2)}{m_i} \tag{37}$$

$$\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t + \delta t/2) + (\mathbf{v}(t) + \mathbf{v}_i(t + \delta t))\frac{\delta t}{2} \tag{38}$$

Here the forces are calculated at intermediate positions $\mathbf{r}_i(t+\delta t/2)$. The equations of both the velocity Verlet and the position Verlet algorithms have the property of propagating velocities or positions on half time steps. Since both schemes decouple into an applied force term and a *free flight* term, the three steps are often called *half-kick/drift/half kick* for the velocity Verlet and correspondingly *half-drift/kick/half-drift* for the position Verlet algorithm.

Both algorithms, the velocity and the position Verlet method, are examples for symplectic algorithms, which are characterized by a volume conservation in phase space. This is equivalent to the fact that the Jacobian matrix of a transform $x' = f(x, p)$ and $p' = g(x, p)$ satisfies

$$\begin{pmatrix} f_x & f_p \\ g_x & g_p \end{pmatrix} \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \begin{pmatrix} f_x & f_p \\ g_x & g_p \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \tag{39}$$

Any method which is based on the splitting of the Hamiltonian, is symplectic. This does not yet, however, guarantee that the method is also time reversible, which may be also be considered as a strong requirement for the integrator. This property is guaranteed by symmetric methods, which also provide a better numerical stability[44]. Methods, which try to enhance the accuracy by taking into account the particles' history (multi-step methods) tend to be incompatible with symplecticness[45,46], which makes symplectic schemes attractive from the point of view of data storage requirements. Another strong argument for symplectic schemes is the so called *backward error ananlysis*[47–49]. This means that the trajectory produced by a discrete integration scheme, may be expressed as the solution of a perturbed ordinary diffential equation whose *rhs* can formally be expressed as a power series in δt. It could be shown that the system, described by the ordinary differential equation is Hamiltonian, if the integrator is symplectic[50,51]. In general, the power series in δt diverges. However, if the series is truncated, the trajectory will differ only as $\mathcal{O}(\delta t^p)$ of the trajectory, generated by the symplectic integrator on timescales $\mathcal{O}(1/\delta t)$[52].

3.3 Multiple Time Step Methods

It was already mentioned that the rigorous approach of the decomposition of the Liouville operator offers the opportunity for a decomposition of time scales in the system. Supposing that there are different time scales present in the system, e.g. fast intramolecular vibrations and slow domain motions of molecules, then the factorization of Eq.29 may be written in

a more general way

$$e^{i\mathcal{L}\Delta t} = e^{i\mathcal{L}_1^{(s)}\Delta t/2} e^{i\mathcal{L}_1^{(f)}\Delta t/2} e^{i\mathcal{L}_2\delta t} e^{i\mathcal{L}_1^{(f)}\Delta t/2} e^{i\mathcal{L}_1^{(s)}\Delta t/2} \tag{40}$$

$$= e^{i\mathcal{L}_1^{(s)}\Delta t/2} \left\{ e^{i\mathcal{L}_1^{(f)}\delta t/2} e^{i\mathcal{L}_2\delta t} e^{i\mathcal{L}_1^{(f)}\delta t/2} \right\}^p e^{i\mathcal{L}_1^{(s)}\Delta t/2} \tag{41}$$

where the time increment is $\Delta t = p\delta$. The decomposition of the Liouville operator may be chosen in the convenient way

$$i\mathcal{L}_1^{(s)} = \mathbf{F}_i^{(s)} \frac{\partial}{\partial \mathbf{p}_i} \quad , \quad i\mathcal{L}_1^{(f)} = \mathbf{F}_i^{(f)} \frac{\partial}{\partial \mathbf{p}_i} \quad , \quad i\mathcal{L}_2 = \mathbf{v}_i \frac{\partial}{\partial \mathbf{q}_i} \tag{42}$$

where the superscript (s) and (f) mean slow and fast contributions to the forces. The idea behind this decomposition is simply to take into account contributions from slowly varying components only every p'th timestep with a large time interval. Therefore, the force computation may be considerably speeded up in the the $p-1$ intermediate force computation steps. In general, the scheme may be extended to account for more time scales. Examples for this may be found in Refs.[53–55]. One obvious problem, however, is to separate the timescales in a proper way. The scheme of Eq.41 is *exact* if the time scales decouple completely. This, however, is very rarely found and most often timescales are coupled due to nonlinear effects. Nevertheless, for the case where Δt is not very much larger than δt ($p \approx 10$), the separation may be often justified and lead to stable results. Another criteria for the separation is to distinguish between long range and short range contributions to the force. Since the magnitude and the fluctuation frequency is very much larger for the short range contributions this separation makes sense for speeding up computations including long range interactions[56].

The method has, however, its limitations[57,58]. As described, a particle gets every n'th timestep a *kick* due to the slow components. It was reported in literature that this may excite a system's resonance which will lead to strong artifacts or even instabilities[59,60]. Recently different schemes were proposed to overcome these resonances by keeping the property of symplecticness[61–67].

3.4 Constraint Dynamics

The methods discussed so far are quite general for the cases of free particles. If constraints are applied, e.g. a fixed bond length between particles in a molecule or a fixed bond angle, the integration scheme has either to be modified or extended. A modification of the integrator means that the equations of motion have to be formulated for rotational and translational degrees of freedoms. On the other hand an extension of the integrator means that constraints have to be taken into account when moving an individual particle via the integration scheme. The first method is mainly applied to molecules which are modeled as rigid body, i.e. the motion may easily be described as the translation of the center of mass and a rotation around its principle axis of inertia. In the case of large molecules, where not all bond lengths and angles are fixed and which exhibit internal degrees of freedom such as side chain motions or rotation of atomic groups, constraint methods have to be applied. In the following the motion of rigid bodies and the constraint dynamics will be described.

3.4.1 Rigid Body Motion

The natural way of describing a rigid body is to specify the coordinates and the moment of the center of mass as well as the orientation with respect to a space fixed coordinate system and the angular velocity around the principle molecular axis. The translational motion is thereby described by the total force acting on the molecule and the integration schemes, described earlier may be applied. The rotational motion requires a description of the orientation of the moelcule and a calculation of the torque. As a first choice for the orientational description one could use the Euler angles $(\varphi, \vartheta, \psi)$ to build up the rotation matrix. However, a numerical problem appears with solving the equations of motions

$$\frac{\partial \varphi}{\partial t} = -\omega_x \frac{\sin\varphi\cos\vartheta}{\sin\vartheta} + \omega_y \frac{\cos\varphi\cos\vartheta}{\sin\vartheta} + \omega_z \tag{43}$$

$$\frac{\partial \vartheta}{\partial t} = \omega_x \cos\varphi + \omega_y \sin\varphi \tag{44}$$

$$\frac{\partial \psi}{\partial t} = \omega_x \frac{\sin\varphi}{\sin\vartheta} - \omega_y \frac{\cos\varphi}{\sin\vartheta} \tag{45}$$

It is obvious that for the case, when ϑ is close or equal zero, the terms in $\partial_t\varphi$ and $\partial_t\psi$ diverge and lead to numerical instabilities. Since the orientation where $\vartheta = 0$ is physically not a special case but only related to the special convention of a chosen coordinate system, one can in principle switch the description into another coordinate system when ϑ approaches zero[68]. This is, however, not very efficient since variables have to be calculated and stored in different coordinate systems at the same time.

An elegant method which avoids these problems is the orientational description in terms of quaternions, $\mathbf{q} = (q_1, \ldots, q_4)$, originally introduced by Hamilton in order to extend the complex numbers[69]. They are defined by algebraic relations and have the property $\sum_i q_i^2 = 1$. Also, they can be expressed in terms of Eulerangles

$$q_1 = \cos\frac{\vartheta}{2}\cos\frac{\varphi+\psi}{2} \tag{46}$$

$$q_2 = \sin\frac{\vartheta}{2}\cos\frac{\varphi-\psi}{2} \tag{47}$$

$$q_3 = \sin\frac{\vartheta}{2}\sin\frac{\varphi-\psi}{2} \tag{48}$$

$$q_4 = \cos\frac{\vartheta}{2}\sin\frac{\varphi+\psi}{2} \tag{49}$$

so that they can completely describe the orientation of a fixed body in space. The equations of motion for $\mathbf{q}$ are given by

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{1}{2}\mathbf{Q}\boldsymbol{\omega}^b \tag{50}$$

where

$$\mathbf{Q} = \begin{pmatrix} q_1 & -q_2 & -q_3 & -q_4 \\ q_2 & q_1 & -q_4 & q_3 \\ q_3 & q_4 & q_1 & -q_2 \\ q_4 & -q_3 & q_2 & q_1 \end{pmatrix} \tag{51}$$

and

$$(\boldsymbol{\omega}^b)^T = (0, \omega_x^b, \omega_y^b, \omega_z^b) \tag{52}$$

where the superscript b denotes that the angular velocities are evaluated in the body fixed frame. From Eq.50 it becomes obvious that the divergence problems have disappeared. The transformation between a fixed body frame ($\hat{\mathbf{x}}$) and and a space fixed description ($\mathbf{x}$) is then provided by $\hat{\mathbf{x}} = \mathbf{R}\mathbf{x}$, where the rotation matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \begin{pmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2q_3 + q_1q_4) & 2(q_2q_4 - q_1q_3) \\ 2(q_2q_3 - q_1q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3q_4 + q_1q_2) \\ 2(q_2q_4 + q_1q_3) & 2(q_3q_4 - q_1q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{pmatrix} \tag{53}$$

The integration scheme for the rotational part is more involved than it is for its translational counterpart. Since the calculation of angular momenta is most easily done in the fixed body (molecular) frame, where the moment of inertia tensor is diagonal, a transformation from the space fixed (laboratory) frame to the molecular frame is required.

In the following a leap-frog *like* scheme is described, which uses information of half-step results[70]. In a first step the torque on the molecule is calculated by $\mathbf{T}_i = \sum_\alpha \mathbf{d}_\alpha \times \mathbf{F}_\alpha$, where α denotes the molecular sites and $\mathbf{d}_\alpha = \mathbf{R}(\mathbf{Q})\hat{\mathbf{d}}_\alpha$ is the vector pointing from the center of mass of the molecule to site α, written in the laboratory frame. Having the torque, the angular momentum $\mathbf{j}$ and the angular velocity $\boldsymbol{\omega}$ in the molecular frame can be obtained via

$$\mathbf{j}_i(t) = \mathbf{j}_i(t - \delta t/2) + \frac{1}{2}\mathbf{T}_i \delta t \tag{54}$$

$$\hat{\boldsymbol{\omega}}_i = \hat{\mathbf{I}}^{-1}\mathbf{R}^T(\mathbf{Q_i})\mathbf{j}_i(t) \tag{55}$$

where $\hat{\mathbf{I}}^{-1}$ is the inverse of the diagonal moment of inertia tensor. A similar step is performed to calculate the quaternions at time $t + \delta t/2$

$$\mathbf{q}_i(t + \delta t/2) = \mathbf{q}_i(t) + \frac{\delta t}{2}\mathbf{Q}(\mathbf{q}_i(t))\hat{\boldsymbol{\omega}}_i \tag{56}$$

In the next step the angular momenta are propagated from where the angular velocity can be obtained, in order to complete the intergation step

$$\hat{\mathbf{j}}_i(t + \delta t/2) = \mathbf{R}^T(\mathbf{Q_i}(t + \delta t/2))(\mathbf{j}_i(t - \delta t/2) + \delta t \mathbf{T}_i(t)) \tag{57}$$

$$\hat{\boldsymbol{\omega}}_i(t + \delta t/2) = \hat{\mathbf{I}}^{-1}\hat{\mathbf{j}}_i(t + \delta t/2) \tag{58}$$

$$\mathbf{q}_i(t + \delta t) = \mathbf{q}_i(t) + \frac{\delta t}{2}\mathbf{Q_i}(t + \delta t/2)\hat{\boldsymbol{\omega}}_i(t + \delta t/2) \tag{59}$$

Examples for rigid-body algorithms, based on a splitting method, which conserve the symplectic structure and are time reversible can be found in Refs.[71,72]. The method presented in Ref.[71] is implemented in the downloadable research program ORIENT[73].

3.4.2 Constrained Motion

Describing large molecules as rigid bodies is often a poor approximation. Especially, if conformational changes of a molecule are expected, it is not possible to freeze all internal degrees of freedom. However, if one is not interested in the high frequency intramolecular vibrational motions, which require a rather small timestep of integration (and therefore will slow down a simulation considerably), one can fix the bond lengths between neighbored sites and allow for bending and dihedral motions. The control of the bond length can be achieved by introducing Lagrangian multipliers. If the constraints are formulated as holonomic constraints the equations of motion are modified according to

$$\frac{\partial \mathbf{q}_i}{\partial t} = \frac{\mathbf{p}_i}{m_i} \tag{60}$$

$$\frac{\partial \mathbf{p}_i}{\partial t} = -\frac{\partial U(\mathbf{q})}{\partial \mathbf{q}_i} + \mathbf{g}'(\mathbf{q})\lambda \tag{61}$$

$$g(\mathbf{q}) = 0 \tag{62}$$

The *advantage* of this method is that the atoms of the molecule may be treated individually with a simple integrator scheme. No transformation from a laboratory to molecular frame has to be performed. Also the integration of angular degrees of freedom and the calculation of torques is not required. Solving for the Lagrange multiplier, generally leads to a diagonalization of an $P \times P$ matrix, where P is the number of constraints. However, since constraints are applied most often only to nearby atoms the matrix is sparse and and fast methods may be applied[74]. An alternative to this direct method is to fulfill the constraints in an iterative way one by one up to a given precision. First an unconstrained motion of the atoms of a molecule is performed which leads in general to positions which do not satisfy the constraints. The correction of the positions is then achieved via

$$\mathbf{r}_i(t+\delta t) \rightarrow \mathbf{r}_i(t+\delta t) + \frac{\delta t^2}{2m_i} \sum_\gamma \mathbf{F}_\gamma^c \tag{63}$$

where γ runs over all constraints and the constraint forces are given by

$$\mathbf{F}^c = \frac{\mu}{2\delta t^2} \frac{(\mathbf{d}_0^2 - \mathbf{d}^2)}{|\mathbf{d}_0 \mathbf{d}|} \mathbf{d}_0 \tag{64}$$

where $\mathbf{d}_0$ is the constrained bond vector at the start of the integration step and $\mathbf{d}$ is the bond vector after the unconstrained integration step, $\mu = m_i m_j/(m_i + m_j)$ is the reduced mass of the atom pair i, j. For a molecule with multiple constraints, Eq.63 is a first order correction , which can be applied in an iterative way up to a given precision, $|\mathbf{d}_0 - \mathbf{d}|/|\mathbf{d}_0| < 10^{-k}$, where k is the desired precision which is often chosen as $k \geq 4$ in order to conserve energy. This method is used in the algorithms SHAKE[75], invented by Ryckaert et al. and in RATTLE[76] invented by Andersen. The latter one was proven to be

symplectic and time reversible[59]. Refinements of the SHAKE algorithm were proposed in Ref.[77]. Schemes, where rotations of linked bodies are taken into account by quaternion methods, were proposed in Refs.[78,79].

4 Simulating in Different Ensembles

In MD simulations it is possible to realize different types of thermodynamic ensembles which are characterized by the control of certain thermodynamic quantities. If one knows how to calculate a thermodynamic quantity, e.g. the temperature or pressure, it is often possible to formulate an algorithm which fixes this property to a desired value. However, it is not always clear whether this algorithm decribes the properties of a given thermodynamic ensemble.

One can distinguish four different types of control mechanisms:

Differential control : the thermodynamic quantity is fixed to the prescribed value and no fluctuations around an average value occur.

Proportional control : the variables, coupled to the thermodynamic property f, are corrected in each interation step through a coupling constant towards the prescribed value of f. The coupling constant determines the strength of the fluctuations around $\langle f \rangle$.

Integral control : the system's Hamiltonian is extended and variables are introduced which represent the effect of an external system which fix the state to the desired ensemble. The time evolution of these variables is determined by the equations of motion derived from the extended Hamiltonian.

Stochastic control : the values of the variables coupled to the thermodynamic property f are propagated according to modified equations of motion, where certain degrees of freedom are additionally modified stochastically in order to give the desired mean value of f.

In the following, different statistical ensembles are presented and all methods will be discussed via examples.

4.1 The Microcanonical Ensemble

The microcanonical ensemble (NVE) may be considered as the *natural* ensemble for molecular dynamics simulations (as it is the canonical ensemble (NVT) for Monte Carlo simulations). If no time dependent external forces are considered, the system's Hamiltonian is constant, implying that the system's dynamics evolves on a constant energy surface. The corresponding probability density in phase space is therefore given by

$$\rho(\mathbf{q}, \mathbf{p}) = \delta(\mathcal{H}(\mathbf{q}, \mathbf{p}) - E) \tag{65}$$

In a computer simulation this theoretical condition is generally violated, due to limited accuracy in integrating the equations of motion and due to roundoff errors resulting from a limited precision of number representation. In Ref.[80] a numerical experiment was performed showing that tiny perturbations of the initial positions of a trjectory are doubled

about every picosecond. This would mean even for double precision arithmetic that after about $50\ ps$ roundoff errors will be dominant[59]. This is, however, often not a too serious restriction, since most time correlation functions drop to zero on a much shorter time scale. Only for the case where long time correlations are expected one does have to be very careful in generating trajectories.

4.2 The Canonical Ensemble

The simplest extension to the microcanonical ensemble is the canonical one (N,V,T), where the number of particles, the volume and the temperature are fixed to prescribed values. The temperature T is, in contrast to N and V, an intensive parameter. The extensive counterpart would be the kinetic energy of the system. In the following, different control mechanisms, introduced in Sec. 4 are described.

4.2.1 The Differential Thermostat

Different methods were proposed to fix the temperature to a fixed value during a simulation without allowing fluctuations of T. The first method was introduced by Woodcock[81], where the velocities were scaled according to $\mathbf{p}_i \rightarrow \sqrt{T_0/T}\mathbf{p}_i$, where T_0 is the reference temperature and T the actual temperature, calculated from the velocity of the particles. This method leads to discontinuities in the momentum part of the phase space trajectory due to the rescaling procedure.

An extension of this method implies a constraint of the equations of motion to keep the temperature fixed[82–84]. The principle of least constraint by Gauss states that a force added to restrict a particle motion on a constraint hypersurface should be normal to the surface in a realistic dynamics. From this principle the equations of motion are derived

$$\frac{\partial \mathbf{q}_i}{\partial t} = \mathbf{p}_i \tag{66}$$

$$\frac{\partial \mathbf{p}_i}{\partial t} = -\frac{\partial V}{\partial \mathbf{q}_i} - \zeta \mathbf{p}_i \tag{67}$$

where ζ is a constraint force term, calculated as

$$\zeta = -\frac{\sum_{i=1}^{N} \frac{\mathbf{p}_i}{m_i}\frac{\partial V}{\partial \mathbf{q}_i}}{\sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{m_i}} \tag{68}$$

Since the principle of least constraint by Gauss is used, this algorithm is also called *Gaussian thermostat*. It may be shown for this method that the configurational part of the phase space density is of canonical form, i.e.

$$\rho(\mathbf{q}, \mathbf{p}) = \delta(T - T_0)\, e^{-\beta U(\mathbf{q})} \tag{69}$$

4.2.2 The Proportional Thermostat

The proportional thermostat tries to correct deviations of the actual temperature T form the prescribed one T_0 by multiplying the velocities by a certain factor λ in order to move the system dynamics towards one corresponding to T_0. The difference with respect to the differential control is that the method allows for fluctuations of the temperature, thereby not fixing it to a constant value. In each integration step it is insured that the T is corrected to a value more close to T_0. A thermostat of this type was proposed by Berendsen et al.[85,86] who introduced *weak coupling methods to an external bath*. The weak coupling thermostat was motivated by the minimization of local disturbances of a stochastic thermostat while keeping the global effects unchanged. This leads to a modification of the momenta $\mathbf{p}_i \rightarrow \lambda \mathbf{p}_i$, where

$$\lambda = \left[1 + \frac{\delta t}{\tau_T}\left(\frac{T_0}{T} - 1\right)\right]^{\frac{1}{2}} \tag{70}$$

The constant τ_T, appearing in Eq.70, is a so called coupling time constant which determines the time scale on which the desired temperature is reached. It is easy to show that the proportional thermostat conserves a Maxwell distribution. However, the method cannot be mapped onto a specific thermodynamic ensemble. In Ref.[87] the phase space distribution could be shown to be

$$\rho(\mathbf{q}, \mathbf{p}) = f(\mathbf{p})\, e^{-\beta\left(U(\mathbf{q}) - \alpha\beta\delta U(\mathbf{q})^2/3N\right)} \tag{71}$$

where $\alpha \simeq (1 - \delta E/\delta U)$ and $\delta U,\ \delta E$ are the mean fluctuations of the potential and total energy. $f(\mathbf{p})$ is in general an unknown function of the momenta, so that the full density cannot be determined. For $\alpha = 0$, which corresponds in Eq.70 to $\tau_T = \delta t$, the fluctuations in the kinetic energy vanish and Eq.71 reduces to Eq.69, i.e. it represents the canonical distribution. The other extreme of $\tau_T \rightarrow \infty$ corresponds to an isolated system and the energy should be conserved, i.e. $\delta E = \delta K + \delta U = 0$ and $\alpha = 1$. In this case, Eq.71 corresponds to the microcanonical distribution[87]. Eq.71 may therefore be understood as an interpolation between the canonical and the microcanonical ensemble.

4.2.3 The Stochastic Thermostat

In the case of a stochastic thermostat, all or a subset of the degrees of freedom of the system are subject to collisions with *virtual* particles. This method can be motivated by a Langevin stochastic differential equation which escribes the motion of a particle due to the thermal agitation of a heat bath

$$\frac{\partial \mathbf{p}_i}{\partial t} = -\frac{\partial U}{\partial \mathbf{q}_i} - \gamma \mathbf{p}_i + \mathbf{F}^+ \tag{72}$$

where γ is a friction constant and $\mathbf{F}^+$ a Gaussian random force. The amplitude of $\mathbf{F}^+$ is determined by the second fluctuation dissipation theorem

$$\langle \mathbf{F}_i^+(t_1)\mathbf{F}_j^+(t_2)\rangle = 2\gamma k_B T \delta_{ij}\delta(t_1 - t_2) \tag{73}$$

A larger value for γ will increase thermal fluctuations, while $\gamma = 0$ reduces to the microcanonicle ensemble. This method was applied to molecular dynamics in Ref.[88]. A more

direct way was followed in Refs.[89,90] where particles collide occasionally with virtual paricles from a Maxwell distribution corresponding to a temperature T_0 and after collisions loose their memory completely, i.e. the motion is totally randomized and the momenta become discontinuous. In order not to disturb the phase space trajectory too much, the collision frequency has to be chosen not too high. Since a large collision frequency will lead to a strong loss of the particle's memory, it will lead to a fast decay of dynamic correlation functions[91]. The characteristic decay time of correlation functions should therefore be a measure for the collision time. It was proved for the stochastic thermostat[89] that it leads to a canonical distribution function.

A slightly different method which is able to control the coupling to an external bath was suggested in Refs.[92,93]. In this approach the memory of the particle is not completely destroyed but the new momenta are chosen to be

$$\mathbf{p}_{i,n} = \sqrt{1-\alpha^2}\,\mathbf{p}_{i,o} + \alpha\,\mathbf{p}_r \tag{74}$$

where $\mathbf{p}_r$ is chosen a momentum, drawn from a Maxwell distribution corresponding to T_0. Similar to the proportional thermostat, the parameter α may be tuned to give distributions ranging from the microcanonical to the canonical ensemble.

4.2.4 The Integral Thermostat

The integral method is also often called *extended system method* as it introduces additional degrees of freedom into the system's Hamiltonian for which equations of motion can be derived. They are integrated in line with the equations for the spatial coordinates and momenta. The idea of the method invented by Nosé[94,95], is to reduce the effect of an external system acting as heat reservoir to kepp the temperature of the system constant, to one additional degree of freedom. The thermal interactions between a heat reservoir and the system result in a change of the kinetic energy, i.e. the velocity of the particles in the system. Formally it may therefore be expressed a scaling of the velocities. Nosé introduced two sets of variables: real and so called virtual ones. The virtual variables are consistently direved from a Sundman transformation[96] $d\tau/dt = s$, where τ is a virtual time and s is a resulting scaling factor, which is treated as dynamical variable. The transformation from virtual to real variables is then performed as

$$\mathbf{p}_i = \boldsymbol{\pi}_i s \qquad , \qquad \mathbf{q}_i = \boldsymbol{\rho}_i \tag{75}$$

The introduction of the *effective mass*, M_s, connects also a momentum to the additional degree of freedom, π_s. The resulting Hamiltonian, expressed in virtual coordinates reads

$$\mathcal{H}^* = \sum_{i=1}^{N} \frac{\boldsymbol{\pi}_i^2}{2m_i s^2} + U(\boldsymbol{\rho}) + \frac{\pi_s^2}{2M_s} + g k_B T \ln s \tag{76}$$

where $g = 3N + 1$ is the number of degrees of freedom (system of N free particles). The Hamiltonian in Eq.76 was shown to lead to a probability density in phase space, corresponding to the canonical ensemble.

The equations of motion drawn from this Hamiltonian are

$$\frac{\partial \boldsymbol{\rho}_i}{\partial \tau} = \frac{\boldsymbol{\pi}_i}{s^2} \tag{77}$$

$$\frac{\partial \boldsymbol{\pi}_i}{\partial \tau} = -\frac{\partial U(\boldsymbol{\rho})}{\partial \boldsymbol{\rho}_i} \tag{78}$$

$$\frac{\partial s}{\partial \tau} = \frac{\pi_s}{M_s} \tag{79}$$

$$\frac{\partial \pi_s}{\partial \tau} = \frac{1}{s^3}\sum_{i=1}^{N}\frac{\boldsymbol{\pi}_i^2}{m_i} - \frac{g k_B T}{s} \tag{80}$$

If one transforms these equations back into real variables, it is found[97] that they can be simplified by introducing the new variable $\zeta = \partial s/\partial t = s p_s/M_s$ (p_s is *real* momentum connected to the heat bath)

$$\frac{\partial \mathbf{q}_i}{\partial t} = \frac{\mathbf{p}_i}{m_i} \tag{81}$$

$$\frac{\partial \mathbf{p}_i}{\partial t} = -\frac{\partial U(\mathbf{q})}{\partial \mathbf{q}_i} - \zeta \mathbf{p}_i \tag{82}$$

$$\frac{\partial \ln s}{\partial t} = \zeta \tag{83}$$

$$\frac{\partial \zeta}{\partial t} = \frac{1}{M_s}\left(\sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{m_i} - g k_B T\right) \tag{84}$$

These equations describe the so called Nosé-Hoover thermostat.

4.3 The Constant-Pressure Constant-Enthalpy Ensemble

In order to control the pressure in an MD simulation cell, it is necessary to allow for volume variations. A simple picture for a constant pressure system is a box the walls of which are coupled to a piston which controls the pressure. In contrast to the case where the temperature is controled, no coupling to the dynamcics of the particles (timescales) is performed but the length scales of the system will be modified. In the following, different algorithms are described for a constant pressure ensemble. The conserved quantity will not be the system's energy, since there will be an energy transfer to or from the *external* system (piston etc.), but the enthalpy H will be constant. In line with the constant temperature methods there are also differential, proportional, integral and stochastic methods to achieve a constant pressure situation in simulations. The differential method, however, is not dicussed here, since there are problems with that method related to the *correct initial* pressure[98,99]. A scheme for the calculation of the pressure in MD simulations for various model systems is given in the appendix.

4.3.1 The Proportional Barostat

The proportional thermostat in Sec. 4.2.2 was introduced as an extension for the equation of the momentum, since it couples to the kinetics of the particles. Since the barostat acts on a volume change, which may be expressed in a scaling of particles' positions, a phenomenological extension for the equation of motion of the coordinates may be formulated[85]

$$\frac{\partial \mathbf{q}_i}{\partial t} = \frac{\mathbf{p}_i}{m_i} + \alpha \mathbf{q}_i \tag{85}$$

while a change in volume is postulated as

$$\dot{V} = 3\alpha V \tag{86}$$

A change in pressure is related to the isothermal compressibility κ_T

$$\dot{P} = -\frac{1}{\kappa_T V}\frac{\partial V}{\partial t} = -\frac{3\alpha}{\kappa_T} \tag{87}$$

which is approximated as

$$\frac{(P_0 - P)}{\tau_P} = -\frac{3\alpha}{\kappa_T} \tag{88}$$

and therefore Eq.85 can be written as

$$\frac{\partial \mathbf{q}_i}{\partial t} = \frac{\mathbf{p}_i}{m_i} - \frac{\kappa_T}{3\tau_P}(P_0 - P) \tag{89}$$

which corresponds to a scaling of the boxlength and coordinates $\mathbf{q} \rightarrow s\mathbf{q}$ and $L \rightarrow sL$, where

$$s = 1 - \frac{\kappa_T \delta t}{3\tau_P}(P_0 - P) \tag{90}$$

The time constant τ_P was introduced into Eq.88 as a characteristic timescale on which the system pressure will approach the desired pressure P_0. It also controls the strength of the coupling to the barostat and therefore the strength of the volume/pressure fluctuations. If the isothermal compressiblity, κ_T, is not known for the system, the constant $\tau_P' = \tau_P/\kappa_T$ may be considered as a phenomenological coupling time which can be adjusted to the system under consideration. As for the proportional thermostat, a drawback for this method is that the statistical ensemble is not known. In analog to the thermostat, it may be assumed to *interpolate* between the microcanonical and the constant-pressure/constant-enthalpy ensemble, depending on the coupling constant τ_P.

4.3.2 The Integral Barostat

In line with the integral thermostat one can introduce a new degree freedom into the systems Hamiltonian which controls volume fluctuations. This method was first proposed by Andersen[89]. The idea is to include the volume as an additional degree of freedom and to wrte the Hamiltonian in a scaled form, where lengths are expressed in units of the boxlength $L = V^{1/3}$, i.e. $\mathbf{q}_i = L\ \boldsymbol{\rho}_i$ and $\mathbf{p}_i = L\ \boldsymbol{\pi}_i$. Since L is also a dynamical

quantity, the momentum is not related to the simple time derivative of the coordinates but $\partial_t \mathbf{q}_i = L\,\partial_t \boldsymbol{\rho}_i + \boldsymbol{\rho}_i\,\partial_t L$. The extended system Hamiltonian is then written as

$$\mathcal{H}^* = \frac{1}{V^{2/3}} \sum_{i=1}^{N} \frac{\boldsymbol{\pi}_i}{2m_i} + U(V^{1/3}\boldsymbol{\rho}) + P_{ex}V + \frac{\boldsymbol{\pi}_V^2}{2M_V} \tag{91}$$

where P_{ex} is the prescribed external pressure and $\boldsymbol{\pi}_V$ and M_V are a momentum and a mass associated with the fluctuations of the volume.

The equations of motion which are derived from this Hamiltonian are

$$\frac{\partial \boldsymbol{\rho}_i}{\partial t} = \frac{1}{V^{2/3}} \frac{\boldsymbol{\pi}_i}{m_i} \tag{92}$$

$$\frac{\partial \boldsymbol{\pi}_i}{\partial t} = \frac{\partial U(V^{1/3}\boldsymbol{\rho})}{\partial \boldsymbol{\rho}_i} \tag{93}$$

$$\frac{\partial V}{\partial t} = \frac{\boldsymbol{\pi}_V}{M_V} \tag{94}$$

$$\frac{\partial \boldsymbol{\pi}_V}{\partial t} = \frac{1}{3V} \left(\frac{1}{V^{2/3}} \sum_{i=1}^{N} \frac{\boldsymbol{\pi}_i}{m_i} - V^{1/3} \boldsymbol{\rho}_i \frac{\partial U(\mathbf{q})}{\partial \mathbf{q}_i} \right) \tag{95}$$

A transformation to real variables then gives

$$\frac{\partial \mathbf{q}_i}{\partial t} = \frac{\mathbf{p}_i}{m_i} + \frac{1}{3V} \frac{\partial V}{\partial t} \mathbf{q}_i \tag{96}$$

$$\frac{\partial \mathbf{p}_i}{\partial t} = -\frac{\partial U(\mathbf{q})}{\partial \mathbf{q}_i} - \frac{1}{3V} \frac{\partial V}{\partial t} \mathbf{p}_i \tag{97}$$

$$\frac{\partial V}{\partial t} = \frac{\boldsymbol{p}_V}{M_V} \tag{98}$$

$$\frac{\partial \boldsymbol{p}_V}{\partial t} = \underbrace{\frac{1}{3V} \left(\sum_{i=1}^{N} \frac{\mathbf{p}_i}{m_i} - \mathbf{q}_i \frac{\partial U(\mathbf{q})}{\partial \mathbf{q}_i} \right)}_{P} - P_{ex} \tag{99}$$

In the last equation the term in brackets corresponds to the pressure, calculated from the virial theorem (cf. Appendix A). The associated volume force, introducing fluctuations of the box volume is therefore controlled by the internal pressure, originating from the particle dynamics and the external pressure, P_{ex}.

5 Parallel Molecular Dynamics

With the advent of massively parallel computers, where thousands of processors may work on a single task, it has become possible to increase the size of the numerical problems considerably. As has been already mentioned in Sec.1 it is in principle possible to treat

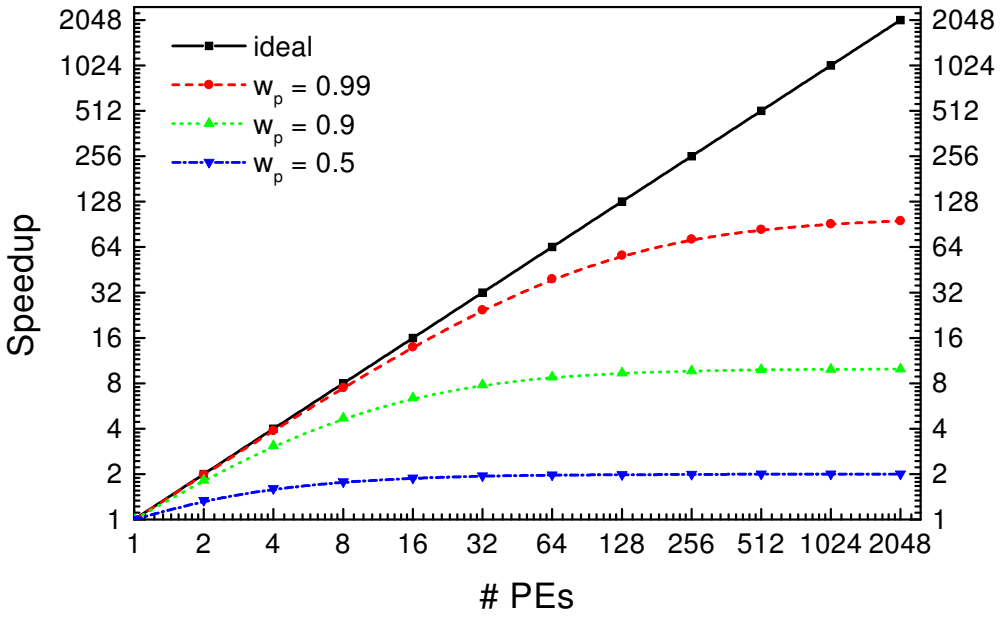


Figure 5. The ideal speedup for parallel applications with 50%, 90%, 99% and 100% (ideal) parallel work as a function of the number of processors.

multi-billion particle systems. However, the whole success of parallel computing strongly depends both on the underlying problem to be solved and the optimization of the computer program. The former point is related to a principle problem which is manifested in the so called Amdahl's law[100]. If a problem has inherently certain parts which can be solved only in serial, this will give an upper limit for the parallelization which is possible. The speedup σ, which is a measure for the gain of using multiple processors with respect to a single one, is therefore bound

$$\sigma = \frac{N_p}{w_p + N_p w_s}. \tag{100}$$

Here, N_p is the number of processors, w_p and w_s is the amount of work, which can be executed in parallel and in serial, i.e. $w_p + w_s = 1$. From Eq.100 it is obvious that the maximum efficiency is obtained when the problem is completely parallelizable, i.e. $w_p = 1$ which gives an N_p times faster execution of the program. In the other extreme, when $w_s = 1$ there is no gain in program execution at all and $\sigma = 1$, independent of N_p. In Fig.5 this limitation is illustrated for several cases, where the relative amount for the serial work was modified. If the parallel work is 50%, the maximum speedup is bound to $\sigma = 2$. If one aims to execute a program on a real massively parallel computer with hundreds or thousands of processors, the problem at hand must be inherently parallel for 99.99...%. Therefore, not only big parallel computers garuantee a fast execution of programs, but the problem itself has to be chosen properly.

Concerning MD programs there are only a few parts which have to be analysed for parallelization. As was shown, an MD program consists essentially of the force routine, which

Machine	CPU	Network	Latency	Bandwidth
CRAY T3E-1200 (ZAM Jülich)	DEC 21164 (600 MHz)	CRAY T3E interconnect	$\approx 8\mu s$ ($2\mu s$)	$\approx$ 350 MB/s
ZAMpano (ZAM Jülich)	Intel Pentium III Xeon (550 MHz)	Myrinet	$\approx 80\mu s$ ($15\mu s$)	$\approx$ 65 MB/s
MPCB (CNRS Orléans)	Intel Pentium III (550 MHz)	Fast Eathernet	$\approx 470\mu s$	$\approx$ 10 MB/s

Table 1. Interprocessor communication networks for a massively parallel machine (CRAY T3E-1200) with 512 processors and two Linux based PC clusters with 32 (Zampano) and 40 (MPCB) processors.

costs usually more than 90% of the execution time. If one uses neighbor lists, these may be also rather expensive while reducing the time for the force evaluation. Other important tasks are the integration of motion, the parameter setup at the beginning of the simulation and the file input/output (I/O). In the next chapter it will be shown how to parallelize the force routine. The integrator may be naturally parallelized, since the loop over N particles may be subdivided and performed on different processors. The parameter setup has either to be done in serial so that every processor has information about relevant system parameters, or it may be done in parallel and information is distributed from every processor via a broadcast. The file I/O is a more complicated problem. The message passing interface MPI I does not offer a parallel I/O operation. In this case, if every node writes some information to the same file there is, depending on the configuration of the system, often only one node for I/O, to which internally the data are sent from the other nodes. The same applies for reading data. Since on this node the data from/for the nodes are written/read sequentially, this is a serial process which limits the speedup of the execution. The new MPI II standard offers parallel read/write operations, which lead to a considerable efficiency gain with respect to MPI. However, the efficiency obtained depend strongly on the implementation on different architectures.

Another serious point is the implementation into the computer code. A problem which is inherently 100% parallel will not be solved with maximum speed if the program is not 100% mapped onto this problem. Implementation details for parallel algorithms will be discussed in the following sections. Independent of the implementation of the code, Eq.100 gives only an upper theoretical limit which will only be reached in very rare cases. For most problems it is necessary to communicate data from one processor to another or even to all other processors in order to take into account data dependencies. This implies an overhead which depends on the latency and the bandwidth of the interprocessor network. This strongly depends on the hardware implementation, as is shown in Table 1.

It is shown that the latency time (time which is used to initialize a communication) differs by a factor of about 50, while bandwidths differ by a factor of about 35. The effect of the data exchange will be included into Amdahl's law later on and more realistic speed up curves will be obtained.

5.1 Particle Decomposition

In order to share the work between N_p processors one can distribute N particles in the beginning of the simulation homogenously onto the processors. If a particle is assigned

permanently to a certain processor, the method is called particle decomposition (PD). If particles are distributed in the beginning of the simulation according to their spatial arrangement, topologically neighbored processors will contain neighbored particles in space. This remains almost true if the system is very viscous or it is a solid. However, in the case of liquids or gases, the particles will mix after a short time due to diffusive motions and neighbored processors may store data of particles which are spatially far apart from each other and *vice versa*. In order to calculate the interparticle forces, a global communiaction between processors is necessary.

5.1.1 Replicated Data Algorithm

The simplest way to parallelize a serial program or to write a new parallel code is the so called replicated data algorithm. Every node stores the coordinates of all N particles in the system. However, in contrast to a serial program, each processor computes the forces only of a certain subset of particles N/N_p, the so called *local particles*, i.e. the number of loop iterations is N^2/N_p. Having computed the forces on the local particles, the integration step is performed and the positions and velocities are updated. The next step consists in broadcasting the new positions to the other $N_p - 1$ processors. This involves a global communication which can either be implemented in a loop over all processors or can be realized by optimized library operations, e.g. `MPI_AllToAll`, which sends a subarray to all other processors, where it is sorted into the whole array, and receives subarrays from all other processors which are sorted into the local whole array. A straightforward implementation of this algorithm thus requires $N_p - 1$ send/receive operations.

A more efficient way makes use of a tree-like communication pattern, which needs only $\log_2(N_p)$ send/receive operations. This algorithm was proposed by Fox et al.[101]. Assuming a linear processor topology with periodic boundary conditions this algorithm works as follows: in a first step all processors receive the particle coordinates from their neighbored processor to their right. Each processor element (PE) now stores the updated coordinates from 2 PEs. In the next step the updated coordinates are received from the second PE to the right, i.e. in this step the information of already four PEs is obtained. One proceeds in step n to receive the coordinate vector from the $2^{(n-1)}$th neighbor to the right and to send the vector from the local PE to the $2^{(n-1)}$th neighbor node to the left. For the case of 128 PEs the whole communication is finished after 7 steps instead of 127 steps in the case of an all-to-all communication scheme. Note, however, that the total amount of data which has to be sent is unaffected by the algorithm. Minimized is only the latency time of the send/receive operations. Furthermore, for the tree-like communication one has to introduce an temporary array which stores the incoming/outgoing data from/to neighbored PEs which have to sorted into the *global* coordinate vector.

The scheme described up to now does not take into account the principle of action and counteraction (Newton's 3rd law). Implemented in an optimal way, this may speed up a computation by nearly a factor of two. An easy way to account for this symmetry relation is to divide the matrix of particle interactions from PE p_i and PE p_j ($p_i \neq p_j$) into a checkerboard scheme. Now, interactions are calculated on p_i when the coordinate index of the particles obey the property $i > j$ and $i + j$ is an even number. On the other hand, interactions on p_j are taken into account when $i < j$ and $i + j$ is an odd number. In a next step the locally computed forces have to be sent to the $N_p - 1$ processors and Newton's

3rd law can be applied. This scheme implies two global communications with coordinates and forces. It has to checked which method is more preferential. If communication is of minor importance, i.e. in the case of fast network, the 2nd variant will be faster.

5.1.2 Distributed Data Algorithm - Systolic Loop

The replicated data algorithm is easy and fast to implement. However, the disadvantage is the storage of a large coordinate vector. This may become difficult when very large numbers of particles are to be simulated and/or the computer memory is small.

Distributed data algorithms are then favorable. One type of this strategy is the systolic loop algorithm. In this algorithm a local PE stores only the coordinates, velocities and forces of the local particles.

Forces on the local PE are calculated in the usual way in a $N_L(N_L-1)/2$-loop, where N_L is the number of local particles. The communication between the PEs are organized in the following way. The processor indices are extended periodically, i.e. PE$_i$=PE$_{i+P}$, where $i = 1, \ldots, P$ is the index of the processor. Coordinates are sent from PE$_j$ to PE$_i$, with $i < j$, where the forces between the particles are calculated explicitly. The coordinates are stored in a temporary array of length $3N_L$. Applying the principle of action/counteraction the calculated force vectors are to be sent from PE p_i to PE p_j. The total force evaluation on a tagged particle is completed after $(N_p-1)/2$ send/receive operations of the coordinates in the first step and $(N_p-1)/2$ send/receive operations of the forces in the second step.

The algorithm exhibits a special feature when N_p is an even number. In that case half of the processors do not work in the last loop over processors, since redundant information would be communicated[c]. This implies a nonlinear scaling of the algorithm with increasing number of PEs, e.g. from $P = 1$ to $P = 2$ the maximum speed-up is $4/3$. Only for large numbers of PEs, the effect is reduced and the speed-up approaches a linear behavior. This effect may be avoided if one refers to the procedure discussed before (cf. Sec.5.1.1). In the last step of the loop over processors the send operations of coordinates are $p_1 \rightarrow p_{N_p/2-1}, \ldots, p_{N_p/2} \rightarrow p_{N_p}$. In addition one communicates now coordinates also from $p_{N_p/2-1} \rightarrow p_1, \ldots, p_{N_p} \rightarrow p_{N_p/2}$. In the following the force matrix is subdivided and its elements are only calculated explicitly if the particle index pair $i > j$ and the sum $i + j$ is even or if $i < j$ and the sum $i + j$ is odd. In the next step the force vector elements are send from PE p_j to PE p_i so that Newton's third law may be applied.

5.1.3 An Intermediate Algorithm - Hypersystolic Loop

In the case of the systolic loop algorithm, coordinates and forces are sent from one PE to the next and are stored only temporarily. On the other hand, if these coordinates would be stored on each PE, the information to calculate all forces in the system would be distributed in less than $N_p - 1$ send operations. This is illustrated in Table 2 for the case of 6 and 8 processors. In the case of 6 PEs the cycle is finished after 2 communication steps. As is seen, the interactions $p_1 \rightarrow p_2$ and $p_1 \rightarrow p_4$ are calculated on PE 1, $p_1 \rightarrow p_5$ is calculated

[c]In the last communication step PE N_p sends data to PE $N_p/2$ where forces are calculated and send back to PE N_p using the principle of action/counteraction. If PE $N_p/2$ would also send its coordinates to PE N_p and would receive the forces back from the same PE this results in a double calculation of forces leading to a wrong result.

PE		1	2	3	4	5	6
Step	0	1	2	3	4	5	**6**
	1	**2**	3	4	**5**	6	1
	2	**4**	5	6	1	2	**3**

PE		1	2	3	4	5	6	7	8
Step	0	1	2	3	4	5	**6**	7	**8**
	1	**2**	3	4	5	6	**7**	8	1
	2	**4**	5	6	7	8	1	2	**3**
	3	**5**	6	7	8	1	2	3	4

Table 2. Hypersystolic matrices for the cases of 6 and 8 processors. Bold numbers indcate the location of coordinates, which are used to calculate interactions with particles on processor 1.

on PE 4 and $p_1 \rightarrow p_3$ and $p_1 \rightarrow p_6$ are calculated on PE 6 [d]. This algorithm of sending and receiving data is called hypersystolic loop[102,103] and the scheme of communicating processors (cf. Table 2) is called the hypersystolic matrix. As a further example the case of 8 processors is also shown in Table 2, which finishes after 3 communication steps. For large numbers of processors this scheme is very promising as a compromise between the memory intensive replicated data algorithm and the communication intensive systolic loop algorithm. Unfortunately, however, a general scheme is not yet known how to build up the hypersystolic matrix, which makes the method unsortable for an arbitrary number of processors.

5.2 Force Decomposition

Another parallelization strategy aims to distribute the $N \times N$ force matrix onto the N_p processors. Two different implementations will be explained in the following.

5.2.1 Data Replicated Algorithm

This algorithm is closely related to the one discussed in Sec. 5.1.1. Particles are distributed uniformly on the processors and each PE stores the whole information of the N particles. The algorithm aims to use Newton's 3rd law, i.e. the calculation of the force matrix is reduced to the upper triangular matrix $\mathbf{M}^u$, and to distribute the work homogenously, i.e. each PE should have the same number of force loop iterations. This is achieved in the following way[104]. Each PE is assigned a fixed number of rows of $\mathbf{M}^u$ having the same *area* A on each PE. The total number of force iterations on each PE is given by

$$A(L_k) = \left(N - \sum_{j=1}^{k-1} L_j - \frac{L_k + 1}{2} \right) \tag{101}$$

with $k \in [1, N_p]$ and L_k being the number of rows in $\mathbf{M}^u$ being assigned to PE p_k, i.e. $\sum_{k=1}^{N_p} L_k = N$. As is required for an equal workload, all areas $A(L_k)$ should be equal. This leads to

$$L_k = \frac{1}{2} \left(Q_k - \sqrt{Q_k^2 - \frac{4N(N-1)}{N_p}} \right) \tag{102}$$

[d] The notation $p_i \rightarrow p_j$ means thereby the interaction between particles on processor p_i with those on processor p_j.

where

$$Q_k = 2N - 1 - 2\sum_{j=1}^{k-1} L_j \tag{103}$$

This subdivision of forces guarantees equal lengths of force loops on each PE. In order to get the complete force on every particle, a global reduction of the force vector has to be done, which can, in analogy to Sec. 5.1.1, also be performed in a tree-like structure, requiring $\log_2(N_p)$ steps. In order to save a global reduction of positions, the propagation of positions and velocities from step n to $n+1$ is done for all particles on every PE.

5.2.2 Low-Communication Version

Another type of decomposition of the force matrix requires a quadratic number of processors $N_p = n^2$, which are arranged in a $\sqrt{N_p} \times \sqrt{N_p}$ matrix P_{ij}[105]. The particle distribution on the processors is performed according to the following rule (cf. Fig.6)

$$I_{ij} = \begin{cases} [\frac{N}{N_p}(j - \frac{1}{2}) + 1; \frac{N}{N_p} j] & : \quad i < j \\ [\frac{N}{N_p}(i-1) + 1; \frac{N}{N_p} i] & : \quad i = j \\ [\frac{N}{N_p}(i-1) + 1; \frac{N}{N_p}(i - \frac{1}{2})] & : \quad i > j \end{cases} \tag{104}$$

where I_{ij} denotes the interval of particle indices, stored on processor P_{ij}. Calculation of particle interactions is then performed on each local processor without exchange of particle coordinates with other PEs. In the following the symmetry of the transpose matrix is used and forces are exchanged between processor P_{ij} and P_{ji} $(i \neq j)$. Now, in every row i of the matrix is the whole information for the forces which is necessary for of the diagonal element P_{ii}. Consequently a reduction step is performed in the following to sum up the

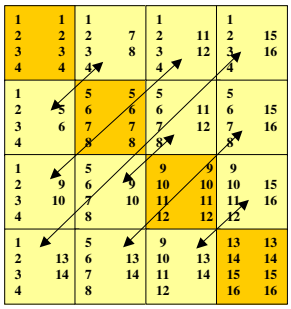


Figure 6. Communication pattern between 16 processors for the case of 16 particles (indicated numbers).

forces row wise on the diagonal elements of the matrix. In a next the particle coordinates and velocities are propagated only on the processors belonging to the diagonal elements of P_{ij}. In a last step, the updated positions are distributed according to Eq. 104. This scheme is less communication intensive as the algorithm discussed in Sec.5.2.1, since it

requires only the row and column wise replication of data ($2(\sqrt{(}N_p) - 1)$ operations), the transpose exchange (one communication operation for every *non-diagonal* PE) and the force reduction ($(\sqrt{(}N_p) - 1)$ operations). This reduction in communication is, however, achieved by a very much more complicated implementation and a less balanced workload, since not all processors have to calculate the same number of pair forces (cf. Fig.6).

5.3 Domain Decomposition

The principle of spatial decomposition methods is to assign geometrical domains to different processors. This implies that particles are no longer bound to a certain processor but will be transfered from one PE to another, according to their spatial position. This algorithm is especially designed for systems with short range interactions or to any other algorithm where a certain cut-off in space may be applied. Since neighbored processors contain all relevant data needed to compute forces on particles located on a given PE, this algorithm avoids the problem of global communications. Given that the range of interaction between particles is a cut-off radius of size R_c, the size, D of the domains is preferentially chosen to be $D > R_c$, so that only the $3^d - 1$ neighbored processors have to communicate data (d is the dimension of the problem). Whether this can be fulfilled depends on the interplay between size of the system and the numbers of processors. If a small system is treated with a lare number of processors, the domains will be small and $D < R_c$. In this case not only the next but also the second or even higher order neighbor PEs have to send their coordinates to a given PE. For simplicity we assume here $D > R_c$.

The algorithm then works as follows. Particles are distributed in the beginning of the simulation to a geometrical region. The domains are constructed to have a rather homogeneous distribution of particles on each processor, e.g. for homogeneous bulk liquids the domains can be chosen as equally sized cuboids which fill the simulation box. In order to calculate forces between particles on different processors, coordinates of the so called *boundary particles* (those which are located in the outer region of size $R_b \geq R_c$ of the domains) have to be exchanged. Two types of lists are constructed for this purpose. The one contains all particle indices, which have left the local domain and which have consequently to be transferred to the neighbored PE. The other one contains all particle indices, which lie in the outer region of size R_b of a domain. The first list is used to update the particles' *address*, i.e. all information like positions, velocities, forces etc. are sent to the neighbored PE and are erased in the old domain. The second list is used to send temporarily position coordinates which are only needed for the force computation. The calculation of forces then operates in two steps. First, the forces due to local particles are computed using Newton's 3rd law. In a next step, forces due to the boundary particles are calculated. The latter forces are thus calculated twice: on the local PE and the neighbored PE. This extra computation has the advantage that there is no communication step for forces. A more elaborate scheme has nevertheless been proposed which includes also Newton's 3rd law for the boundary particles and thus the communication of forces[106, 107]. Having finished the evaluation of forces, the new positions and velocities are evaluated only for local particles.

A naive method would require $3^d - 1$ send/receive operations. However, this may be reduced to $2\log_d(3^d - 1)$ operations with a similar tree-like method, as described in Sec.5.1.1. The method is described here for the case of $d = 2$. It may be generalized

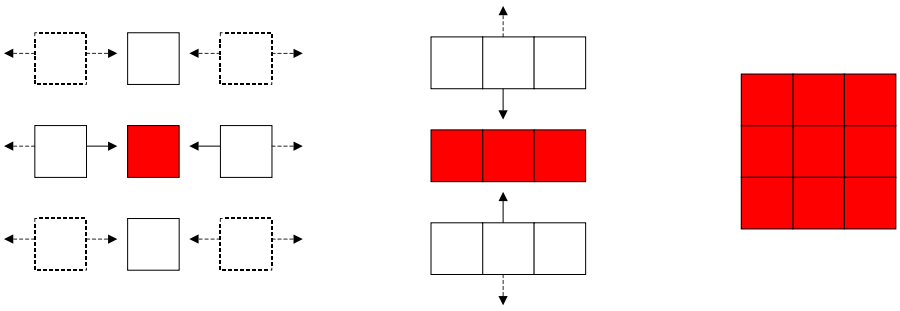


Figure 7. Communication pattern for the domain decomposition algorithm in 2 dimensions.

rather easily. The 4 processors, located directly at the edges of a given one are labeled as left/right and up/down. Then in a first step, information is sent/received to/from the left and the right PE, i.e. each processor now stores the coordinates of three PEs (including local information). The next step proceeds in sending/receiving the data to the up and down PEs. This step finishes already the whole communication process.

The updating process is not necessarily done in each time step. If the width of the boundary region is chosen as $R_b = R_c + \delta r$, it is possible to trigger the update automatically via the criterion $\max(|\mathbf{x}(t_0 + t) - \mathbf{x}(t_0)|) \leq \delta r$, which is the maximum change in distance of any particle in the system, measaured from the last update.

A special feature of this algorithm is the fact that it shows a theoretical superlinear speed-up if Verlet neighbor lists are used. The construction of the Verlet list requires $N'(N'-1)/2 + N'\delta N$ operations, where δN is the number of boundary particles and N' is the number of particles on a PE. If the number of PEs is increased as twice as large, there are $N'/2$ particles on each processor which therefore requires $N'/2(N'/2-1)/2 + N'/2\delta N$ operations. If $N' \gg \delta N$ and $N'^2 \gg N'$ one gets a speed-up factor of ≈ 4!

5.4 Performance Estimations

In order to estimate the performance of the different algorithms on a theoretical basis it is useful to extend the ideal Amdahl's law to a more realistic case. The ideal law only takes into account the degree of parallel work. From that point of view all parallel algorithms for a given problem should work in the same way. However the communication between the processors is also a limiting factor in parallel applications and so it is natural to extend Amdahl's law in the following way

$$\sigma = \frac{1}{w_p/N_p + w_s + c(N_p)} \tag{105}$$

where $c(N_p)$ is a function of the number of processors which will characterize the different parallel algorithms. The function will contain both communication work, which depends on the bandwidth of the network and the effect of the latency time, i.e. how fast the network responds to the communication instruction. The function $c(N_p)$ expresses the relative portion of communication with respect to computation. Therefore it will depend in general also on the number of particles which are simulated.

In the following an analysis for three different types of parallel algorithms is presented. It is always assumed that the work is strictly parallel, i.e. $w_p = 1$.

5.4.1 All-to-All Communication

An example for the all-to-all communication was given with the systolic loop algorithm. Every processor sends to all other $N_p - 1$ PEs. The amount of data which is sent reduces with increasing N_p, since only the data stored on every PE are sent. The work for communication may therefore be expressed as

$$c(N_p) = (N_p - 1)\left(\lambda + \frac{\chi}{N_p}\right) \tag{106}$$

In Fig.8a the speedup, calculated on basis of Eq.106, is shown for some values of λ and χ. It is found that the slope of the speedup may become negative! Since the latency time λ is in general a small number, the third term in the denominator is small compared with the first. Therefore the speedup curve for small N_p is nearly linear. However, since for larger N_p the first term in c_p grows nearly linear with N_p, it becomes dominant for the behavior of σ. An interesting observation is that the behavior for large N_p is mainly dominated by the latency time than the amount of data which is sent. With an increase of N_p the number of particles which is stored on each PE is reduced and consequently the ratio of computation to communication becomes smaller and smaller. For very few data on each PE, it resembles somebody who wants to give a telephone call but after every spoken word he has to dial again. One can imagine that even with modern telephones the time for finishing a message becomes longer and longer.

5.4.2 Tree-Like Communication

A parallel algorithm which uses a tree-like structure to distribute the data was discussed for the force-decomposition algorithm. In this case the number of message passing calls reduces to $\log_2(N_p)$ whereas the amount of data to be send to the next PE is doubled in each communication step. The expression for c_p may therefore be written as

$$c(N_p) = \log_2(N_p)\lambda + \sum_{n=1}^{\log_2(N_p)} \frac{2^{(n-1)}\chi}{N_p} \tag{107}$$

In Fig.8b the speedup behavior is shown. It is found that it is decreased with respect to the ideal behavior. Due to the slow logarithmic increase of the latency time part no decreasing behavior of σ is observed. For the unrealistic case where the communication part is neglected, the speedup is rather close to the ideal line. A considerable deviation is only found for $N_p > 500$. The result for this algorithm is therefore that it is communication limited rather than latency time limited.

5.4.3 Local Communication

The spatial decomposition algorithm is an example for the case of local communication. As was described in Sec.5.3, only six communication steps are required to distribute the

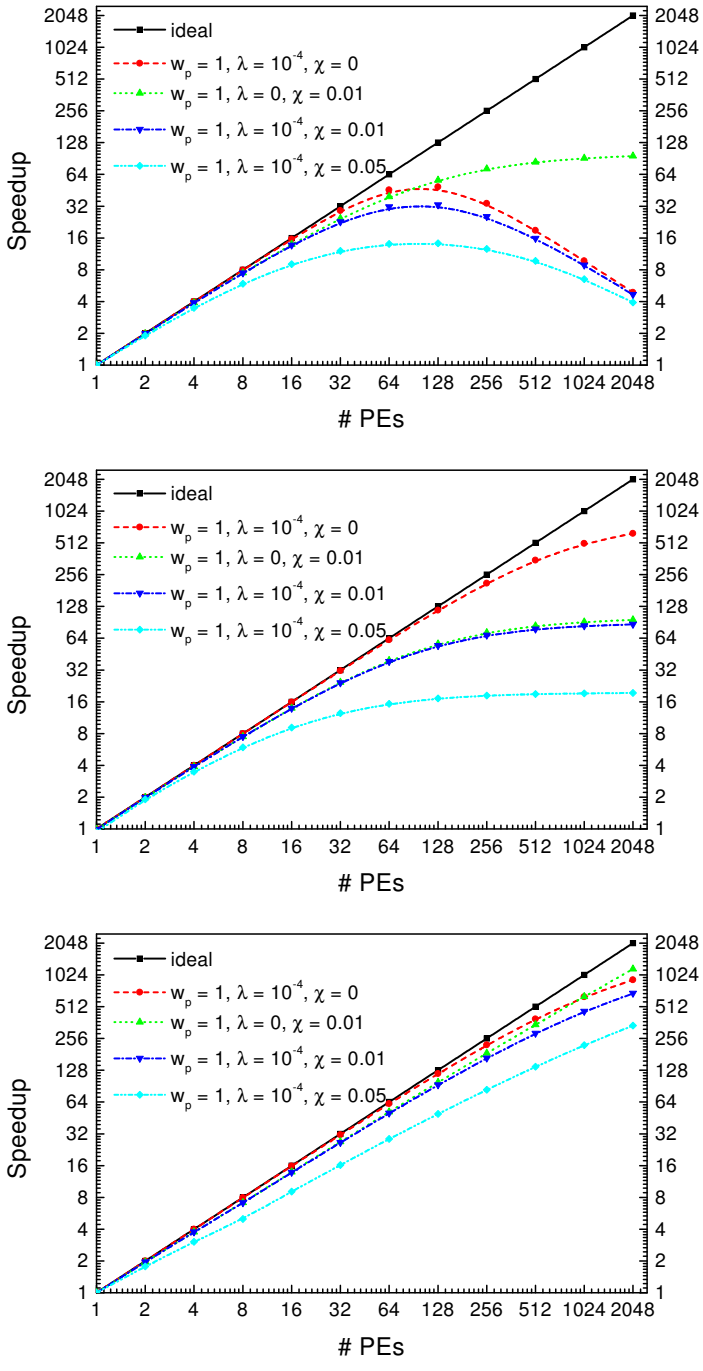


Figure 8. Estimations of realistic speedup curves if one includes the latency time and bandwidth of the processor interconnect. It is assumed that the problem can be parallelized for 100%. Different parameter values are compared for the latency time λ and bandwidth χ for the all-to-all communication (top), a tree like communication (middle) and local nearest neighbor communications (bottom). The ideal curve neglects communication completely.

data to neighbored PEs. Therefore the latency time part is constant whereas the amount of data to be sent and consequently the communication part is decreased with larger N_p. The communication function reads therefore

$$c(N_p) = f(N_p)\left(\lambda + \frac{\chi}{N_p^{2/3}}\right) \quad , \quad f(N_p) = \begin{cases} 0 & N_p = 1 \\ 2 & N_p = 2 \\ 4 & N_p = 4 \\ 6 & N_p \geq 8 \end{cases} \tag{108}$$

Here the function $f(N_p)$ was introduced to cover also the cases for small numbers of PEs, where a data exchange is not necessary in each spatial direction. As seen from Fig.8c the speedup curves are nearly linear with a slightly smaller slope than unity. However, for very large numbers of PEs the curves will also flatten. Nevertheless, the local communication model provides the best speedup behavior from all parallel algorithms and seems to be best suited for large parallel architectures.

5.4.4 Final Remark

Note that the local communication model in its present form is only valid for short range interaction potentials. If the potential is longer ranged than one spatial domain, the function $f(N_p)$ has to be modified. For long range interactions, all-to-all communications are generally required. In that case the tree-method may be mostly prefered.

This theoretical analysis demonstrates the importance of a fast interconnect between processors for the case of molecular dynamics simulations. Not included in the communication function $c(N_p)$ is the bandwidth function of the network. This, however, will only slightly change Figs.8a-c.

Appendix

A Calculating the Pressure

As is well known from thermodynamics the pressure may be calculated via the virial theorem. However, there are problems in deriving the expression for the pressure when working in periodic boundary conditions[108], due to the missing walls on which the pressure acts. The usual derivation of the virial theorem where a gas or a liquid is bound in a certain volume, fails in this case. However, using a ficticious surface, where particles may cross over, it is possible to obtain a similar expression for the pressure in periodic boundary conditions (PBC) as in finite volumes, which reads

$$P = \frac{2}{3V}\Big\langle \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 \Big\rangle - \Big\langle \frac{\partial U}{\partial \mathbf{r}_{ij}} \mathbf{r}_{ij} \Big\rangle \tag{109}$$

$$= \frac{NkT}{V} + \frac{1}{3V}\Big\langle \sum_{\boldsymbol{\alpha}} \sum_{i,j} \mathbf{F}(\mathbf{r}_{ij} - \boldsymbol{\alpha} L)(\mathbf{r}_{ij} - \boldsymbol{\alpha} L) \Big\rangle \tag{110}$$

where L is the length of the simulation box and the parameter $\boldsymbol{\alpha}$ accounts for the periodic images. If the range of interaction between particles is smaller than $L/2$, this translation corresponds to the so called *minimum image convention*.

A different derivation of the pressure, which is rather convenient for the case of molecular systems, starts from the thermodynamic definition

$$P = -\left(\frac{\partial A}{\partial V}\right)_T \tag{111}$$

where A is the Helmholtz free energy of the system, which can be expanded to give

$$P = \frac{1}{\beta Q}\frac{\partial Q}{\partial V} \tag{112}$$

where Q is the partition function

$$Q(V,T) = \frac{1}{N!h^{3N}}\int d\mathbf{r}\; d\mathbf{p}\; e^{-\beta\mathcal{H}(\mathbf{r},\mathbf{p})} \tag{113}$$

and $\mathcal{H}$ is the system's Hamiltonian. In order to introduce the volume as an independent parameter, the following change in variables is performed

$$\mathbf{r} = V^{1/3}\boldsymbol{\rho} \quad ; \quad \mathbf{p} = V^{-1/3}\boldsymbol{\pi} \quad ; \quad \boldsymbol{\pi} = mV^{1/3}\partial_t\boldsymbol{\rho} \tag{114}$$

Therefore the expression for the pressure can be written as

$$P = \frac{1}{N!h^{3N}}\frac{1}{\beta Q}\frac{\partial}{\partial V}\int\int d\boldsymbol{\rho}d\boldsymbol{\pi}\; e^{-\beta\mathcal{H}(V^{1/3}\boldsymbol{\rho},V^{-1/3}\boldsymbol{\pi})} \tag{115}$$

$$= -\frac{1}{N!h^{3N}}\frac{1}{Q}\int\int d\boldsymbol{\rho}d\boldsymbol{\pi}\;\frac{\partial}{\partial V}\mathcal{H}(V^{1/3}\boldsymbol{\rho},V^{-1/3}\boldsymbol{\pi})\; e^{-\beta\mathcal{H}(V^{1/3}\boldsymbol{\rho},V^{-1/3}\boldsymbol{\pi})} \tag{116}$$

$$= -\left\langle\frac{\partial}{\partial V}\mathcal{H}(V^{1/3}\boldsymbol{\rho},V^{-1/3}\boldsymbol{\pi})\right\rangle_T \tag{117}$$

In the following two examples are given how to calculate the pressure for specific systems.

A.1 Monatomic Systems

The Hamiltonian for a monatomic system, interacting via pair-forces may be written in scaled variables as

$$\mathcal{H} = \frac{1}{V^{\frac{2}{3}}}\sum_{i=1}^{N}\frac{\boldsymbol{\pi}^2}{2m_i} + \sum_{i,j=1;i<j}^{N}U(V^{\frac{1}{3}}\rho_{ij}) \tag{118}$$

where $\rho_{ij} = |\boldsymbol{\rho}_i - \boldsymbol{\rho}_j|$. Differentiating with respect to the volume gives

$$\frac{\partial\mathcal{H}}{\partial V} = -\frac{2}{3V^{\frac{5}{3}}}\sum_{i=1}^{N}\frac{\boldsymbol{\pi}_i^2}{2m_i} + \frac{1}{3V^{\frac{2}{3}}}\sum_{i,j=1;i<j}^{N}U'(V^{\frac{1}{3}}\rho_{ij})\rho_{ij} \tag{119}$$

Transforming back this expression into *real variables*

$$\frac{\partial\mathcal{H}}{\partial V} = -\frac{2}{3V}\sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{2m_i} + \frac{1}{3V}\sum_{i,j=1;i<j}^{N}U'(r_{ij})r_{ij} \tag{120}$$

leads to the equation for the pressure

$$P = \frac{1}{3V}\left\langle \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{m_i} - \sum_{i,j=1;i\neq j}^{N} U'(r_{ij}) r_{ij} \right\rangle_T \tag{121}$$

which is the same expression predicted by the Virial theorem[109].

A.2 Rigid Nonlinear Molecules

As outlined in Sec.3.4.1 the motion of a rigid body can be described as a translation of the center-of-mass (COM) and a rotation around the principal axis. The Hamiltonian is therefore described in terms of positional and orientational coordinates. The scaling procedure, Eq.114, is applied again only on the COM coordinates and momenta, since orientational degrees of freedom have no inherent length scale. The Hamiltonian in scaled variables can therefore be written as

$$\mathcal{H} = \frac{1}{V^{\frac{2}{3}}} \sum_{i=1}^{N} \frac{\boldsymbol{\pi}_i^2}{2M_i} + \frac{1}{2} \sum_{i=1}^{N} \boldsymbol{\omega}_i^T \mathbf{I}\, \boldsymbol{\omega}_i + \sum_{i,j=1;i<j}^{N} \sum_{a,b=1}^{N_s} U(V^{\frac{1}{3}} \rho_{ij}^{ab}) \tag{122}$$

Here $\boldsymbol{\pi}_i$ denotes the scaled momentum of the COM of molecule i, M_i the molecular mass, N_s the number of molecular sites and $\rho_{ij}^{ab} = |\boldsymbol{\rho}_i^a - \boldsymbol{\rho}_j^b|$ is the scaled distance between site a on molecule i and site b on molecule j. The position of site a is thereby given as

$$\boldsymbol{\rho}_i^a = \boldsymbol{\rho}_i + \mathbf{d}^a\, V^{-\frac{1}{3}} \tag{123}$$

where $\boldsymbol{\rho}_i$ is the position of the COM of molecule i and $\mathbf{d}^a$ the distance vector from the COM to site a. Since only the center of mass vector, $\mathbf{R}$, is scaled with the volume term, the distance vector, $\mathbf{d}$, is multiplied by $V^{-1/3}$. Differentiating Eq.122 with respect to the volume gives

$$P = \frac{1}{3V}\left\langle \frac{\boldsymbol{\pi}_i^2}{2M_i} - \sum_{i,j=1;i<j}^{N} \sum_{a,b=1}^{N_s} U'(R_{ij}^{ab}) \frac{1}{R_{ij}^{ab}} (\mathbf{R}_{ij} - \mathbf{d}^{ab}) \mathbf{R}_{ij} \right\rangle_T \tag{124}$$

$$= \frac{1}{3V}\left\langle \frac{\boldsymbol{\pi}_i^2}{2M_i} - \sum_{i,j=1;i<j}^{N} \sum_{a,b=1}^{N_s} \mathbf{F}_{ij}^{ab} \mathbf{R}_{ij} + \sum_{i=1}^{N} \sum_{a=1}^{N_s} \mathbf{F}_i^a \mathbf{d}_a \right\rangle_T \tag{125}$$

where $\mathbf{F}_{ij}^{ab}$ is the force, acting from site a of molecule i on site b of molecule j and $\mathbf{F}_i^a$ the total force on site a of molecule i. In order to write Eq.125 in the present form, it was used that the mean value $\langle \mathbf{F}_i^a \mathbf{d}_b \rangle$ vanishes. The actual values of the scalar product, however, may give contributions to the fluctuations of the pressure. The last term in Eq.125 acts as a kind of correction, which reduces the pressure with respect to a simple superposition of pairforces in the expression of the virial theorem. This fact may be interpreted as arising from the constraint forces which keep the molecule rigid.

References

1. Y. Duan and P. A. Kollman. Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution. *Science*, 282:740, 1998.
2. J. Roth, F. Gähler, and H.-R. Trebin. A molecular dynamics run with 5.180.116.000 particles. *Int. J. Mod. Phys. C*, 11:317–322, 2000.
3. B. J. Alder and T. E. Wainwright. Phase transition for a hard sphere system. *J. Chem. Phys.*, 27:1208–1209, 1957.
4. B. J. Alder and T. E. Wainwright. Studies in molecular dynamics. I. General method. *J. Chem. Phys.*, 31:459, 1959.
5. J. Stadler, R. Mikulla, and H.-R. Trebin. IMD: A software package for molecular dynamics studies on parallel computers. *Int. J. Mod. Phys. C*, 8:1131–1140, 1997.
6. Y. Duan, L. Wang, and P. A. Kollman. The early stage of folding of villin headpiece subdomain observed in 200-nanosecond fully solvated molecular dynamics simultion. *Proc. Natl. Acad. Sci. USA*, 95:9897, 1998.
7. `http://wserv1.dl.ac.uk/CCP/CCP5/`.
8. `http://www.amber.ucsf.edu/amber/amber.html`.
9. `http://www.scripps.edu/brooks/c27docs/Charmm27.Html`.
10. `http://www.ks.uiuc.edu/Research/namd/`.
11. `http://www.emsl.pnl.gov:2080/docs/nwchem/nwchem.html`.
12. `http://www.cs.sandia.gov/ sjplimp/lammps.html`.
13. W. L. Cui, F. B. Li, and N. L. Allinger. *J. Amer. Chem. Soc.*, 115:2943, 1993.
14. N. Nevins, J. H. Lii, and N. L. Allinger. *J. Comp. Chem.*, 17:695, 1996.
15. S. L. Mayo, B. D. Olafson, and W. A. Goddard. *J. Phys. Chem.*, 94:8897, 1990.
16. M. J. Bearpark, M. A. Robb, F. Bernardi, and M. Olivucci. *Chem. Phys. Lett.*, 217:513, 1994.
17. T. Cleveland and C. R. Landis. *J. Amer. Chem. Soc.*, 118:6020, 1996.
18. A. K. Rappé, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff. *J. Amer. Chem. Soc.*, 114:10024, 1992.
19. Z. W. Peng, C. S. Ewig, M.-J. Hwang, M. Waldman, and A. T. Hagler. Derivation of class ii force fields. 4. van der Waals parameters of Alkali metal cations and Halide anions. *J. Phys. Chem.*, 101:7243–7252, 1997.
20. W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Amer. Chem. Soc.*, 117:5179–5197, 1995.
21. A. D. Mackerell, J. Wiorkiewiczkuczera, and M. Karplus. *J. Amer. Chem. Soc.*, 117:11946, 1995.
22. W. L. Jorgensen, D. S. Maxwell, and J. Tiradorives. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Amer. Chem. Soc.*, 118:11225–11236, 1996.
23. T. A. Halgren. Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *J. Comp. Chem.*, 17:490–519, 1996.
24. J. Kong. *J. Chem. Phys.*, 59:2464, 1973.
25. M. Waldman and A. T. Hagler. *J. Comp. Chem.*, 14:1077, 1993.
26. J. Delhommelle and P. Millié. Inadequacy of the lorentz-bertelot combining rules

for accurate predictions of equilibrium properties by molecular simulation. *Molec. Phys.*, 99:619–625, 2001.
27. E. L. Pollock and B. J. Alder. Static dielectric properties of stockmayer fluids. *Physica*, 102A:1, 1980.
28. P. Ewald. Die berechnung optischer und elektrostatischer gitterpotentiale. *Ann. Phys.*, 64:253, 1921.
29. S. W. de Leeuw, J. M. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants. *Proc. R. Soc. London*, A373:27, 1980.
30. S. W. de Leeuw, J. M. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. II. Equivalence of boundary conditions. *Proc. R. Soc. London*, A373:57, 1980.
31. P. Gibbon. Long range interactions in many-particle simulation. (this volume).
32. J. W. Perram, H. G. Petersen, and S. W. de Leeuw. An algorithm for the simulation of condensed matter which grows as the 3/2 power of the number of particles. *Molec. Phys.*, 65:875–893, 1988.
33. D. Fincham. Optimisation of the Ewald sum for large systems. *Molec. Sim.*, 13:1–9, 1994.
34. J. Kolafa and J. W. Perram. Cutoff errors in the Ewald summation formulae for point charge systems. *Molec. Sim.*, 9:351–368, 1992.
35. D. M. Heyes. Electrostatic potentials and fields in infinite point charge lattices. *J. Chem. Phys.*, 74:1924–1929, 1980.
36. W. Smith. Point multipoles in the Ewald sum. *CCP5 Newsletter*, 46:18–30, 1998.
37. T. M. Nymand and P. Linse. Ewald summation and reaction field methods for potentials with atomic charges, dipoles and polarizabilities. *J. Chem. Phys.*, 112:6152–6160, 2000.
38. G. Salin and J. P. Caillol. Ewald sums for yukawa potentials. *J. Chem. Phys.*, 113:10459–10463, 2000.
39. L. Verlet. Computer experiments on classical fluids. I. Thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98, 1967.
40. C. W. Gear. *Numerical initial value problems in ordinary differential equations*. Prentice Hall, Englewood Cliffs, NJ, 1971.
41. R. W. Hockney. The potential calculation and some applications. *Meth. Comput. Phys.*, 9:136–211, 1970.
42. D. Beeman. Some multistep methods for use in molecular dynamics calculations. *J. Comp. Phys.*, 20:130–139, 1976.
43. H. F. Trotter. On the product of semi-groups of operators. *Proc. Am. Math. Soc.*, 10:545–551, 1959.
44. O. Buneman. Time-reversible difference procedures. *J. Comp. Phys.*, 1:517–535, 1967.
45. E. Hairer and P. Leone. Order barriers for symplectic multi-value methods. In D. Griffiths, D. Higham, and G. Watson, editors, *Pitman Research Notes in Mathematics*, volume 380, pages 133–149, 1998.
46. D. Okunbor and R. D. Skeel. Explicit canonical methods for Hamiltonian systems. *Math. Comput.*, 59:439–455, 1992.
47. E. Hairer. Backward error analysis of numerical integrators and symplectic methods.

Ann. Numer. Math., 1:107–132, 1994.
48. E. Hairer and C. Lubich. The lifespan of backward error analysis for numerical integrators. *Numer. Math.*, 76:441–462, 1997.
49. S. Reich. Backward error analysis for numerical integrators. *SIAM J. Numer. Anal.*, 36:1549–1570, 1999.
50. D. M. Stoffer. *Some geometrical and numerical methods for perturbed integrable systems*. PhD thesis, Swiss Federal Institute of Technology, Zürich, 1988.
51. J. M. Sanz-Serna M. Calvo. *Numerical Hamiltonian Problems*. Chapman and Hall, London, 1994.
52. R. D. Skeel. Integration schemes for molecular dynamics and related applications. In M. Ainsworth, J. Levesley, and M. Marletta, editors, *The Graduate Student's Guide to Numerical Analysis*, pages 119–176, New York, 1999. Springer.
53. M. E. Tuckerman and W. Langel. Multiple time scale simulation of a flexible model of co_2. *J. Chem. Ohys.*, 100:6368, 1994.
54. P. Procacci, T. Darden, and M. Marchi. A very fast Molecular Dynamics method to simulate biomolecular systems with realistic electrostatic interactions. *J. Phys. Chem.*, 100:10464–10468, 1996.
55. P. Procacci, M. Marchi, and G. L. Martyna. Electrostatic calculations and multiple time scales in molecular dynamics simulation of flexible molecular systems. *J. Chem. Phys.*, 108:8799–8803, 1998.
56. P. Procacci and M. Marchi. Taming the Ewald sum in molecular dynamics simulations of solvated proteins via a multiple time step algorithm. *J. Chem. Phys.*, 104:3003–3012, 1996.
57. J. J. Biesiadecki and R. D. Skeel. Dangers of multiple time step methods. *J. Comp. Phys.*, 109:318–328, 1993.
58. J. L. Scully and J. Hermans. Multiple time steps: limits on the speedup of molecular dynamics simulations of aqueous systems. *Molec. Sim.*, 11:67–77, 1993.
59. B. J. Leimkuhler and R. D. Skeel. Symplectic numerical integrators in constrained Hamiltonian systems. *J. Comp. Phys.*, 112:117–125, 1994.
60. T. Schlick. Some failures and success of long timestep approaches to biomolecular simulations. In P. Deuflhard, J. Hermans, B. J. Leimkuhler, A. Mark, S. Reich, and R. D. Skeel, editors, *Lecture notes in computational science and engineerung. Algorithms for macromolecular modelling*, volume 4, pages 221–250, New York, 1998. Springer.
61. E. Barth and T. Schlick. Overcoming stability limitations in biomolecular danamics. I. Combining force splitting via extrapolation with Langevin dynamics. *J. Chem. Phys.*, 109:1617–1632, 1998.
62. E. Barth and T. Schlick. Extrapolation versus impulse in multiple-timestepping schemes. II. Linear analysis and applications to Newtonian and Langevin dynamics. *J. Chem. Phys.*, 109:1633–1642, 1998.
63. B. Garcia-Archilla, J. M. Sanz-Serna, and R. D. Skeel. Long-time-step methods for oscillatory differential equations. *SIAM J. Sci. Comp.*, 20:930–963, 1998.
64. B. Garcia-Archilla, J. M. Sanz-Serna, and R. D. Skeel. The mollified impulse method for oscillatory differential equations. In D. F. Griffiths and G. A. Watson, editors, *Numerical analysis 1997*, pages 111–123, London, 1998. Pitman.
65. B. Garcia-Archilla, J. M. Sanz-Serna, and R. D. Skeel. The mollified impulse method

for oscillatory differential equations. *SIAM J. Sci. Comp.*, 20:930–963, 1998.
66. J. A. Izaguirre. *Longer time steps for molecular dynamics*. PhD thesis, University of Illinois at Urbana-Champaign, 1999.
67. J. A. Izaguirre, S. Reich, and R. D. Skeel. Longer time steps for molecular dynamics. *J. Chem. Phys.*, 110:9853, 1999.
68. J. Barojas, D. Levesque, and B. Quentrec. Simulation of diatomic homonuclear liquids. *Phys. Rev. A*, 7:1092–1105, 1973.
69. J. B. Kuipers. *Quaternions and rotation sequences*. Princeton University Press, Princeton, New Jersey, 1998.
70. D. Fincham. Leapfrog rotational algorithms. *Molec. Simul.*, 8:165, 1992.
71. A. Dullweber, B. Leimkuhler, and R. McLachlan. Symplectic splitting methods for rigid body molecular dynamics. *J. Chem. Phys.*, 107:5840–5851, 1997.
72. A. Kol, B. B. Laird, and B. J. Leimkuhler. A symplectic method for rigid-body molecular simulation. *J. Chem. Phys.*, 107:2580–2588, 1997.
73. A. J. Stone, A. Dullweber, M. P. Hodges, P. L. A. Popelier and D. J. Wales. ORIENT: A program for studying interactions between molecules, Version 3.2, University of Cambridge (1995-1997), available at `http://www-stone.ch.cam.ac.uk/programs.html`.
74. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran*. Cambridge University Press, Cambridge, 1992.
75. J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: Molecular dynamics of n-Alkanes. *J. Comp. Phys.*, 23:327–341, 1977.
76. H. C. Andersen. Rattle: a velocity version of the Shake algorithm for molecular dynamics calculations. *J. Comp. Phys.*, 52:24–34, 1982.
77. B. Hess, H. Bekker, H. J. C. Berendsen, and J. G. E. M. Fraaije. LINCS: a linear constraint solver for molecular simulations. *J. Comp. Chem.*, 18:1463–1472, 1997.
78. T. R. Forester and W. Smith. SHAKE, Rattle and Roll: efficient constraint algorithms for linked rigid bodies. *J. Comp. Chem.*, 19:102–111, 1998.
79. X.-W. Wu and S.-S. Sung. Constraint dynamics algorithm for simaultion of semiflexible macromolecules. *J. Comp. Chem.*, 19:1555–1566, 1998.
80. M. P. Allen and D. J. Tildesley. *Computer simulation of liquids*. Oxford Science Publications, Oxford, 1987.
81. L. V. Woodcock. Isothermal molecular dynamics calculations for liquid salt. *Chem. Phys. Lett.*, 10:257–261, 1971.
82. W. G. Hoover, A. J. C. Ladd, and B. Moran. High strain rate plastic flow studied via nonequilibrium molecular dynamics. *Phys. Rev. Lett.*, 48:1818–1820, 1982.
83. D. J. Evans, W. G. Hoover, B. H. Failor, B. Moran, and A. J. C. Ladd. Nonequilibrium molecular dynamics via Gauss's principle of least constraint. *Phys. Rev. A*, 28:1016–1021, 1983.
84. D. J. Evans. Computer experiment for nonlinear thermodynamics of Couette flow. *J. Chem. Phys.*, 78:3298–3302, 1983.
85. H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, 81:3684, 1984.
86. H. J. C. Berendsen. Transport properties computed by linear response through weak

coupling to a bath. In M. Meyer and V. Pontikis, editors, *Computer Simulation in Materials Science*, pages 139–155, Amsterdam, 1991. Kluwer Academic Publishers.
87. T. Morishita. Fluctuation formulas in molecular dynamics simulations with the weak coupling heat bath. *J. Chem. Phys.*, 113:2976–2982, 2000.
88. T. Schneider and E. Stoll. Molecular dynamics study of a three dimensional one-component model for distortive phase transitions. *Phys. Rev. B*, 17:1302–1322, 1978.
89. H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.*, 72:2384, 1980.
90. E. Bonomi. *J. Stat. Phys.*, 39:167, 1985.
91. H. Tanaka, K. Nakanishi, and N. Watanabe. *J. Chem. Phys.*, 78:2626, 1983.
92. M. E. Riley, M. E. Coltrin, and D. J. Diestler. A velocity reset method of simulating thermal motion and damping in gas-solid collisions. *J. Chem. Phys.*, 88:5934–5942, 1988.
93. G. Sutmann and B. Steffen. Correction of finite siye effects in molecular dynamics simulation applied to friction, 2001. submitted to Comp. Phys. Comm.
94. S. Nosé. A unified formulation of the constant temperature molecular dynamics methods. *J. Chem. Phys.*, 81:511–519, 1984.
95. S. Nosé. A molecular dynamics methood for simulations in the canonical ensemble. *Molec. Phys.*, 52:255–268, 1984.
96. K. Zare and V. Szebehely. Time tranformations for the extended phase space. *Celestial Mech.*, 11:469, 1975.
97. W. G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31:1695–1697, 1985.
98. D. J. Evans and G. P. Morris. The isothermal isobaric molecular dynamics ensemble. *Phys. Lett. A*, 98:433–436, 1983.
99. D. J. Evans and G. P. Morris. Isothermal isobaric molecular dynamics. *Chem. Phys.*, 77:63–66, 1983.
100. G. M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 483–485, Reston, Va., 1967. AFIPS Press.
101. G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker. *Solving problems on concurrent processors: Volume 1*. Prentice Hall, Englewood Cliffs, NJ, 1988.
102. T. Lippert, H. Hoeber, G. Ritzenhöfer, and K. Schilling. Hyper-Systolic Processing on APE100/Quadrics N^2-Loop Computations. Technical Report HLRZ 95-45, WUB 95-21, 1995.
103. T. Lippert, A. Seyfried, A. Bode, and Schilling K. Hyper-systolic parallel computing. *IEEE Trans. Paral. Distr. Syst.*, 9:97–108, 1998.
104. R. Murty and D. Okunbor. Efficient parallel algorithms for molecular dynamics simulations. *Parall. Comp.*, 25:217–230, 1999.
105. V. E. Taylor, R. L. Stevens, and K. E. Arnold. Parallel molecular dynamics: Communication requirements for parallel machines. In *Proc. of the fifth Symposium on the Frontiers of Massively Parallel Computation*, pages 156–163, 1994.
106. D. Brown, J. H. R. Clarke, M. Okuda, and T. Yamazaki. A domain decomposition parallel processing algorithm for molecular dynamics simulations of polymers. *Comp.*

Phys. Comm., 83:1, 1994.
107. M. Pütz and A. Kolb. Optimization techniques for parallel molecular dynamics using domain decomposition. *Comp. Phys. Comm.*, 113:145–167, 1998.
108. J. J. Erpenbeck and W. W. Wood. Molecular dynamics techniques for hard core systems. In B. J. Berne, editor, *Statistical mechanics B: Modern theoretical chemistry*, volume 6, pages 1–40, New York, 1977. Plenum.
109. H. Goldstein. *Classical Mechanics*. Addison Wesley, Reading, Massachusetts, 1950.