FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

Classical Molecular Dynamics and Parallel Computing

Godehard Sutmann

FZJ-ZAM-IB-2002-11

September 2002

(letzte Änderung: 05.09.2002)

Preprint: Parallelization of Algorithms in Physics
13th Summer School on Computing Techniques in Physics
Třešt, Czech Republic, September 16-23, 2002

Classical Molecular Dynamics and Parallel Computing[a]

Godehard Sutmann

John von Neumann Institute for Computing (NIC) and
Central Institute for Applied Mathematics (ZAM)
Research Centre Jülich, 52425 Jülich (FZJ), Germany
g.sutmann@fz-juelich.de

Abstract: an overview is given about classical molecular dynamics computer simulation. The main ingredients which enter into a program are presented and special emphasis is put on parallelisation approaches. Also the limits and challenges of parallel algorithms are discussed.

Overview

Computer simulation methods have become a very powerful tool to attack the many-body problem in statistical physics, physical chemistry, biophysics and astrophysics[b]. 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 either originate from theoretical considerations or are derived from 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 before carrying out the experiment to prove the potential outcome by computer simulations. In that way one can say that the field of computational science, especially computer simulation methods, has developed into an 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 by its own.

The traditional simulation methods for many-body systems can be divided into the 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 energies are always accepted, those to higher energies are accepted with a probability governed by the 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, H, and consequently Hamilton's equations of motion

[a] An extended version of the article may be found in: *Quantum simulations of many-body systems: from theory to algorithm.* Eds. J.Grotendorst, D.Marx and A.Muramatsu. NIC-series Vol.10, pp. 211-254 (2002).
[b] In this article the focus of molecular dynamics methods is put on problems which have their origin in the field of liquids and solids, although the method covers a very much broader field. Textbooks on MD simulation methods are e.g. [1,2,3,4].

(1) $$\frac{\partial p_i}{\partial t} = -\frac{\partial H}{\partial q_i} \quad , \quad \frac{\partial q_i}{\partial t} = \frac{\partial H}{\partial p_i}$$

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 the configuration space but the whole phase space is probed 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 constituents. This model has to be tested against (i) experimental results, i.e. it should reproduce or approximate experimental findings like distribution functions or phase diagrams, and (ii) 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:

- A model for the interaction between system constituents (atoms, molecules, surfaces etc.) is needed. Often, it is assumed that particles interact only pair-wise, 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.
- An integrator is needed, which propagates particle positions and velocities from time t to $t+\delta t$. It is a finite difference scheme which propagates trajectories discretely in time. The time step δt has to be properly chosen in order to guarantee stability of the integrator, i.e. there should be no drift in the system's energy.
- 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. With this method it is then 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 (unless the physical system is not very simple, it is often the case that the model is *constructed* to reproduce a small class of properties rather accurate. At the same time other properties may be predicted in a rather poor way. For example, there are tens of models of water which all have their advantages and disadvantages in predicting “real water”.). 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 are the accessible time- and length-scales which may be explored by microscopic simulations. 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 "length-time diagram" and typical scales are of order of Angstrom and picoseconds. 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 intra-molecular vibrations, which are orders of magnitude slower than those of electron motions. Consequently, the time step of integration is larger (~1fs) and trajectory lengths are of order nanoseconds and accessible length scales 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 time steps. Furthermore, since the solvent is not simulated explicitly, the length scales 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 [5] who simulated the phase transition of a system of hard spheres. The general method, however, was presented only two years later [6]. In this early simulation, which was run 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 [7]. In this case, a model system for Argon was simulated and not only binary collisions were taken into account but the interactions were modelled 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 (e.g. autocorrelation functions), and transport coefficients (e.g. 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 45secs. The calculation of 50000 time steps then took more than three weeks![c]

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

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 $5x10^9$ particles [8], corresponding to a length scale of several 1000Å. This simulation was carried out with the highly memory optimised MD program IMD [9], which used the 512 nodes with 256 MB memory each, quite efficiently. However, the limits of such a demonstration run 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 1μsec, where they simulated the folding process of a small protein [10]. The biopolymer was modelled with a 596 interaction site model dissolved in a system of 3000 water molecules. Using a timestep of integration of $2x10^{-15}$ secs, the program was run for $5x10^8$ 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 it 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) [11] 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 available free of charge, are, e.g., Amber/Sander [12], CHARMM [13], NAMD [14], NWCHEM [15] and LAMMPS [16].

Interactions between particles

A many particle system may be completely characterized by it's Hamiltonian $H=H_0+H_1$, where H_0 is the *internal* part of the Hamiltonian, given as

$$H_0 = \sum_{i=1}^{N} \frac{p_i^2}{2m_i} + \sum_{i<j}^{N} u(r_i, r_j) + \sum_{i<j<k}^{N} u^{(3)}(r_i, r_j, r_k) + \ldots \qquad (2)$$

Here p is the momentum of the particles, m the mass and u and $u^{(3)}$ are pair and three-body interaction potentials. H_1 is an external part, which can include time dependent effects or external sources for a force. Besides boundary conditions, which are imposed, it is the potential model which completely determines the system from the physical point of view. In classical simulations the *objects* are most often described by point-like centres which interact through pair- or multi-body 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 molecules are modelled 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 $\boldsymbol{F}=-\nabla U$, the combination of parameters and functions is also called a *force field*. Different types of

force fields were developed during the last ten years (e.g. AMBER [17], CHARMM [18] or OPLS [19]).

Typical forcefield components include for nonbonded contributions Lennard-Jones (12,6) or (9,6) potentials and Coulomb r^{-1} terms. Furthermore, bonded terms are often modelled by potential types for bond stretching (e.g. quadratic- or Morse type), bond bending (3-body potentials) and improper and proper dihedrals (4-body potentials).

There are major differences to be noticed for the potential forms. The first distinction is to be made between pair- and multi-body 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 multi-body potentials come into play for non-bonded interactions, the evaluation of interaction partners becomes increasingly more complex and, without using interaction lists, may dramatically slow down the execution of the program. Only for the case of bonded interactions, e.g. torsional or bending motions of a molecule, where interaction partners may be stored in static lists in the beginning of the simulation, the interaction can be calculated efficiently.

A second important difference between interactions is the spatial extent of the potential, classifying it into short and long range interactions. Assume that the potential form is r^{-n} for Large separations *r* between particles. If $n>d$, where *d* is the dimension of the system, the potential is called short ranged, otherwise it is long ranged (the integral over the potential function does or does not converge). In the latter case, a particles' potential energy gets contributions from *all particles of the universe*, otherwise the interaction is bound to a certain region, which is often modelled 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 systems, where vacuum boundary conditions are applied, a direct summation of Coulomb interactions may be applied. This is the most accurate but also the most expensive way to calculate interactions ($O(N^2)$). For the case of periodic boundary conditions, lattice summation methods like the Ewald method [20,21] may be applied. Formally this method, which splits the lattice sum into a direct part and a reciprocal part, is an exact method. However, the implementation uses three parameters, which may be tuned to result in an acceptable error and a fast execution time. The interplay between these parameters leads, for a constant error, to a scaling behaviour of $O(N^{3/2})$. During the last years several other, more sophisticated methods like Particle Mesh Ewald [22], Fast Multipole Method [23] or Multigrid methods [24,25] were developed, which scale as O(NlogN) or even O(N) [26].

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 occurring due to a finite number representation. The round off errors may already have a strong influence on the accuracy of the representation of “true” trajectories. In a numerical study [27] it was shown for Lennard-Jones fluid that a single change of the sequence of floating-point operations in the force summation routine induces a perturbation which

leads to a divergent growth between a reference trajectory (without numerical perturbation) and the disturbed one. With a 64-bit number representation the two trajectories reach a stochastic regime after ~10 ps[d]. This gives an estimate for the upper time which is possible to apply in the calculation of correlation functions.

From this result it is clear that particle trajectories can only be interpreted in a statistical sense and correlation functions do make sense only up to a certain time. In addition, 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 adequately (this may be checked with simple model systems for which analytical solutions exist)
- stable, in the sense that it conserves energy and that small perturbations 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

A famous integration scheme, which is used in MD simulations is the Verlet scheme, which may be derived from the Liouville equation as so called position or velocity Verlet schemes [28]. The advantage of this method is that it is symplectic (it conserves the volume in phase space) and time reversible. The Verlet method has received strong attraction during the last time since it allows to build up symplectic multiple-time-step methods [29] (separation of different time scales occurring in the system) which allow for a considerable speed-up of the program performance.

Thermodynamic 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 describes the properties of a certain thermodynamic ensemble.

One can distinguish four different types of control mechanisms:

1. Differential control: the thermodynamic quantity is fixed to the prescribed value and no fluctuations around an average value occur.
2. Proportional control: the variables, coupled to the thermodynamic property *f*, are corrected in each integration step through a coupling constant towards the prescribed value of *f*. The coupling constant determines the strength of the fluctuations around the mean value of *f*.
3. 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.
4. 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*.

[d] This time depends on the potential model and the thermodynamic conditions.

Molecular Dynamics on Parallel Computers

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 [30]. As has been already mentioned in the introduction it is, in principle, possible to treat multi-billion particle systems. However, the whole success of parallel computing strongly depends both on the underlying problem to be solved and the optimisation of the computer program. The former point is related to a principle problem which is manifested in the so called Amdahl's law [31]. If a problem has inherently certain parts which can be solved only in serial, this will give an upper limit for the parallelisation 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

$$\sigma = \frac{N_p}{w_p + N_p w_s} \tag{3}$$

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.3 it is obvious that the maximum efficiency is obtained when the problem is completely parallelisable, i.e. $w_p=1$, which gives theoretically 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.

If the parallel work is limited to 50%, the maximum speedup is bound to σ=2, a parallel work of 90% gives a σ=10 and a parallel work of 99% leads to σ=100. These are of course bad news if one wants to speed up the program execution just by running it on hundreds and thousands of processors and this was criticised by Gene Amdahl in view of building big parallel computers. Thus, 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 guarantee a fast execution of programs, but the problem itself has to be chosen properly.

It should be noted, however, that there were different metrics introduced by other authors to save the perspectives of parallel computing [32,33,34]. One is the so called *isoefficiency function,* introduced by Vipin Kumar [32]. The idea behind this function is not to consider a problem of a given size and to ask, how much faster one can solve this problem on a set of parallel processors. It is more important to consider the question, whether it is possible to increase both the number of processors and the problem size while keeping the efficiency (defined as the ratio of speedup and number of processors) at a constant value. Algorithms which allow for this are, according to this criterion, *scalable*. This criterion somehow overcomes the limitations, introduced by Amdahl's law and points into the direction that larger computers should be related to the solution of bigger problems. However, this criterion overlooks that not only the size of a system matters but that in several areas also the length of a simulation becomes more and more important (e.g. protein folding, cosmology). For these types of simulations the only solution to become tractable on massively parallel systems is to exclude serial work.

Concerning MD programs there are good news about the inherent parallelism. There are only a few components which have to be considered. The heart of an MD program is essentially the force routine, where mainly intermolecular distances are computed. Since this routine has the largest complexity in an MD program it usually costs more than 90% of

the overall execution time. Other important tasks are the integration of motion, the parameter set-up at the beginning of the simulation and the file input/output (I/O). The integrator, which has *O(N)* complexity, may easily be parallelised, since the loop over *N* particles may be subdivided and performed on different processors. The parameter set-up most often cannot be done completely in parallel. It is here where every processor gets its relevant information in order to start the simulation, i.e. system parameters are calculated, memory for arrays is allocated and initial values for the arrays are set. If a certain part can be done in parallel, the information has to be broadcasted. However, if the number of time steps, calculated in the simulation is large, the ratio between initial set-up and main simulation part may be considered negligible. The file I/O is a more complicated problem. The message passing interface MPI 1 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 2 standard offers parallel read/write operations, which lead to a considerable efficiency gain with respect to MPI 1. However, the efficiency obtained depend strongly on the implementation on different architectures.

Another crucial point for the parallel performance is, of course, the implementation of the program. A problem which is inherently 100% parallel will not be solved with maximum speed if the program is not 100% mapped onto this problem, i.e. if the code is not optimally implemented. In this case performance analysis tools may be used to optimise the code. But not only implementation problems may result in a bad performance. Also the workload between the processors may differ considerably if the system has strong inhomogeneities. Load-balancing methods have to be used in this case in order to reorganise particle or spatial domain distributions on the processors.

Independently of the implementation of the code, Eq.3 gives only an upper theoretical limit which will only be reached in very rare cases. The original formulation of Amdahl's law does consider only the structure of the problem, which has to be parallelised. In general, it will be 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 some extra overhead which depends on the latency and the bandwidth of the inter-processor network. This strongly depends on the hardware implementation. Considering the extreme cases of the fast interconnect of the CRAY-T3E and the cheap and often used Fast Ethernet it is found that the latency time (time which is used to initialise a communication) differs by a factor of ~50, while bandwidths differ by a factor of ~35. In order to get a more realistic estimation of the speedup behaviour of a parallel program it is therefore very important to include the communication step between processors or, in the case of SMP systems, the access time of global memory. The disadvantage of this procedure is, however, that the analysis becomes strongly dependent of hardware on which the program is executed. Also the number of parameters is increased by the latency time and a bandwidth function. Nevertheless, this type of analysis gives very important information whether or not it is useful or even possible to map a program with a certain problem size on a given computer system.

Parallelisation approaches may be subdivided into five different methods:

1. Farming or cloning: every processor executes its own program independently of the others. This is a trivial parallelisation, which is limited to small systems. A performance gain may be achieved through averaging over different trajectories, created in different

runs. This approach avoids completely the limitations of Amdahl's law but due to the small system restriction, this approach is rarely used for molecular dynamics.

2. Master-slave approach: a master-node distributes the work on all other processors and controls the workload. This approach may be chosen if not only the data are distributed but also the tasks on the processors are different. If the distribution of data and tasks vary during the simulation a control node may be useful. This approach may be, however, rather communication intensive, if the master has to distribute and to balance the work frequently. Most often, problems which are solved by MD simulations are organized in a way that every processor can work on the same problem, i.e. no distribution of tasks. Therefore a master-slave approach is not often applied.
3. Atom-decomposition scheme [35,36]: In this algorithm particles are distributed onto the processors which store their coordinates during the whole simulation run. In order to calculate interactions between the particles, coordinates have to be distributed to all other processors, where the forces are calculated and subsequently are sent back (systolic loop algorithm). This implies a global communication between processors. With increasing number of processors the calculation/communication-ratio becomes smaller and smaller and may even lead to a non-monotonous speed-up curve. Another variant of this decomposition scheme relies on a data replication among the processors. The force-loops are then parallelised in a way that every processor works on a certain subset of particle indices. The forces are then distributed again via a global data exchange (O(p)).
4. Force-decomposition scheme [35,37]: the force-matrix is distributed onto the processors and every processor calculates partial forces on $N/\sqrt{p}$ particles. Coordinates and forces may either be exchanged via a systolic loop algorithm or the coordinates may be replicated on every processor and the forces are distributed in a communication step. With respect to the atom-decomposition scheme this algorithm shows a better scaling due the reduced amount of data exchange ($O(\sqrt{p})$).
5. Domain-decomposition scheme [35]: the physical space in which the system under study is located, is subdivided into sub-domains, which are distributed onto the processors. Every processor then simply works on those particles which are located in these sub-domains. If the particles motion is diffusive, it will be necessary to update the particles coordinates from time to time on the processors since particles will move between sub-domains. If there are only short range interactions present between particles, it is only necessary to exchange coordinates between those processors which are adjoined to each other. In 2-dimensional problems these are 8 neighboured processors, in 3-dimensional problems 26. This keeps the overall communication bound to fixed number of processors and makes it possible to achieve rather good speed-up behaviour.

References

1. M.P. Allen and D.J. Tildesley, *Computer simulation of liquids*, Oxford Science Publications, Oxford, 1987.
2. J.M. Haile, *Molecular Dynamics Simulation*, Wiley, New York, 1997.
3. D. Frenkel and B. Smit, *Understanding molecular simulation. From algorithms to applications*, Academic Press, San Diego, 1996.
4. D.C. Rapaport, *The art of molecular dynamics simulation*, Cambridge University Press, Cambridge, 1995.
5. B.J. Alder and T.E. Wainwright. *Phase transition for a hard sphere system.* J. Chem. Phys., 27:1208-1209, 1957.
6. B.J. Alder and T.E. Wainwright. *Studies in molecular dynamics. I. General method.* J. Chem. Phys., 31:459, 1959.

7. A. Rahman, *Correlation in the motion of atoms in liquid Argon*, Phys. Rev. A, 136, 405-411, 1964.
8. 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.
9. 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.
10. 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.
11. http://wserv1.dl.ac.uk/CCP/CCP5/.
12. http://www.amber.ucsf.edu/amber/amber.html.
13. http://www.scripps.edu/brooks/c27docs/Charmm27.html.
14. http://www.ks.uiuc.edu/Research/namd/.
15. http://www.emsl.pnl.gov:2080/docs/nwchem/nwchem.html.
16. http://www.cs.sandia.gov/~sjplimp/lammps.html.
17. 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.
18. A.D. Mackerell, J. Wiorkiewiczkuczera, and M. Karplus. J. Amer. Chem. Soc., 117:11946, 1995.
19. 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.
20. P.Ewald. *Die berechnung optischer und elektrostatischer gitterpotentiale.* Ann. Phys., 64:253, 1921.
21. 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.
22. T. Darden, D. York and L. Pedersen, *A NlogN method for Ewald sums in large systems*, J. Chem. Phys., 98, 10089, 1993.
23. L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys., 73, 325-348, 1987.
24. B. Sandak and A. Brandt, *Multiscale fast summation of long range charge and dipolar interactions*, in: Multiscale computational methods in chemistry and physics, Eds. A. Brandt, J. Berholc, K. Binder, IOS Press, Amsterdam, pp. 6-31, 2001.
25. R.D. Skeel and I. Tezcan and D.J. Hardy, *Multiple grid methods for classical molecular dynamics*, J. Comp. Chem., 23, 673-684, 2002.
26. P. Gibbon and G. Sutmann, *Long-range interactions in many particle simulation*, in: Quantum simulations of many-body systems: from theory to algorithm. Eds. J.Grotendorst, D.Marx and A.Muramatsu. NIC-series Vol.10, pp. 467-506 (2002).
27. V. Stegailov and G. Sutmann (unpublished results).
28. L. Verlet, *Computer experiments on classical fluids. Thermodynamical properties of Lennard-Jones molecules*, Phys. Rev., 159, 98, 1967.
29. M. Tuckerman, B.J. Berne and G.J. Martyna, *Reversible Multiple Time Scale Molecular Dynamics*, J. Chem. Phys., 97, 1990, 1992.
30. *Molecular Dynamics on Parallel Computers,* Eds. R. Esser, P. Grassberger, J. Grotendorst, M. Lewerenz. World Scientific, Singapore, 2000.
31. G.M. Amdahl. *Validity of the single-processor approach to achieving large scale computing capabilities.* In: AFIPS Conference Proceedings, Vol. 30, pp. 483-485, Reston, Va., 1967. AFIPS Press.
32. A. Grama, A. Gupta and V. Kumar, *Isoefficiency Function: A Scalability Metric for Parallel Algorithms and Architectures*, in: IEEE Parallel and Distributed Technology, Special Issue on Parallel and Distributed Systems: From Theory to Practice, Vol. 1, pp. 12-21, (1993).
33. J.L. Gustafson, *Revaluating Amdahl's law*, in: Communications of the ACM, Vol. 31, ACM press, New York, pp. 532-533, (1988).
34. A. Grama, A. Gupta, E.-H. Han and V. Kumar, *Parallel Algorithm Scalability Issues in Petaflops Architectures.* Ultrascale Computing, 2000.
Available at: http://www-users.cs.umn.edu/~kumar/papers/papers.html#cccc
35. S. Plimpton, *Fast parallel algorithms for short range molecular dynamics*, J. Comp. Phys., 117, 1, 1995.
36. N. Attig, M. Lewerenz, G. Sutmann and R. Vogelsang, *Parallel Algorithms for Massively Parallel Computers*, in: Molecular Dynamics on Parallel Computers, Eds. R. Esser, P. Grassberger, J. Grotendorst, M. Lewerenz, World Scientific, Singapore, 46-69, 2000.
37. R. Murty and D. Okunbor, *Efficient parallel algorithms for molecular dynamics simulations*, Parall. Comp., 25, 217-230, 1999.