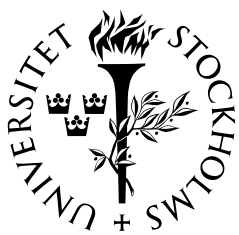
Algorithms for Molecular Dynamics Simulations

Algorithms for Molecular Dynamics Simulations

Advancing the Computational Horizon

FREDRIK HEDMAN



Avdelningen för fysikalisk kemi
Arrheniuslaboratoriet
Stockholms Universitet
Stockholm, 2006

Doktorsavhandling
Avdelningen för fysikalisk kemi
Arrheniuslaboratoriet
Stockholms Universitet

ISBN 91-7155-277-4, pp i–93
Typeset with LaTeX 2_ε.

Tryck: Universitetsservice US-AB, Stockholm

Till Karin, Johanna och Oskar.

Contents

Abstract

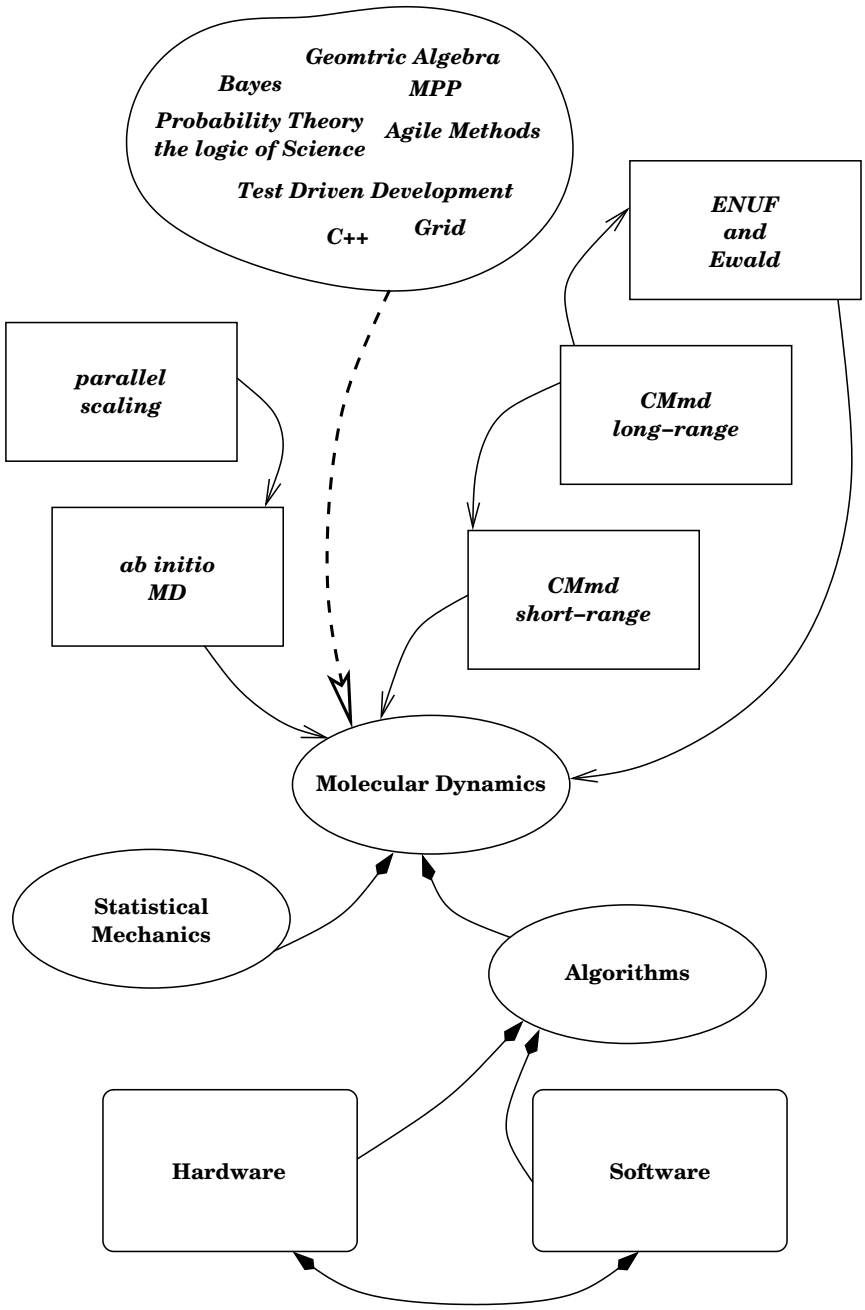
Methods for performing large-scale parallel Molecular Dynamics(MD) simulations are investigated. A perspective on the field of parallel MD simulations is given. Hardware and software aspects are characterized and the interplay between the two is briefly discussed.

A method for performing *ab initio* MD is described; the method essentially recomputes the interaction potential at each time-step. It has been tested on a system of liquid water by comparing results with other simulation methods and experimental results. Different strategies for parallelization are explored.

Furthermore, data-parallel methods for short-range and long-range interactions on massively parallel platforms are described and compared.

Next, a method for treating electrostatic interactions in MD simulations is developed. It combines the traditional Ewald summation technique with the nonuniform Fast Fourier transform—ENUF for short. The method scales as $\mathcal{O}(N \log N)$, where N is the number of charges in the system. ENUF has a behavior very similar to Ewald summation and can be easily and efficiently implemented in existing simulation programs.

Finally, an outlook is given and some directions for further developments are suggested.


Geomtric Algebra
Bayes
MPP
Probability Theory
the logic of Science
Agile Methods
Test Driven Development
C++
Grid
ENUF
and
Ewald
parallel
scaling
CMmd
long–range
ab initio
MD
CMmd
short–range
Molecular Dynamics
Statistical
Mechanics
Algorithms
Hardware
Software

List of Publications

This thesis is based on

1. Fredrik Hedman and Aatto Laaksonen. A parallel quantum mechanical MD simulation of liquids. *Mol. Simul.*, 20(5):265–284, 1998. Reprinted with permission from *Taylor and Francis Group*©(1998)[1].

2. Fredrik Hedman and Aatto Laaksonen. Parallel aspects of quantum molecular dynamics simulations of liquids. *Comput. Phys. Commun.*, 128(1–2):284–294, 2000. Reprinted with permission from *Elsevier*©(2000).

3. Fredrik Hedman and Aatto Laaksonen. Data parallel large-scale molecular dynamics for liquids. *Int. J. of Quantum Chem.*, 46(1):27–38, 1993. Reprinted with permission from *John Wiley & Sons*©(1993).

4. Fredrik Hedman and Aatto Laaksonen. A data-parallel molecular dynamics method for liquids with Coulombic interactions. *Mol. Simul.*, 14(4–5):235–244, 1995. Reprinted with permission from *Taylor and Francis Group*©(1995)[1].

5. Fredrik Hedman and Aatto Laaksonen. Ewald summation based on nonuniform fast Fourier transform. Article in press *Chemical Physics Letters*, 2006, DOI. Reprinted with permission from *Elsevier*©(2006).

[1] www.tandf.co.uk

Related Publications

Related works that have not been included in the thesis

[1] Fredrik Hedman and Aatto Laaksonen. An approach to data parallel molecular dynamics for liquids. *Int. J. Mod. Phys. C*, 4(1):41–48, 1993.

[2] Fredrik Hedman and Aatto Laaksonen. An approach to data parallel molecular dynamics for liquids. In Th. Lippert, K. Schilling, and P. Ueberholz, editors, *Science on the Connection Machine*, pages 41–48. World Scientific, 1992. (Proceedings of the First European CM Users Meeting, June 16–17).

[3] Erik Wallin, Christian Wettergren, Fredrik Hedman, and Gunnar von Heijne. Fast Needleman–Wunsch scanning of sequence databanks on a massively parallel computer. *CABIOS*, 9(1):117–118, 1993.

[4] Erik Aurell, Peter Frick, Fredrik Hedman, and Vladimir Shaidurov. Implementation of hierarchical model of 2D-turbulence on a CM200. Technical report, Center for Parallel Computers, KTH, 1993.

[5] Fredrik Hedman and Aatto Laaksonen. An embarrassingly parallel *ab initio* MD method for liquids. In Bo Kågström, Erik Elmroth, Jack Dongarra, and Jerzy Waśniewski, editors, *Applied Parallel Computing, 4th International Workshop, PARA'98*, volume 1541 of *Lecture Notes in Computer Science*, pages 224–229. Springer-Verlag, June 1998. 41

[6] Fredrik Hedman and Aatto Laaksonen. Quantum mechanical parallel MD simulation of liquid water. In Adrian Tentner, editor, *High Performance Computing Symposium — HPC'99*, pages 31–36. The Society for Computer Simulation International (SCS), 1999. Proceedings of HPC'99, San Diego, California, April 1999.

[7] Fredrik Hedman and Aatto Laaksonen. Large scale parallel molecular dynamics simulations. volume 7 of *Theoretical and Computational Chemistry*, chapter 7, pages 231–280. Elsevier Science Publishers, 1999. 11

[8] Ulf Andersson and Fredrik Hedman. Performance of an IBM pwr4 node for the GEMS TD codes and Parallacs. In Juha Fagerholm et al., editors, *Applied Parallel Computing, PARA 2002*, volume 2367 of *Lecture Notes in Computer Science*, pages 467–475. Springer-Verlag, June 2002.

Förord

Jag vill här ta tillfället i akt och tacka dem som på olika sätt hjälpt och influerat mig i arbetet med denna avhandling. Först och främst vill jag tacka båda mina handledare Aatto Laaksonen och Björn Engquist som under alla år varit ett avgörande och inspirerande stöd. Under de år jag arbetade på Parallelldatorcentrum (PDC) var Björn min huvudhandledare. När jag senare fick möjlighet att flytta över till Arrheniuslaboratoriet tog Aatto över denna roll. Tack vare ett generöst stöd från både Björn och Aatto har jag haft förmånen att kunna kombinera och dra nytta av miljöerna på PDC och Arrheniuslaboratoriet. Detta har för mig varit mycket positiv och berikande.

På PDC har jag haft stor hjälp av Per Öster, Nils Smeds, Lars Malinowsky, Mike Hammill, Per Ekman och Johan Danielsson. På Arrheniuslaboratoriet har jag haft många och intressanta diskutioner med Martin Dahlberg och Patric Schyman. Ett tack också till Jozef Kowalewski för att du bidrog till att ge mig möjligheten att komma till avdelningen för fysikalisk kemi.

Ett speciellt tack till Magnus Forsberg då du som vikare för matematik och fysik på Engelbrektskolan i Borås, tog dig tid att diskutera och dessutom fixade fram läroböcker i matematik.

Ett stor tack till Martin Dahlberg och Per Hammarlund som har läst och givit värdefulla kommentarer till avhandlingens olika delar.

Jag vill också tacka mina föräldrar, Marianne och Bengt Hedman, för uppmuntran och för att ni påminde mig om att våga välja *och* stå fast vid det som kändes rätt.

Avslutningsvis vill jag tacka Johanna och Oskar för att ni fördragit en något disträ pappa. Sist men inte minst, till min älskade Karin för all uppmuntran och stöd och för att du hjälpt mig att inte ge upp hoppet om ett slutdatum för avhandlingen. *Nu* är den klar ☺☺☺

Reimersholme, maj 2006
Fredrik Hedman

[Statistical mechanics provides] the methods that must be employed when we wish to predict the behaviour of a mechanical system on the basis of less knowledge as to its actual state then would in principle be allowable or possible. Such partial knowledge of state is in reality all that we ever do have, and the discipline of statistical mechanics must always remain necessary.

Richard C. Tolman
The Principles of Statistical Mechanics

Chapter 1

A Perspective on Molecular Dynamics

Molecular Dynamics (MD) models properties of a system of interacting particles by repeatedly calculating the interactions between the particles and integrating their equations of motion. This process traces out a discrete phase-space trajectory. Combining statistical mechanics and kinetic theory, microscopic properties of the system can be calculated. Sampling and averaging these properties of the system along a sufficiently long trajectory approximates the corresponding macroscopic properties.

There are three main scenarios for the use of MD. In the first scenario the simulated properties are compared with experimental results, and when the two agree it is reasonable to claim that the experimental results can be explained by the simulation model. In the second scenario, MD simulations are used to interpret experimental results. In a sense the second scenario is the inverse of the first. In the third scenario, simulations are used as an exploratory tool to help gain an initial understanding of a problem and give guidance among possible lines of investigation, be it theoretical or experimental. In all these scenarios it is often the case that a larger simulation is a more realistic model.

Molecular Dynamics simulation is today an increasingly common approach for performing many-body calculations on the condensed states of matter. Since the first proof-of-concept simulations, almost five decades ago, it has, over the last thirty years, become an established area of science and is continuously developing with the aid of improved physical models, more efficient algorithms and faster computers. Simulations can always be extended to cover longer time periods and the modeled systems can always be made larger than the largest systems studied so far. Furthermore, the models can also be made more accurate and brought closer to the fundamental physics.

The interactions calculated during each time-step are all independent, and once the total interaction on each particle is known the new position can be found inde-

pendently of all the other particles. In principle, this means that the calculations performed during each time-step could all be done in parallel, but achieving this in practice, while maintaining efficiency and generality, remains a considerable challenge. Consequently, there is a strong motivation for developing more accurate and efficient methods for performing MD simulations that also has favorable parallel properties. The focus of this thesis is the investigation of a few of the possible avenues of method development: MD using interactions calculated via first-principles, parallel methods for short-range interactions and a novel mentod for long-range electrostatic interactions. To set the stage for presenting the results I give a historical background followed by a concise description of the MD method and some aspects of how software and hardware influence efficient implementation for large-scale problems. Finally, I complete this Chapter by discussing the scope of the thesis.

1.1 Historical and Scientific Backdrop

SYSTEMS of interacting particles is a general modeling paradigm that can be applied to a wide variety of problems [1]; current examples cover the range from galaxy- and star-formation to molecular phenomena. Many times it has been observed that the time evolution of particle systems, the N-body problem, is well worth studying. Often the more particles involved in the model, the more realistic the model becomes. Below I give two examples of the historical background.

The gang of four: Tycho, Kepler, Galilei and Newton The earliest particle models of celestial phenomena predates Newtonian Mechanics and also provided one of the early successful applications of Newtonian Mechanics, *i.e.*, Kepler's laws of planetary motion. This was to a large part made possible by the accumulated work of Tycho Brahe, Johannes Kepler, Galileo Galilei and Isaac Newton and span several centuries, but still makes a good example of how experiment, calculation and theory interplay.

It is rather surprising, but we have a very exact description of how it all started [2]. On the evening of November 11, 1572 the sky was clear. A young Danish nobleman, Tycho Brahe (1546–1601), was returning home for supper from his alchemical laboratory. He observed an unfamiliar starlike object in the sky, much brighter than Sirius, Vega and even Venus. This observation was to become decisive for the young man's life. Using his own home-built and much improved sextant, Tycho Brahe was able to show that the new star did not move relative to the other fixed stars. This was against all established religious dogma and scientific wisdom of the time – a new object among the fixed stars! Because of this extraordinary discovery, he soon became famous throughout Europe and was given the title of the Royal Danish Astronomer. For financial support he also received the island Ven, between Denmark and Sweden. On Ven he built Uranienborg ("the castle of the heavens") and dedicated it to accurate astronomical studies. During

a period of over twenty years, Tycho Brahe and his numerous assistants collected an extensive amount of precise astronomical observations. However, in 1597, he was forced to leave Ven. Tycho Brahe and his entourage finally settled in Prague, where he received an appointment as the Imperial Mathematicus. While in Prague, he invited Johannes Kepler (1571–1630) to join his group. Kepler eagerly accepted the invitation. Unfortunately, their collaboration did not last more than about a year, because Tycho Brahe died unexpectedly in 1601. Kepler took well care of Brahe's extensive and detailed observations. In a letter, dated 1605 Kepler wrote *"I confess that when Tycho died, I quickly took advantage of the absence, or lack of circumspection, of the heirs, by taking the observations under my care, or perhaps usurping them ..."* (from page 280 of [3]). Or to put it more bluntly, he purloined Tycho's observations.

Nevertheless, Kepler used these observations in a very clever way – in the introduction of Carola Baumgardt's "Life of Kepler" [4], Albert Einstein uses the expression "an idea of true genius" to describe Kepler's work to formulate laws of planetary motion. After many years of hard work, as well as keeping out of the way of Tycho Brahe's heirs who wanted the observations returned, Kepler made his results public in "Astronomia Nova" in 1609. During the same period, Galileo Galilei (1564–1642) carried out systematic experiments with moving objects [5, 6], and was able to formulate the laws for velocity and acceleration. He later published them as "Two New Sciences" (1638). Finally, Isaac Newton (1642–1727), who built on, combined, and greatly generalized the work of Galilei and Kepler, was instrumental in creating a working scientific method firmly grounded in mathematics. Newton tested his own ideas by rederiving the laws of Kepler, while Kepler had deduced his three laws from Tycho's observational data. So in fact, at the very foundation of Newtonian Mechanics we find this very fruitful relationship between observation, calculation and theory. In the case of Tycho Brahe, Johannes Kepler and Isaac Newton, using a modern vocabulary, it was Kepler who did the work of a "computer", while Tycho Brahe provided the experimental data and Galilei and Newton supplied the theoretical and mathematical models.

Laplace's vision *"Given for one instant an intelligence which could comprehend all the forces by which nature is animated and the respective situation of the beings who compose it – and intelligence sufficiently vast to submit these data to analysis – it would embrace in the same formula the movements of the greatest bodies of the universe and those of the lightest atoms; for it, nothing would be uncertain and the future, as the past, would be present to its eyes."* [7] In this magnificent statement from 1814, Laplace is basically stating that the solution to the N-body problem is the "answer to everything",[1] provided that all the necessary calculations could be performed.

[1] Depending on the context it is either 4711 [8] or 42 [9];)

Molecular Dynamics enabled by the computer From the above quote of Laplace, it is clear that an embryonic idea, similar to Molecular Dynamics, had been around for about 150 years, when the first published simulations results using MD appeared in 1957 in conjunction with a study of the same system using the method of Monte Carlo (MC)[10, 11].

The enabling technology, the "intelligence sufficiently vast to submit these data to analysis", was the electronic computer. It was becoming available to a wider group of scientists after World War II, and one of the earliest problem domains addressed was computer simulations of particle systems [12, 13]. The computer simulation approach to statistical mechanics via MD and MC was first met with some skepticism [14], and it was not until the mid 1960's when the first simulation of a liquid system was published [15] that it started to gain a wider acceptance. One of the systems used in this work contained 864 particles and each time-step took about 45 seconds using floating point arithmetic on a CDC-3600.[2] The simplicity and elegance of the method coupled with a drive to perform more extensive simulations sparked an interest in developing improved algorithms. An early improvement was the neighbor-list method as well as the rediscovery of the Störmer-Verlet method for time integration [16]. With larger and more complicated systems, like water[17] and ionic salts [18], there was also steady progress in method development for both short-range interactions [19, 20, 21] and long-range interactions [20].

Today we know that Laplace's vision was too magnificent, but all the same, numerically solving the N-body problem on fast computational resources is certainly one of the most widely applicable scientific models. Since the mid 1970's, the exponential increase in computer capability has had a profound influence on how scientific problems can be addressed. The scientific method of interplay between experiment and theory is now complemented by computational science. Molecular Dynamics is a typical example of this.

1.2 Molecular Dynamics in a Nutshell

MOLECULAR Dynamics simulation is one of a growing number of methods to study the macroscopic behavior of systems by following the evolution of the system at the molecular scale. One way of categorizing these methods is by the degree of determinism used in generating molecular positions [22]. On the scale from the completely stochastic method of Metropolis Monte Carlo to the pure deterministic method of Molecular Dynamics, we find a multitude of methods; to name just a few examples: Force-Biased Monte Carlo, Brownian Dynamics, General Langevin Dynamics [23], Dissipative Particle Dynamics [24, 25], Collisional Dynamics [26] and Reduced Variable Molecular Dynamics [27]. I give a sketch of the method of Molecular Dynamics by describing how integrators, force fields and boundary conditions are combined with the classical equations of motion. More

[2] CDC—Control Data Corporation, started in 1957. Seymour R. Cray was one of the founders of CDC. In 1972 he left CDC to start Cray Research Inc.

details can be found in a number of excellent textbooks that describe the theory and practice of Molecular Dynamics [28, 29, 30, 31, 32, 33]. For understanding integrators and how they can be adapted for other ensembles I have found [34] outstanding.

1.2.1 The MD method

Let $\mathbf{r}^N$ denote the set of vectors that locate each center of mass of each of the atoms in a system, $\mathbf{r}^N = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$. The particles in the system each have d degrees of freedom. Assume that the N point particles interact through a continuous pair potential, $V(\mathbf{r}^N)$. This pair potential is a model of the interaction between two particles. Let the mass of each particle i be m_i, and let $\mathbf{F}_i$ be the total force acting on particle i at time t. Newton's equation of motion for each particle, $i = 1, \ldots, N$, can then be written as

$$m_i \frac{d^2\mathbf{r}_i}{dt^2} \equiv m_i \ddot{\mathbf{r}}_i = \mathbf{F}_i. \tag{1.1}$$

With the assumption of pairwise additive, conservative inter-atomic forces, that are only a function of the pair separation, the force that particle j exerts on particle i is

$$\mathbf{f}_{ij} = -\nabla_i V(r_{ij}), \tag{1.2}$$

where $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ is the pair separation. The total potential energy of the system is a sum over all pairs and the total force acting on each particle i is found by summing over all pairwise interactions

$$E_p = \frac{1}{2} \sum_{i,j} V(r_{ij}), \tag{1.3}$$

$$\mathbf{F}_i = \sum_{j=1}^{N} \mathbf{f}_{ij}, (j \neq i). \tag{1.4}$$

It should be noted that because of Newton's third law $\mathbf{f}_{ij} = -\mathbf{f}_{ij}$, each pair interaction has to be calculated only once, but of course must still be summed into both $\mathbf{F}_i$ and $\mathbf{F}_j$ with opposite signs.

In equilibrium NVE Molecular Dynamics simulation new molecular positions are obtained by solving Newton's equation of motion numerically. In solving Equation (1.1) use Equation (1.4) and also specify the initial and boundary conditions of our d dimensional system. This results in a set of $d \times N$ coupled second-order ordinary differential equations and a total of $d \times N$ degrees of freedom.[3] The equations are discretized and new positions and velocities for each atom is found numerically by integrating forward in time:

[3] With total momentum conserved, subtract d.

1. Specify the initial conditions (N, initial temperature, boundary conditions, model potentials, molecular connectivity, time-step, density, ...).
2. Construct initial structure of the system and give initial velocities to the particles.
3. For each time-step of the simulation
   a) Compute all forces, energies and optional properties (temperature, pressure, ...).
   b) Integrate equations of motion.
   c) Sample system properties at regular intervals.
4. Compute averages of system properties.

In equilibrium MD an isolated system of fixed volume V and a fixed number of molecules N is studied. Because the system is isolated its total energy E is constant and thus the variables N, V and E determine the thermodynamic state. The molecules of the system interact through model potentials. The positions of the molecules are obtained by solving the equations of motion for each molecule. Usually the dimension $d = 3$, but $d = 2$ is also common, and there are even examples of four dimensional simulations [35]. From the solution one gets the positions and velocities of each particle as a function of time. By applying kinetic theory, statistical mechanics and time averaging over these particle trajectories, macroscopic properties can be computed from microscopic variables. In Molecular Dynamics, these properties can represent both static properties, like temperature, pressure and pair distribution functions, and dynamics properties, such as transport coefficients.

After completing the simulation the trajectory is analyzed to produce the simulation results. It is assumed that by averaging over a sufficient number of time-steps these time averages become approximate measures of the corresponding NVE ensemble averages.

In the broader view, there are also other auxiliary tasks that may be included as shown in the Figure on the facing page. To start a simulation of a complicated system can be quite tricky, but is usually not computationally expensive. Analyzing and visualizing massive amounts of data can be a challenge, and just as computationally expensive as producing the simulation itself. The visualization process can also be parallelized.

An other addition that is becoming more and more important is the possibility to get access to large computational resourseces via the network—the Grid. These efforts are large topics in their own right, outside the scope of this work. In passing, I note that there are several interesting possibilities under development that is being explored; a combination of high-speed networks, for sharing simulation results stored in large digital libraries, distributed scheduling, advanced steering and visualization techniques, such as immersive virtual reality, to analyze and understand simulation results [36, 37, 38, 39].

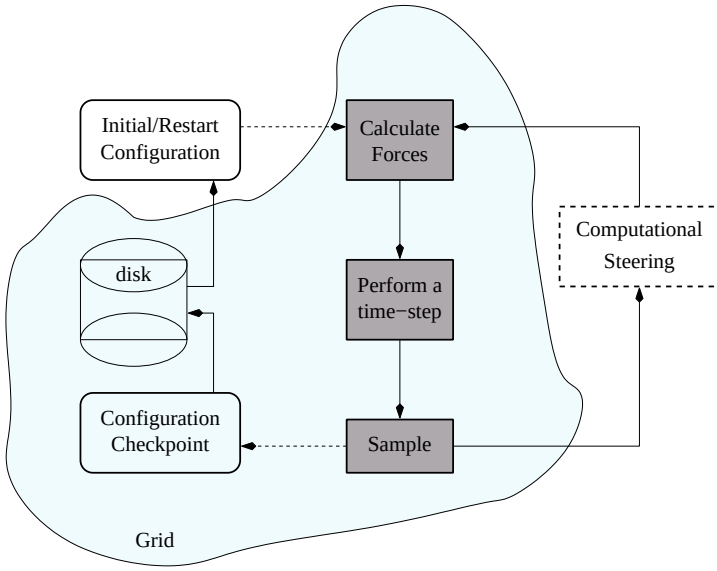


Figure 1.1. The initial configuration can be created on the fly or from a restart file. Complex systems may have to be assembled piece by piece from several specifications. The Grid is represented by the "cloud" in the background.

1.2.2 Boundary conditions

Boundary conditions in simulations with the objective to study equilibrium properties of a bulk fluid should be chosen so as to minimize the finite-size effects and boundary effects. One possible approach to this is to replicate the computational box and use periodic boundary conditions [40], thereby making the simulated system pseudo-infinite. The chosen computational box should be space-filling and it is replicated throughout space in all directions. While there are several different space-filling shapes [41] the cubic box is the simplest and most commonly used.

Periodic boundary conditions result in each particle having an infinite number of interacting neighbors. Express the total potential energy of a periodic system in a cubic box with side length L as

$$V_{\text{tot}} = \frac{1}{2} \sum_{\mathbf{n}} {\sum_{i,j}}' v_{\text{eff}}(\mathbf{r}_{ij} + \mathbf{n}L), \tag{1.5}$$

where the factor $1/2$ makes sure that each pair is counted only once, $\mathbf{n}$ is a vector of integers and the prime over the second sum is a remainder that for $\mathbf{n} = 0$ the term $i = j$ should be not be counted. For short-range interactions it is reasonable to make an approximation and restrict the number of interactions through the application of a cut-off, r_c, around each particle. This can be well motivated by

shielding effects. For long-range interactions, the periodic images must be taken into account. In terms of Equation (1.5) it means that for the short-range part we only need to include the $\mathbf{n} = 0$ term, However note that for reproducible results it is crucial that the cut-off is made smooth [42]. The cut-off, r_c, defines a spherical neighborhood around each particle and for consistency with the minimum image convention it should fit inside the computational box. In general, if L is the shortest edge of the simulation box, $r_c < L/2$.

The particles in the central computational box is surrounded by image particles residing in each of the periodic replicas of the central box. The image particles move in exactly the same way as the particles in the central computational box. The periodic boundary conditions are implemented so that when a particle moves out of the central computational box during the course of the simulation, its periodic image reappears at the opposite side of the central computational box.

The inter-particle distance used in the simulation is calculated using the "minimum image" convention. It assumes that the size of the simulation box $L > 2r_c$. This ensures that all particles j within the interaction range of particle i are uniquely determined. The distance between two particles i and j is the smallest of all the possible distances between particle i and j, including all the replica images of particle j. Take the cubic box with edge length L centered at the origin as an example. This restricts the Cartesian coordinates in each dimension to lie in the interval $[-L/2, L/2[$ and consequently the difference in each coordinate value, $\Delta_{ij,\alpha}, \alpha = \{x, y, z\}$, is in the interval $[0, L[$. We find the minimum image distance in each coordinate direction by taking the smallest absolute value from the three possible minimum values $\{\Delta_{ij,\alpha} - L, \Delta_{ij,\alpha}, \Delta_{ij,\alpha} + L\}, \alpha = \{x, y, z\}$.

A different computational box also means that a different minimum image distance criteria must be used. For example the truncated octahedron which has a reasonably simple minimum image transformation [41, 43, 44]. A further refinement by Bekker [45, 46] develops an interesting simulation box transformation method: a MD simulation formulated for example in a truncated octahedron box can be transformed into triclinic simulation box via a preprocessing phase. This procedure implies that every Molecular Dynamics may be done in the same type of box.

Using the minimum image distance criteria ensures that the distance between two particles varies continuously as particles move out of the central computational box and reappears at the opposite side. Furthermore, the periodic boundary conditions have the effect of restraining unphysical density fluctuations. However, it also means that particles in the central computational box will never be more than half the box length L apart and phenomena with a characteristic length-scale longer than L become suppressed [47, 48].

1.2.3 Force fields in atomic systems

The interactions between atoms are the most fundamental input of MD simulations. From a physical point of view all the important contributions to the forces originate from electronic interactions between the nuclei and the electron clouds of the

atoms [49, 50, 51]. Some of these contributions are classical, like the Coulomb interaction, while others, like dispersion, require a quantum mechanical explanation. These model interactions have both attractive and repulsive parts and are often non-additive. In general the total potential energy of the system is often written as a sum of n-body terms, $n = 1, 2, 3, \ldots$

$$V(r_1, r_2, \ldots, r_N) = \sum_i v_1(r_i) + \sum_{i,j} v_2(r_i, r_j) + \sum_{i,j,k} v_3(r_i, r_j, r_k) + \ldots . \quad (1.6)$$

The first term on the right hand side represents the effect of an external field on the system. Examples are external, magnetic or electric fields or fields which model container walls. The second term is the so called pair potential. It is summed over all distinct pairs of particles. The third term is the three-body potential and should be summed over all distinct triplets. Higher order terms are expected to be small compared to the two-body and three-body terms and are consequently neglected.

In the vast majority of MD applications a further simplification is made by using effective and pair-wise additive potentials for atomic interactions. In simulations which contain flexible molecules, it is common practice to add terms which represent chemical bonds, bond angles, improper torsions and dihedrals. Interactions between atoms of molecules are represented by effective additive potentials. This empirical approach splits the total potential energy of the system into a bonded (inter-molecule) and non-bonded (intra-molecular) part.

For an isolated system there are no external influences, so the first term of Equation (1.6) is zero. The effective pair potential is a function of the pair separation $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ and is constructed in such a way as to include the true pair potential and average effects of higher order terms; it often includes also electrostatic and dipolar effects. The total potential energy of the system is then a sum over all distinct pairs of particles

$$V_{\text{eff}} = \frac{1}{2} \sum_{i,j} v_{\text{eff}}(r_{ij}). \quad (1.7)$$

Interactions can be described using different attributes, but from a computational point of view the most important dividing line is between long-range and short-range forces. The prototypical short-range interaction is the ubiquitous 12-6 Lennard-Jones potential. The Coulomb interaction is an example of a long-range interaction.

In MD it is common to describe an interaction as short-range when its potential decays more rapidly than r^{-d}, where d is the dimensionality of the system. The reason being that the approximate contribution to the total potential energy of all particles outside the cut-off is well defined. For example, in 3D this contribution is proportional to $\int_{r_c}^{\infty} V(r) 4\pi r^2 dr$, which is well defined when $V(r)$ decays more rapidly than r^{-3}.

In calculating long-range interactions it is quite tempting to apply a simple cut-off in the same way as in the case of short-range interactions, but this can be ruled

out since it creates unphysical effects at the boundary of the cut-off sphere. For some systems it seems that more advanced cut-off methods based on charge grouping may constitute an acceptable solution [52]. However, there is a growing number of cases which clearly show that long-range interactions, when present, are very important and must be given careful consideration, especially in ionic systems [53, 54, 55, 56]. Consequently, a number of alternative approaches have been developed. Examples are lattice sum methods [57, 58, 59, 60, 61], reaction field methods [62], cut-off methods [63, 64, 65, 66], the isotropic periodic sum method [66], and hierarchical methods [67] such as multigrid [68, 69, 70, 71, 72] and fast multipole methods[73]. Further approaches can be found in the excellent review [74].

The essential difference between these approaches is the way in which the interactions beyond the cut-off are represented. In the cut-off methods, interactions are set to zero for distances longer than the cut-off used, possibly with a constant long-range correction applied. In the reaction field method, a continous dielectric medium with a given dielectric constant, is assumed beyond the cut-off; and the long-range interactions are replaced with reaction field interactions. Replicating the simulation box in all directions, using periodic boundary conditions (PBC), and by summing over all the images when calculating long-range interactions, leads to the lattice sum methods. Furthermore, these alternatives give different trade-offs when it comes to computational cost and accuracy: the cut-off methods is the least costly, but often introduces serious artifacts; the reaction field methods can be viewed as a compromise between computational cost and accuracy and finally the lattice sum methods are the most costly, but also widely considered the most accurate.

The most prominent of the lattice sum method is the Ewald summation method. It has its origin in condensed matter physics [57, 75] and is reckoned to be the reference method for electrostatic interactions in systems modeled with periodic boundary conditions. Even though the Ewald method avoids the truncation of long-range interactions and thus removes the simulation artifacts caused by truncation methods, it should be noted that there are still subtle, system dependent, issues with how the periodic boundary conditions affect a simulation[66, 76].

1.3 Software and Hardware Interplay

Now that commonly available parallel hardware and software platforms have evolved to a sufficiently stable and mature state, their combination and application to large-scale Molecular Dynamics provide both new challenges and new opportunities. This is further underscored by the comparatively affordable parallel platforms available through clustering of commodity processors and high-speed networks. At first sight it may seem that parallel algorithms for large scale Molecular Dynamics simulations would be straightforward to design and implement, simply because both the underlying physical phenomena as well as the MD method itself can be described as naturally parallel. However, the large number and variety of

published reviews of parallel MD algorithms from the mid 1980's and onward – a *small* selection is given by [77, 78, 79, 80, 81, 82, 83, 84] – is an indication that there are many different parallelization aspects that can be taken into account when designing and implementing parallel software to perform MD simulations.

1.3.1 General software aspects

Finding a good design often requires several iterations. This software effort can be more or less supported by the programming environment used and it is undoubtedly of considerable help if encapsulation and code reuse has a strong support in the programming language used. When designing and writing efficient programs for large scale parallel MD it is clear that hardware specific details can not be ignored. Incorporating hardware details specializes the program. In the worst case it becomes so specialized that it can not be reused when the target platform becomes outdated. This is a waste of human effort and in the long run simply not acceptable. The challenge is to design programs so that optimization can be applied with minimal loss of versatility, generality and portability.

MD is a typical exponent of the 90/10 rule which states that a program spends 90 percent of its cycles in 10 percent of its code [85]. In MD, these 10 percent of code consist of the force calculation and time integration parts. These parts of the algorithm are consequently the primary target when optimizing.

Not so long ago, there was a great variety in parallel computer architectures [86]. A large number of experimental parallel computer architectures were proposed and built. A number of these became commercial products, but the variations in architecture between vendors and also between generations from a particular vendor was substantial. In the case of large scale Molecular Dynamics simulations, the hardware used has evolved from mainframes and vector-based supercomputers to parallel computers of different designs. Today, high performance computing (HPC) for large scale MD is synonymous with parallel computing. Until the late 1980's, the software development was very much curtailed by the limitations of hardware, where the size of available computer memory was the most critical factor. At present, the most critical aspect for large scale parallel MD simulation is not the hardware, but the software. Coping with increased problem and algorithmic complexity, as well as varying hardware platforms is a daunting task. Adding the requirement of optimal use of hardware resources makes the development or modification of an efficient and portable parallel MD simulation software a formidable challenge.

To meet this challenge users should first ask themselves what kind of problems they intend to solve. Often a combination of improved software and hardware capabilities means that a PC may be good enough for routine simulations. Special purpose hardware, found in two categories today, is also an alternative. The traditional form uses specially designed hardware and software, the more recent uses off-the-shelf hardware and *de facto* standard software.

Of the traditional special purpose approach, there are several successful projects reported in the literature, out of which [87, 88, 89] is a small selection. At the

moment, this approach seems to entail too long development times and also require special software development. However, with sufficient economic incentive and scientific need it may still become a viable alternative [90].

With arrangements like a PC cluster, running Linux and using software like MPI or PVM, we have an alternative special purpose approach that is essentially software based [91, 92, 93]. There are two key reasons why this kind of approach has become so prevalent: first a PC cluster requires modest hardware investments and can be expected to be run in dedicated mode; secondly, since the used software is *de facto* standard, programs can be expected to have a longer lifetime and the methods and algorithms used in the programs can be gradually evolved for best performance.

For optimal use of the available resources, an appropriate computational resource to solve a particular problem should be applied. See Figure 1.2. There is a distinct difference in the services delivered by resources aimed at maximal throughput and resources aimed at maximal speed. Specifically, this means that, in the best of worlds, the many simulations that can run on a PC should use this type of resource and supercomputers should be reserved for truly large scale simulations.

At the high end of the hardware simulation spectrum considerable human effort is needed and can be motivated by the intrinsic scientific or technical nature of the problem. The software developed should have a long useful lifetime and overall efficiency and speed should be of primary importance [94]. However, the list of promising parallel computer vendors that are no longer in business grew long during the end of the 1990's. Despite this sad fact, software for programming parallel computers has made a number of important advances in the past decade. There is now a reasonable and growing basis of standard software tools and languages available that can be used to write efficient programs that can be expected to deliver adequate, but not outstanding, performance on a variety of platforms. Highly efficient codes still require extensive tuning which are processor and platform dependent.

Dealing with this situation requires an approach which separates the general from the specific. Software should be constructed so that hardware independent, general, parts are kept separate from the specific, hardware dependent parts. Valuable aids in this process are programming models which help make the distinction between the general and specific, putting strong emphasis on testing and incremental development [95, 96, 97]. Designing these types of models are large subjects in themselves [98, 99, 100]. In addition, performance models of computer systems and applications are valuable guides in understanding how a particular application performs on a specific platform. When these models are used together they can clarify what parts of an application are critical for performance and also help to show what computational resource to use [101].

1.3.2 Parallel computer models

There are many compelling reasons for having a parallel computer model made to order for parallel MD. The major motivation is that it gives a clear framework for rethinking old algorithms and formulating new ones. It can also help in building the performance models required and suggest which hardware features are most

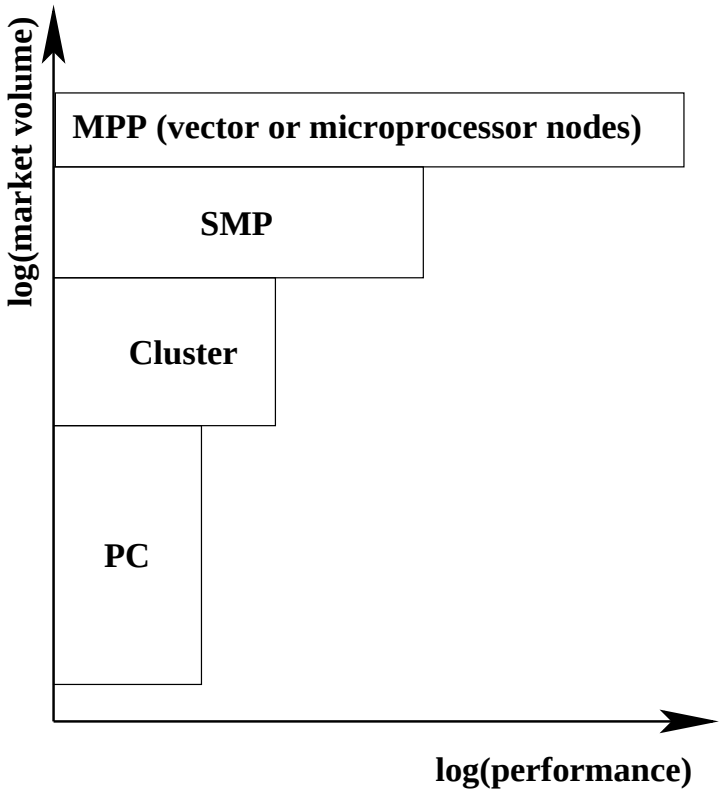


Figure 1.2. Bar-chart sketch of computer resource spectrum. For optimal use of the available resources, an appropriate computational resource to solve a problem should be applied.

important to efficiently solve the problem at hand. Finally, it may also give guidance for how implementations should be done.

When the Eckert-von Neumann computer model appeared, the concept of the stored program computer quickly became the most common way to organize and think about computers in commerce, industry, science and education [102]. The Eckert-von Neumann computer is composed of a memory and a central processing unit (CPU). See Figure 1.3. The memory holds both the program and the data. The CPU executes the program which consists of a sequence of instructions which specify memory addresses, arithmetic-logical operations or branch statements. Memory is assumed to be flat, meaning that there is little or no time difference in accessing different parts of the memory. This simple model has proven remarkably useful and is still the dominant hardware model, often more or less implicitly assumed in algorithm design and programming languages.

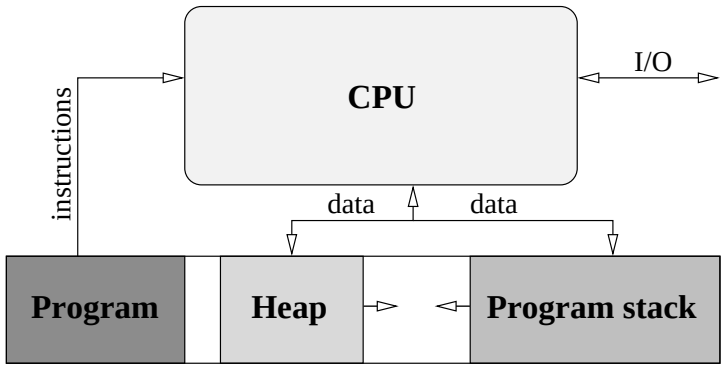


Figure 1.3. Model of Eckert-von Neumann computer. The Eckert-von Neumann computer is composed of a memory and a central processing unit (CPU). The memory holds both the program and the data. The CPU executes the program which consists of a sequence of instructions which specify memory addresses, arithmetic-logical operations or branch statements.

During the early part of the 1990's, there was a vivid debate between proponents of multiple instruction streams, multiple data streams (MIMD) and single instruction stream, multiple data streams (SIMD) parallel computers. This taxonomy, which stem from Flynn's classification [103, 104] of computer architectures, is now mostly of historical interest. From a current hardware point of view the instruction and data stream classification (SIMD, MIMD, ...) does not capture the most important aspects of parallel hardware architecture for scientific computing with large datasets. The vast majority of parallel computers today are clusters [91]. I assume the following characteristics of the parallel computer model (See Figure 1.4)

- The nodes in the cluster are single Eckert-von Neumann processors or scalable multiprocessors (SMPs).
- Nodes may be of varying processing power and have different amounts of memory.
- Nodes communicate via a network of some kind, which may be more or less visible to the programmer.
- The memories of the nodes are private, but a global addressing scheme may be available through a combination of hardware and software.
- The cost of sending a message between two processors is mostly a function of the size of the message and does not depend too much on the relative node locations and other network traffic. However, the start-up cost of sending messages can not be neglected.
- The node local memory is much faster to access than remote memory. This implies that local read and write operations take significantly less time than sending and receiving data from other nodes.

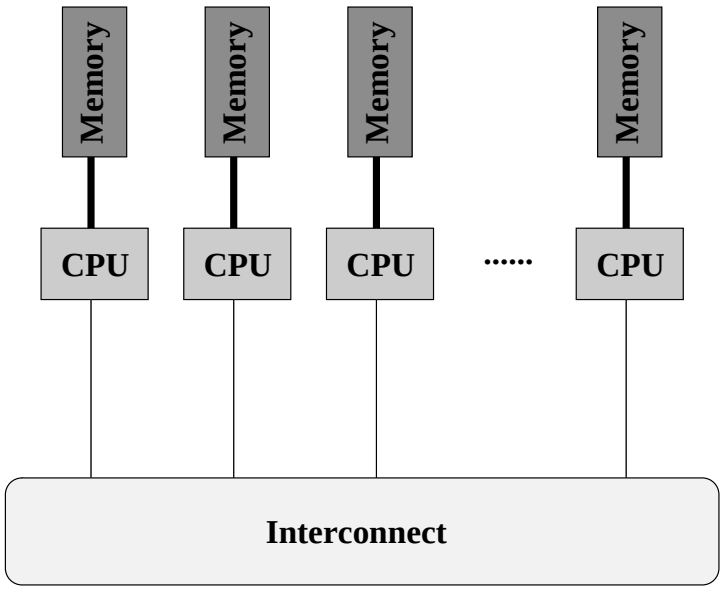


Figure 1.4. The multicomputer. The nodes in the cluster are Eckert-von Neumann or SMP processors. Nodes may be of varying processing power. Nodes communicate via a network of some kind. The cost of sending a message between two processors is mostly a function of the size of the message and does not depend too much on the relative node locations and other network traffic. The memories of the nodes are private. The node local memory is faster to access than remote memory.

The parallel computer model described above is often called a multicomputer. Further discussions of different machine models can be found in [98, 100, 105].

The multicomputer model decouples node-local activities from communication. This simplification decouples the performance critical aspects of parallel hardware: calculations and communications. In practice it treats a multi-processor node on the same footing as a node based on a single CPU microprocessor or a single vector-processor, thereby allowing programming practices developed for these types of nodes to be reused. Of course, there are still difficulties with how to construct efficient programs for each type of node, but these problems have, to a large part, already been addressed and also in some cases solved – at least in principle.

1.3.3 Programming models and languages

Parallel algorithms are designed with a parallel computer model in mind. In the context of this work, programming models are supposed to explain how a program will be executed while the parallel computer model is meant to be an abstraction of the hardware. The programming model acts as a bridge between algorithms and actual implementations in software.

There are a number of parallel programming models. The two most common are SPMD (single program multiple data) and FPMD (few programs multiple data).

Of these two, SPMD is the most widely used. In actual practice it means that the same program runs on all the nodes of the parallel computer, but the nodes will follow different paths through the program. These different paths are chosen based on conditions that may evaluate differently on each node. The conditions can be completely general, but often depend on the node identity or on local data computed at the node. With FPMD, the task at hand is solved by a set of cooperating, but different programs.

Closely related to SPMD is the data-parallel model. In this model the parallelism is extracted from the parallel operations that can be performed on arrays of data. Even though this programming model was first introduced on SIMD computers, it is a misconception that it is tied exclusively to these types of machines.

SPMD programs are most often written using a combination of one of the standard programming languages and message passing libraries. There are at least a half a dozen libraries around that can claim to do message passing, but today the two major ones are PVM [106] and MPI. The latter is a *de facto* standard that is well documented in [107, 108, 109, 110]. The two libraries have slightly different functionality, but at least for MD purposes they are largely interchangeable [111, 112].

The programming models that underlie Fortran [113] and C [114] are very similar. Both support data structures and encapsulation. C++ [115, 116] is for most practical situations a superset of C.[4] C++ supports several programming paradigms, including object oriented constructs. The object oriented programming paradigm takes a radically different view at how a program is organized [121]. It revolves around organization of objects which are encapsulations of data and the operations which can be performed on this data. To manage complexity it uses the concept of inheritance to help create abstractions through a hierarchy of objects. As experience has accumulated it has become clear that not all problems are best solved using only a single programming paradigm [95].

Choosing a particular programming language does in fact also shape the solution domain that is conveniently accessible. While it is possible to emulate object oriented constructs in Fortran 95 [122] I strongly believe that a suitable programming language for a generic and extensible MD program needs built-in support for these facilities. For new and more complex programming projects it is clear that C++ programs can be written with very high performance while still retaining portability [101, 123, 124, 125]. Even though the learning curve is rather steep, I believe that an object oriented software approach will become the rule rather than the exception also in scientific computing. So in summary, as a complement to the parallel computer model, the programming model and languages used go hand in hand to allow modularity, encapsulation and extensibility.

[4] With C89 this is true, but to a lesser degree with the new ISO standard C98 [117, 118]. However, with the revision of this standarad well under way [119], there is a clear goal to reconcile most of the differences. See [120] and references therein.

1.3.4 Performance models

When developing, parallelizing or porting an MD program, performance models can be of great help in understanding how a particular algorithm or implementation behaves and where the major performance bottlenecks are located in the code [126, 100].

The attained performance improvements, when solving a particular problem of size N on a parallel computer with P nodes are often quantified by the speedup. The *speedup*, $S(N, P)$, is defined as the quotient between the best sequential time and the time to solve the problem on P nodes

$$S(N, P) = \frac{T_1^*(N)}{T_P(N)}. \tag{1.8}$$

Often $T_1^*(N)$ can be difficult to obtain because a completely different algorithm should be used on a sequential computer than on a parallel computer, or the problem may be so large that it can not be solved on a single node. To still get a measure of speedup, the best possible sequential time, $T_1^*(N)$, can be replaced by $T_1(N)$. That is, the time it takes to run the parallel algorithm and problem of same size on a single node. This speedup is then called the *scaled speedup*. In theory the best possible speedup that can be obtained is linear, *i.e.*, using a factor f more nodes the execution time is is scaled by a factor $1/f$.

Notwithstanding, there are examples of super-linear speedup. These can most often be attributed to memory effects. Dividing a problem up into smaller pieces on several cooperating processors will make it more likely that a larger part of the active problem data will spend more of its time in a faster memory, compared to the case when the problem is solved on a single node. If the problem has been using disk as a temporary storage media the effects of being able to fit the whole problem into memory can be quite dramatic. For example, relative speedups of 527 when running on a 48 nodes parallel computer is reported in [127].

For parallel computers the performance model is a function of the parallel computer model. A general observation that is always important to have in mind, is Amdahl's law [128]. It can be stated in a number of different ways. A common formulation states that if the sequential component of a program is $1/s$ then the maximum speedup that can be attained on any parallel computer is s. A typical MD program has a sequential component of about 10 percent, which according to Amdahl's law would imply a maximal speedup of 10. What must not be overlooked is that Amdahl's law assumes that the problem size is fixed. For many problems this is not really the case. Furthermore, as the parallel component of a problem grows at some rate, the sequential component will grow at a slower rate or not at all. This observation, often called Gustafson's law [129], implies that large parallel computers can achieve excellent speedup if the problem to solve is allowed to grow with the number of nodes employed.

Performance modeling can be approached in different ways. One way is to formally derive the asymptotic behavior of the most time critical part of the program.

The asymptotic behavior of an algorithm gives an estimate of the execution time as a function of problem size and of possibly other parameters. The notation that is commonly used is called "big-oh" [130]. For example, the statement that says that some method scales as $\mathcal{O}(N^{3/2})$ means that there are positive constants c and N_c such that for all N greater than or equal to N_c the execution time, $T(N)$, of the method is bounded by $cN^{3/2}$. More formally we express this as

$$T(N) \text{ scales as } \mathcal{O}(N^{3/2}) : \exists N_c, c > 0 \text{ such that } T(N) \leq cN^{3/2}, \forall N > N_c. \quad (1.9)$$

The "big-oh" analysis may be misleading since the relevant problem sizes at hand are much smaller than N_c and the constant c may also be quite large. To get more relevant information that can help in the optimization process, it becomes necessary to develop empirical models and perform benchmark runs [100].

Once the program has been verified and its basic performance characteristics is understood, it may also be relevant to perform some processor specific fine-tuning. This can consume a lot of time – and therefore one should be quite sure that it is worth the effort before spending time on it. To first approximation, the parallel computer model I assume (§1.3.2) also decides the performance model. Since I postulate the decoupling of calculations and communications, account for these activities separately; denote them by T_{calc} and T_{comm}, respectively. Furthermore, because of imbalance in the computation or computer system also include the time spent idling, T_{idel}. The total run-time, T, of an application on a system with P processors can thus be factored into

$$\begin{aligned} T &= T_{\text{calc}} + T_{\text{comm}} + T_{\text{idle}} \\ &= \frac{1}{P}\left(\sum_{i=0}^{P-1} T^i_{\text{calc}} + \sum_{i=0}^{P-1} T^i_{\text{comm}} + \sum_{i=0}^{P-1} T^i_{\text{idle}}\right). \end{aligned} \quad (1.10)$$

The latter quantities are straightforward to find, since they can be measured on each node separately and then summed at the end of a run. By reordering or overlapping communications and calculations it may be possible to decrease T_{idle}. Should this not be feasible, a load-balancing strategy should be tried.

The most often used of the time to send a message consisting of L bytes, over a link with a startup time t_s and bandwidth t_w, is given by

$$T_{\text{comm}} = t_s + t_w L. \quad (1.11)$$

In MD simulations the two most common communication patterns are nearest neighbor communication on a logical grid and global all-to-all communication. In both of these cases the amount of data communicated is often not large. This means that the actual communication behavior can differ substantially from the simple model of Equation (1.11) [131, 132]. The upshot of this is that selected methods should be based on actual measurements.

1.3.5 Hardware aspects

In large-scale scientific computing, including MD simulations, the most performance critical parameters of the parallel computer hardware from an application perspective are:

- bandwidth to memory,
- physical memory per processing node,
- sustained processing power of each node,
- network latency and bandwidth,
- sustained bandwidth to secondary media,
- aggregated physical memory.

The items in the above list are ordered in terms of decreasing approximate importance. The first two items on the list are concerned with the physical memory of the machine and the processing power of each node comes only in third place. Network and secondary media follow thereafter. This ordering puts the focus on the primary bottleneck in (parallel) computers used for scientific computing with large datasets.

Organization of memory

Currently, the two most common types of computer memory technology are static and dynamic RAM. Their respective acronyms are SRAM and DRAM. In SRAM designs the emphasis on *speed*[5]. DRAM designs focus primarily on capacity. DRAM designs uses a single transistor to store one bit while SRAM designs uses four to six transistors per bit. This difference in design has consequences for how persistent the contents of the memory are over time and accounts for most of the performance difference between the two flavors.

Assuming comparable memory technologies SRAMs are about 8 to 16 times faster than DRAMs, but also 8 to 16 times more expensive; the capacity of DRAMs are a factor of 4 to 8 to that of SRAMs. The growth rate of DRAM capacity is a factor of four between generations which appear approximately every three years (60% per year). Unfortunately, the speed (access time) is only going up at a rate of 20% per generation, and the latenecy is going down by about 7% per year[85].

Microprocessors have been getting 60% faster every year since 1987. This means that there is a CPU-DRAM performance gap that is growing exponentially with a factor 1.5 every year [85]. This growing performance gap makes it more and more difficult for large scale simulations to extract the performance gains that the increase in microprocessor peak performance appear to promise[133, 134, 135]. It is much more likely that the increase in performance will be on a curve with a similar slope to that of the memory access time.

[5] "SRAM can not swing a dead cat at DRAM capacity."

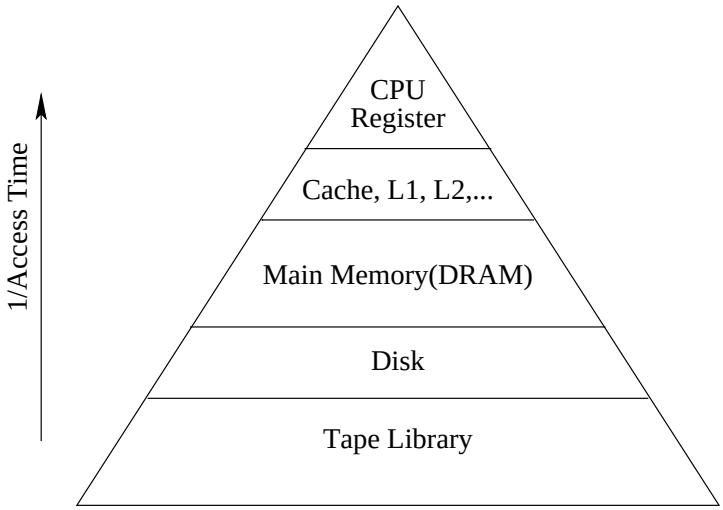


Figure 1.5. The memory organization pyramid. At the base of the pyramid we have the most inexpensive and also largest storage capacity. As data moves up the pyramid the access time becomes shorter, the memory technology more expensive, and because of cost constraints there will be less storage capacity. At the top of the pyramid we find the CPU registers.

In current high performance computer architectures one of the biggest challenges is to optimize the performance of the memory system using a cost constraint. In a broad sense, there are basically two organizations of memory: interleaving of memory banks or a hierarchy of memories.

Interleaving memory banks mean that the memory is organized into B number of banks. Consecutive memory locations are stored in adjacent memory banks: a word with address a is stored in bank number $a \bmod B$. The memory cycle time is the minimum time between accesses to a memory chip. This means that there is a maximum rate at which a memory chip can receive requests and consequently a minimum time between two accesses to the same memory bank, the bank busy time. This has performance implications for a program that has a memory access pattern that hits the same bank more often than the bank busy time.

The hierarchical memory organization tries to maximize performance by using several layers of increasing capacity and decreasing access time. Thinking of the hierarchy as a pyramid, the memory technology of the base may vary. See Figure 1.5. At the top of the pyramid we have the CPU with a small number of fast registers. There are one or more levels of caches often called L1, L2 *etc.* Since a cache should be fast it is constructed using SRAM. A typical solution is an L1 and L2 cache on the same chip as the CPU and an external L3 cache made up of SRAM.

Caches are organized in a number of equal sized slots known as cache lines. A cache line consists of several consecutive memory addresses. The line size varies from design to design, but is usually in the range of 64 to 512 bytes wide. When the CPU requests a data item from main memory, the memory subsystem will check to see if it can be found in cache. If the data is not in cache, a cache miss occurs. When this happens the data item will be searched for at lower levels of the memory system and when it is eventually found it is brought into the cache.

Data are fetched from memory in units of a cache line. This kind of memory organization is motivated by the observation that data which is used often should be accessible as quickly as possible and when a data item is accessed it is also very likely that data items located close to it in memory will also be accessed soon. So memory access patterns which are local in space and time will be quickly serviced. Molecular Dynamics simulation algorithms often have quite a lot of potential for memory access patterns that are local both in time and space. How well this can be exploited is very much dependent on the data-structures that are used in implementations. Which, of all possible data-structures, are optimal for MD is still an open question.

Type of processing node

From an application performance point of view, the peak performance of a processor is very rarely obtained. MD applications are no exceptions. What matters, is the sustained performance delivered, when running the application in production. The sustained performance actually measured is usually in the range from 5 to 50 percent of peak performance [136, 137, 138, 139, 140]. These sustainable performance factors should not be forgotten when a prize-performance analysis is made.

Interconnect

The network topology of parallel computers used to be vividly debated. In fact, to a such extent that one could believe that it was the most important issue in parallel computing. This is no longer the case. Network speed and latency are certainly important factors in deciding how general a particular parallel machine is, but the problem is that developing new networks that can be built reliably and cost-effectively is hard. Closely related to the network is the problem of I/O. For large scale MD simulations it is becoming a very real problem which certainly must be dealt with. See [141], for an interesting discussion on these issues.

1.3.6 Software and hardware interaction

The node type – microprocessor, multi-processor – as well as the communication network and topology must be considered when designing an efficient program. With the current state of affairs, it is clearly the node performance that must be consid-

ered first. Secondly, appropriate parallel algorithms should be used. They should map the problem domain to the network topology in an efficient way. Further fine-tuning can be done by improving the load-balancing and also considering the network topology, but the efforts spent on further iterations on the program's single-node performance is probably more rewarding.

It may sound like a paradox, but currently, in optimizing for a parallel computer most of the effort should be spent on making sure that the individual node performance is as good as possible. This is a consequence of the power of the individual node compared to the network latency and bandwidth. In short, the current parallel machines are of "large grain" type. The parallel algorithm used should make every effort to communicate as seldom as possible. For best performance it often means that a particular calculated value, needed on several nodes, can actually be recalculated more quickly on each node, compared to communicating it to the nodes where it is needed. This is the parallel form of the classic optimization trade-off between memory and CPU cycles.

1.3.7 Cost of calculating interactions

The naive MD simulation algorithm calculates the interactions for each of the N particles in the simulation with all the other $N-1$ particles. This gives rise to an $\mathcal{O}(N^2)$ computational complexity of the force calculation. Depending on the range of the interactions, it is of course possible to do quite a bit better than this.

Since short-range interactions have a rather limited range one usually makes an approximation and applies a cut-off radius, neglecting the interactions outside this distance. We take this into account by denoting the cut-off range of the interaction by r_c and assume that the particle density is approximately uniform throughout the simulation box. This means that the number of particles, found within each particle's cut-off sphere, is going to be roughly constant and proportional to r_c^d. So by taking advantage of the local nature of forces we can bring the computational complexity down to $\mathcal{O}(Nr_c^d)$.

Long-range electrostatic forces between charged atoms can be treated using many different methods. In MD simulations the models usually only include charges and perhaps dipolar effects. Treating point dipole interaction is large subject in itself [51].

Hierarchical methods for calculating long-range interactions achieve a computational complexity between $\mathcal{O}(N)$ and $\mathcal{O}(N \log N)$, but with considerable variations in the constants, hidden in the ordo notation. Practical implementation show that this is very much true, with reported constants varying several orders of magnitude for the same algorithm. The large variation can be attributed to both the efficiency of the implementations, the actual hardware used and to the accuracy achieved. This current state of affairs implies that the traditional approach based on Ewald summation is still viable, especially since it continues to evolve and improve in computational complexity [142, 143, 144]. Further support of this view is the method I present in §2.5.

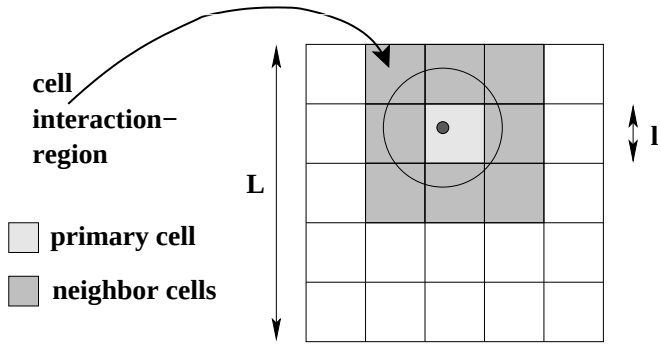


Figure 1.6. The simulation box with side length L, and the sub-cells width side length l. A particle's primary region, primary neighbor region and interaction sphere of radius $r_c \leq l$. The union of the primary region and the primary neighbor region is called the interaction region of the sub-cell.

1.3.8 Algorithms for large-scale MD

I consider first methods for treating interactions where a cut-off r_c is applied. These methods are central for treating both short-range and long-range interactions because the approaches for treating long-range interactions usually split the interaction into a short-range component and a long-range component which can be dealt with separately.

Short-range interactions

The short-range nature of forces can be exploited by making sure that interactions are only calculated between those particles having a chance to interact due to their mutual distance. Conceptually this can be envisioned by subdividing the computational box, with box length L, into smaller cells with a side length, l. The smaller cells completely fill the computational box and each particle is located in exactly one of the smaller cells, which I call the particle's *primary region*. By choosing l so that it is at least as large as the cut-off, $l \geq r_c$, we can be sure to find, for each particle, all the interacting particles in the primary region and the $3^d - 1$ cells that are adjacent to the primary region. Call these adjacent cells, the *primary neighbor region* and the sphere with radius r_c around each particle for the particle's *interaction sphere*. Also, call the union of the primary region and the primary neighbor region for *interaction region* of the sub-cell. See Figure 1.6. All particles located in a sub-cell will have their interaction spheres, by construction, fit inside the interaction region of the primary cell. This geometrical fact explains why it is sufficient to look for possible interacting particles in the cell's interaction region.

Under the assumption of approximately uniform particle density, the computational work in calculating the force on a particle is proportional to the volume that the force calculation algorithm searches for possible interactions. In our nomenclature this is the volume of the cell interaction region, V_I. In 3D we have $3^3 - 1 = 26$ cells in the primary neighbor region and assuming that $l = r_c$ the domain decomposition algorithm above has $V_I = 27r_c^3$. But for each particle the interaction sphere has volume $V_i = 4\pi r_c^3/3$ and $V_i/V_I = 4\pi/81 \approx 0.16$. So of all possible interactions this straightforward method examines, only 16 percent actually do interact.

Now let us further examine the neighborhood of a particle j. The other particles found in the interaction sphere of particle j will change from time-step to time-step. The interaction sphere of particle j moves as j moves and the content of the sphere will change because particle j moves and at the same time other particles move in and out of the sphere. By adding a suitably thick skin, r_s, to the interaction sphere we can expect to find all interacting particles of j within this larger sphere, of radius $r_v = r_c + r_s$, for a number of time-steps. Call the sphere with the radius r_v the particle's *neighborhood*. See Figure 1.7.

By storing the information about those particles belonging to the neighborhood sphere we can use this information to find all the particles in the interaction sphere directly rather than searching through the interaction region every time-step. If the overhead in storing and managing the information about the contents of the neighborhood sphere for each particle can be amortized over a sufficient number of time-steps, N_v, the result should be a significantly faster method than always recalculating the contents of the interaction sphere. The maximum number of time-steps between updates of the contents of the neighborhood sphere will vary during the course of a simulation and will depend of the size of the skin and the nature and state of the system. In general, this can be viewed as a dynamical optimization problem which does not have seem to have a straightforward solution. It is, of course, possible to devise simpler criteria, but experience has shown that it is often better to recalculate the contents of the neighborhood sphere at regular intervals. Choosing the skin r_s in the range from $0.1r_c$ to $0.2r_c$ it will result in N_v being in the range from 10 to 20. In order for the recalculation of the contents of the neighborhood spheres to be sure to find all interacting particles, the size of the sub-cells must be larger than the radius of the neighborhood sphere, $l \geq r_v$.

Again, assume uniform particle density, $l = r_v$, and that the overhead of neighborhood construction is very small. We write down the volume, V_v, that the improved force calculation method will search on average:

$$V_v = 4\pi r_v^3/3 + V_I/N_v \tag{1.12}$$

The quota:

$$V_i/V_v = \frac{4N_v\pi}{(1 + r_s/r_c)^3(81 + 4N_v\pi)}, \tag{1.13}$$

which for $r_s = 0.1r_c$ and $N_v = 10$ gives $V_i/V_v \approx 0.57$. The optimal value may be even better.

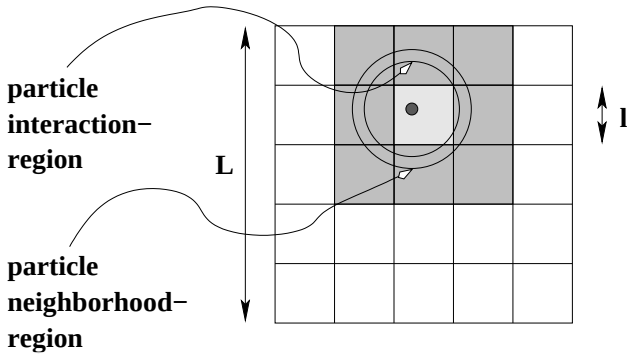


Figure 1.7. The simulation box with side length L, and the sub-cells width side length l. A particle's interaction sphere of radius r_c and its neighborhood sphere of radius $r_v \leq l$.

One of the main drawbacks with the idea that each particle has its own neighborhood sphere is that it will be rather costly in memory. The exact amount of memory needed will depend on the state of the system, the number density and r_s, but a rough estimate is that an order of magnitude more memory is needed. For large-scale simulations this will sooner or later become a real problem. To lower the memory needed, but still do better than the straightforward domain decomposition algorithm we must group particles that are close together. There a several methods reported in the literature [145, 146, 147]. It should be clear from the above that there are a number of variations of the basic domain decomposition algorithm for short-range interaction and there is still a room for improvement both in the methods and the implementations.

Long-range interactions

In this discussion we limit ourselves to electrostatic interactions between point charges. The overall system is assumed to be neutral. The potential field of charge-charge interactions is in fact described by one of the classic differential equations, namely the Poisson equation with periodic boundary conditions. Because of the somewhat unusual boundary conditions it is important to realize that some care must be practiced when applying a solution strategy.

There is a growing number of approaches to treat the essentially infinite reach of charge-charge interactions [74]. To mention just a few which are well adapted to the requirements of MD, we have charge group cut-off [52], the isotropic periodic sum method [66], Lekner summation [148, 149, 150], Ewald [58] summation, smooth particle Ewald [142] summation and particle-particle-particle-mesh (P^3M) [1]. There are also several variations of hierarchical methods [67]; a few examples are the method of Barnes and Hut (BH) [151], multigrid [68, 69, 70, 71, 72] the fast multipole method (FMM), with [152] and without [153] multipoles, and the cell multipole method [154].

Two standard methods are in common use in the MD community: the reaction field method [33, 155] and the Ewald summation technique [57, 58, 156, 157]. There are also various hierarchical algorithms which are quite attractive in principle, but have proved to be challenging to implement efficiently in practice [158, 159, 160, 143, 161, 69, 70, 71, 72].

Since all of these methods solve the same problem, they have some features in common. Excluding the cut-off methods and Lekner's summation formula we have essentially two classes of methods: hierarchical methods and Ewald summation type methods. These two broad classes both view the full long-range interaction as a sum of two components. The first component is short-range and the second component is long-range. The short-range component of both classes can be performed using domain decomposition. The exact manner in which the division into two parts is done differs and this will also result in different approaches for the treating the second component.

Implementation and other issues

The basic methods for treating short-range interactions are often called by the common implementation methods used, *i.e.*, Verlet neighbor lists [16] and linked lists [19, 20]. I believe that this nomenclature should be reserved for the respective implementation methods since they tend to stand in the way for better implementation methods and data-structure that could be developed. It is more appropriate to use names which describe the actual algorithmic ideas.

In support of this view is the observation that neither Verlet neighbor lists nor linked list can be very efficient on cache-based processors, since they have a tendency to access memory in an unstructured way. The same access pattern is also a headache on vector architectures. Examples of data structures that are both efficient and likely to get better cache reuse can be found in [21, 146, 147]. An improvement in the construction of neighbor lists can be found in [162]. It is notable that Everaers and Kremer [147] also report very good vectorization of the method they have developed.

1.3.9 Algorithms for parallel MD

The domain decomposition algorithm described in §1.3.8 can be parallelized in a number of different ways. The MD algorithm contains opportunities for independent operations on several different levels. In principle, the interaction on each particle can be calculated independently of all the others and the same goes for time-integration. For an in-depth discussion of these issues I refer to the review by Fincham [163, 77]. This fine-grain parallelism is not really used in practice because it does not really match the hardware in use today.

Note that the term granularity is often used in two contexts. One referring to the parallel algorithm and the other referring to the hardware. The second meaning refers to the capacity of each node while the first refers to the unit of parallelism

the algorithm exploits. In general, it is important from a performance point of view that the granularity of the parallel algorithm matches the granularity of the hardware. A fine-grain algorithm can easily be made more coarse by the process of agglomeration [100], but the opposite transformation may be much more difficult. In the MD case agglomeration can be accomplished at different levels. Below I discuss some of the levels which can easily be exploited [164] while still retaining the advantages of domain decomposition.

Task queue

At the coarsest level, we may simply run the same program with slightly different starting conditions. This may not look very useful at first sight, but since the objective is to follow a phase space trajectory long enough for the time-scale of the phenomena of interest, it is clear that several simulations running in parallel will accumulate enough statistics faster. So at the start of the simulation several independent tasks are created and given out to the available processors.

This approach is viable if we assume that we depart from an initial state of the system that has been equilibrated and then add small perturbations at the very start of each task. The chaotic nature of the system will make sure that the different trajectories soon become completely uncorrelated and at the end of the run the statistics of the different simulations can be combined. It is also possible to start from an equilibrated system and run two simulations but with opposite direction of time [165].

This approach may be applied using programs that are serial or parallel and is an excellent approach for achieving good parallel speed-up with a minimum of programming.

Replicated data and systolic loops

Replicated data (RD) is an approach which divides the force calculations evenly between the available nodes [166, 80]. Each node is responsible for calculating the forces on the particles which has been assigned to the node. Since the complete system is replicated on each node this is straightforward. When all the forces on the node-local particles have been calculated the positions of these particles can be updated. An all-to-all communication must take place to distribute the new positions of all particles in the system to all nodes. In this formulation of Newton's third law is not used.

By storing a complete force array for the whole system, Newton's third law can be used which halves the force calculations that must be done during each time-step. But before the time-integration step the complete force array must be globally summed and then distributed to all nodes. When this has been done we can choose to integrate the whole system on each node and then go directly to the next time-step [167] or we can update just the node local particle coordinates and then perform and all-to-all communication. See Figure 1.8.

Which one of these variations one should use is mostly a question of the balance between communication and calculation of the parallel computer being used. In

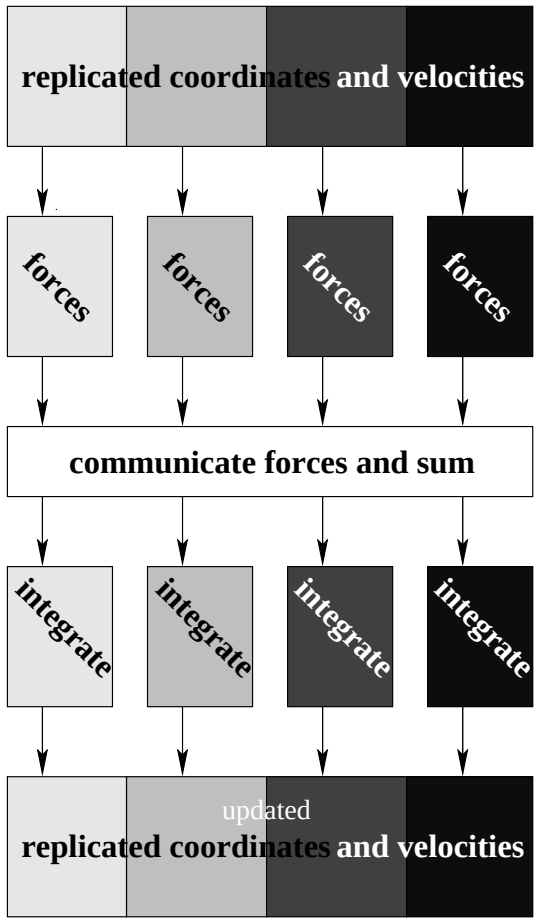


Figure 1.8. Replicated data method using only one global communication step. Particle coordinates and velocities are replicated on all nodes. A complete force array is also stored on each node. By integrating the whole system on each node independently the method only requires on communication step.

any case, as the systems grow larger the RD method will be limited by the all-to-all global communication steps. However, there is an improvement of the RD method which avoids global all-to-all communication [169, 125]. Also it is possible to combine the ideas of RD with systolic loop algorithms. The main reason to do this would be to decrease the need for node memory and it also opens up the possibility for overlapping communication and calculation [166, 170, 171, 172].

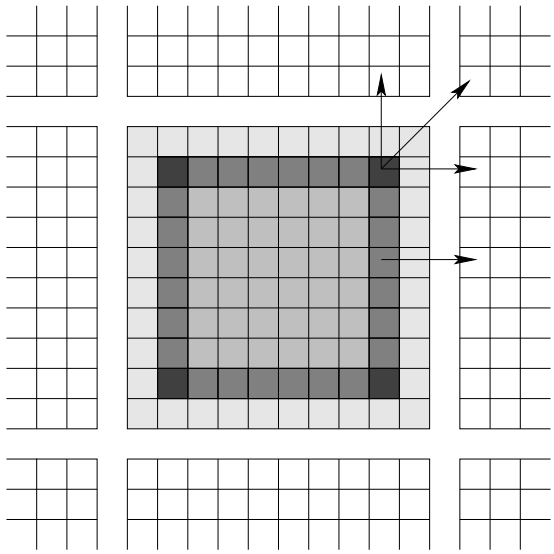


Figure 1.9. The mapping of regions of simulation space to nodes. The arrows show where the sub-cells are needed during the force calculation. The outer layer of sub-cells on each node cell represents the temporary space needed to hold particle position data. (Adapted from [168].)

Spatial decomposition

Spatial decomposition is a parallelization strategy that maps spatial regions of the system to each processor [167]. If these regions are large enough it implies that most of the communication will be between processors that are topologically close and it will also be mostly point-to-point communication. The global communication that is needed will be concerned with obtaining global quantities, like temperature. The domain decomposition algorithm naturally fits with the spatial decomposition parallelization strategy of §1.3.8.

With coarse-grain nodes, fairly large regions of simulation space, containing several sub-cells in each coordinate direction, should be mapped to each node. Call these larger regions of space *node cells*. See Figure 1.9.

Using a cubic simulation cell, there are three basic classes of node cells: slice, beam and block. To minimize the volume of communication the cubic node cell are clearly the most efficient because it has the largest volume to surface ratio. If this subdivision can be used it is clearly preferred, but other factors, like communication latency and mapping of regions to physical nodes, may make it favorable to use a slice or a beam decomposition.

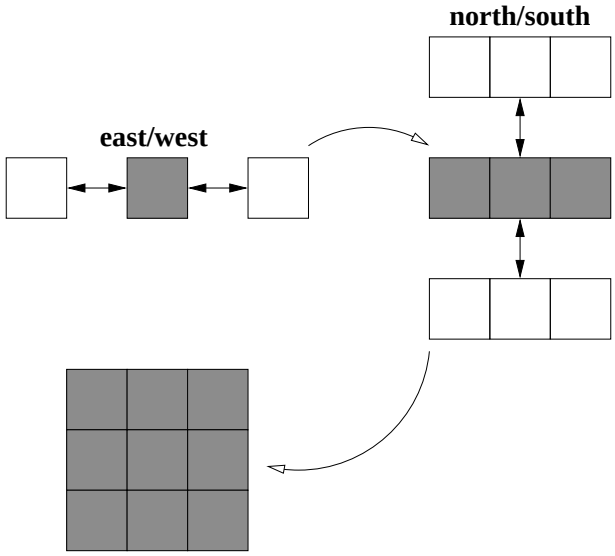


Figure 1.10. Retrieving the complete node region neighborhood consisting of 8 cells in 2D using only 4 communication steps. The generalization to 3D is straightforward by adding another two communication steps. The first two steps communicate in the east/west direction, each sending and receiving particle positions that are within a distance r_v of the respective node cell boundary. The following two step perform the communication steps in the north/south directions, but now some of the newly received particles should also be communicated. See also Figure 1.9. (Adapted from [80].)

During each time-step the surface particles of the node regions must be communicated to the neighbor nodes that require them in their local force calculations. Here we have a choice of using Newton's third law or not; this is a typical parallel optimization trade-off between using more memory and recalculating results [168]. In the case of cubic node cells and not taking advantage of Newton's third law we can bring in the complete node region neighborhood of all 26 cells using only six communication steps [80]. See Figure 1.10. This means that a fair amount of temporary memory has to be available.

An alternative method described in [173] is more aimed at saving memory rather than communication. To take advantage of Newton's third law we must also send back the calculated forces to the originating node. See Figure 1.11. This means that we communicate half as much data but twice as often. The positions have to be sent out and the calculated forces sent back. The overhead in communication may often swamp the gain from not recalculating forces. Still for some computer systems this is still an effective approach [174].

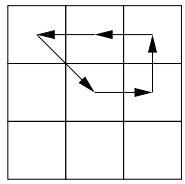


Figure 1.11. This communication pattern should send both particle positions and force accumulators. Between communication steps force calculations are performed and forces are successively accumulated. (Adapted from [173].)

The spatial decomposition strategy has the potential to scale linearly for simple systems. However, for more complex systems with large biomolecules it is not yet clear how to best represent the large molecules in a distributed manner. This challenging problem is discussed in [175]. The strategy may also suffer from load imbalance which results in poor scaling. Some of the possible advanced load balancing strategies are discussed in [176].

1.4 Thesis Scope

As the capacity and availability of computers have increased, the MD models that can be fruitfully simulated have grown in complexity. The construction of the first successful supercomputer with vector registers – the CRAY-1 – in the mid 1970's meant that new methods and algorithms had to be developed in order to fully utilize the hardware capacity. This interplay of hardware and software became more important when parallel computers appeared and started to be used for MD simulations [86, 77] and continues to grow in importance as the peak capacity of of the most powerful hardware platforms are expected to surpass the Pflop/s limit around 2010 [177].

Now that commonly available parallel hardware and software platforms have evolved to a sufficiently stable and mature state, their combination and application to large-scale Molecular Dynamics provide both new challenges and new opportunities. This is further underscored by the comparatively affordable parallel platforms available through clustering of commodity processors and high-speed networks. In describing the scope of this thesis I first attempt to quantify "large-scale" Molecular Dynamics simulations. The term "large-scale" in connection to MD simulations has suffered from a very severe inflation ever since it was invented, in the beginning of the vector-supercomputer era in the early 1980's [178, 99, 177]. This, of course, has been unavoidable due to the exponential growth in capability in computer hardware technology. When classifying Molecular Dynamics simulations quantitatively, the

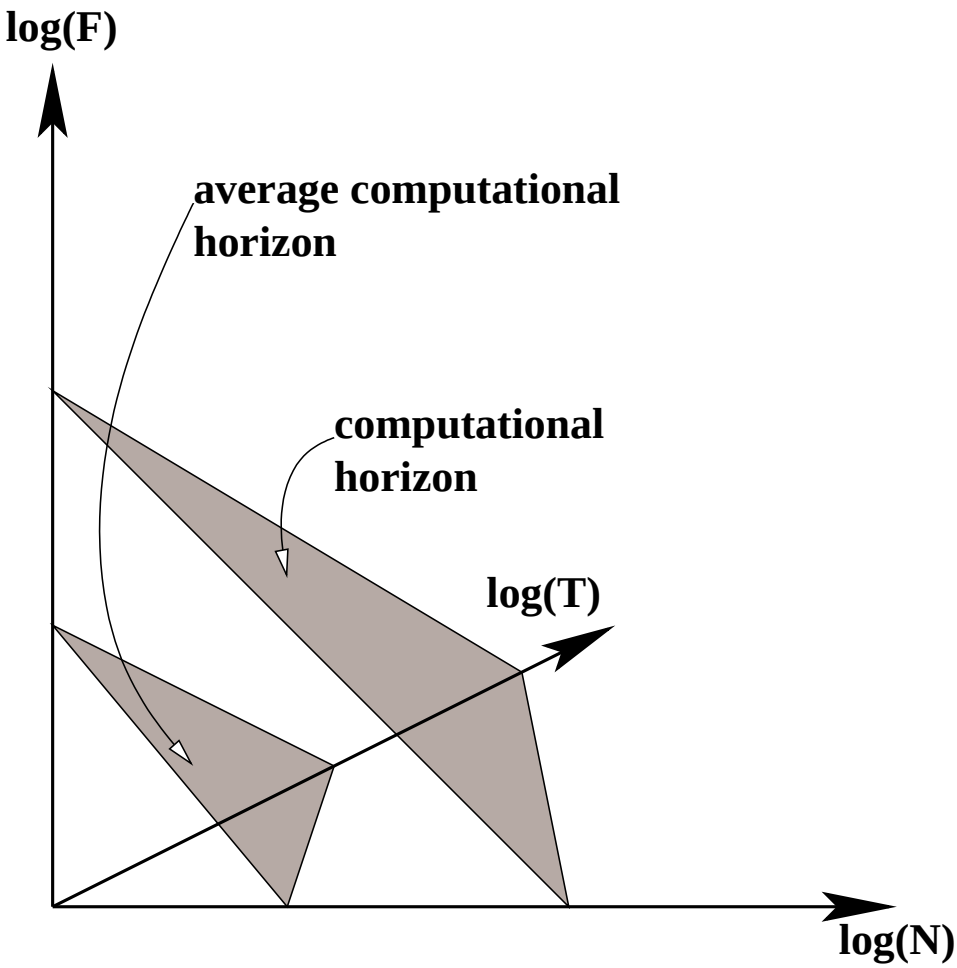


Figure 1.12. A qualitative sketch of the simulation-scale phase-space. The intersection points of the computational horizons with the (N, T, F) axes move towards larger values over time because large simulations become possible to perform.

most important factors are: size of the system, coverage in time and the complexity of the used model. Focusing on these particular aspects, I assign the following three parameters to characterize any MD simulation:

- N, the total number of particles (or mass-points) in the simulation,
- T, the total number of time-steps in the simulation,
- F, the number of floating point operations per interaction and per time-step.

To obtain a overview, I bundle them together into a three-dimensional space, (N, T, F), which I call the *simulation-scale phase-space*. The range of these numbers vary several orders of magnitudes so it makes good sense to use a logarithmic scale in each direction of the space. To define and use the first two dimensions (N and T) is straightforward, while the third is more elusive. However, for our purposes, reasonable estimates are quite sufficient.

In the very first MD simulations [10] only 32 particles were used. This number was soon scaled up to close to 1000 particles [15]. The highest number of particles in a simulation has gone up quite dramatically during the last couple of years. There are reports of production calculations using over a 1 billion [180] and benchmarks calculations for systems containing over 10 billion particles interacting via short-range interactions [181]. The number of time-steps in an average simulation is usually in the range from 10^4 to nearly 10^8 (corresponding to several hundred nanoseconds). So currently I estimate that $\log(N)$ is in the range from 3 to 10, and $\log(T)$ from 5 to 8.

Concerning the number of floating point operations, a simple Lennard-Jones effective pair-potential has $\log(F) \approx 1.5$, while common polarizable potential models have roughly $\log(F) \approx 2.5$, because they have to be solved iteratively for self-consistent results. However, recent developments is bringing this down to $\log(F) \approx 1.8$, by combining fast methods for electostatics and improved iterative methods [182]. By generalizing the interactions and leaving the classical MD interaction regime, I estimate $\log F \approx 8$, for a pure quantum many-body interaction at the Hartree-Fock level [183] using a limited basis sets. To allow for more exact interaction potentials in MD simulations I estimate that $\log F$ will stay in the range from 1 to 10 in the near future. Using these estimates, imagine collecting simulation-scale phase-space points, (N, T, F), for a large number MD simulations. Now, because computer resources are, after all, finite, all points can be found in the first octant, below and to the left of a plane. Acknowledging this state of affairs it is appropriate to call this plane the *horizon of the* MD *simulation world* [184], or simply the *computational horizon*. The major part of all production calculations are still performed with a moderate number of time-steps (corresponding to a few hundred nanoseconds), using empirical pair potentials, and on systems with sizes, much smaller than what could be maximally possible. So, in fact, the vast majority of the points should be expected to lie in a region to the left of a plane I call the *average computational horizon*. The truly large-scale simulations can be found between the two planes.

Due to advances in implementations, algorithms, compilers, system software and computer hardware capabilities, the computational horizons are steadily expanding. A schematic representation of this is given in Figure 1.12. The intersection points of the computational horizons with the (N, T, F) axes move towards larger values over time, as simulations containing larger number of particles, extending over more time steps and using more complex potentials become possible to perform.

A natural question to ask is what is the motivation behind performing larger and larger simulations? And, do we really need large-scale simulations? The simple answer is *yes*! They allow us to perform more reliable and realistic simulations at the same time as bigger and more complicated systems become possible to study. The situation is analogous with computational quantum chemistry or weather forecasting based on computer models. More specifically there are several factors pushing the development towards larger simulations. A few examples based on the (N, T, F) parameters in Figure 1.12:

- Longer simulations or a series of shorter simulations will give a more reliable sampling of the phase space. Especially conformational phase spaces of flexible molecules. Also longer simulations are needed to get reliable statistics for dynamical phenomena with long time constants.
- As the experimental techniques become more refined, it becomes possible to perform simulations containing more or less the same number of particles as the system on which the actual experiment is conducted on. This can obviously be of great help in interpreting experimental results as well as giving very detailed information at the atomic level, which would not otherwise be available to the experimentalist.
- Immersing large biomolecules in solutions with solvent molecules and counterions is computationally expensive. Most of the computational cost is spent on simulating the solvent and electrostatic interactions. In a high quality simulation, the solute should be solvated with several layers of solvent molecules so that even a bulk region is included.
- MD simulations based on first principles quantum mechanical forces will become more and more widespread. These methods are *dramatically* more expensive than classical MD simulations. (F increases several orders of magnitude, while N and T have to be decreased in these simulations in order to make them feasible)

This thesis investigates methods for performing large-scale parallel Molecular Dynamics in the sense given above. Referring to Figure 1.12 I have explored methods that advance along the F and N axis.

As for the search for truth, I know from my own painful searching, with its many blind alleys, how hard it is to take a reliable step, be it ever so small, toward the understanding of that which is truly significant.

Albert Einstein
Essential Einstein

Chapter 2

Results

Molecular Dynamics algorithms tailored for systems requiring large-scale parallel simulations is the focus of this Chapter. I start by describing the development (§2.1) and parallelization (§2.2) of an *ab initio* method that increases the number of operations (F) significantly, essentially recomputing the interaction potential at each time-step; I continue with studies of methods designed for massively parallel hardware platforms and large numbers of particles (N) with short-range (§2.3) and long-range (§2.4) interactions; I end with a novel method (§2.5) for treating electrostatic interactions.

2.1 QMD: a Novel ab initio MD Method

In the paper *"A Parallel Quantum Mechanical MD Simulation of Liquids"* we present and test a novel approach for doing Molecular Dynamics using forces calculated from first principles. The method employs atomic forces calculated as gradients of the variational energy expressions [185, 186]. Arbitrary levels of quantum chemical methodology and electronic state may be used. The computational scheme is simple to implement, although rather expensive to use. Furthermore, the approach can be applied to studies which are beyond the capabilities of current classical simulations, *e.g.*, simple chemical reactions or other chemical processes involving excited electronic states or radicals.

We test the method using simulations of liquid water (with all internal degrees of freedom included) and periodic boundary conditions. A modified version of the computer simulation program "McMoldyn" [187] is the classical starting point and we interface it to a standard quantum chemistry package, "Gaussian94" [188]. For our purposes, any other quantum chemistry package that contains calculations of atomic forces could just as well be used. Forces are calculated at the semi-empirical AM1 molecular orbital (MO) level, and at the *ab initio* SCF MO Hartree-Fock level. Comparisons are made with corresponding classical simulations of water, with simulations using the Car-Parrinello method [189], and to experimental radial distribution functions [190].

This is a time consuming method and it is necessary to use parallel computer platforms to achieve the necessary compute power. Scaling and load balancing properties of the adopted parallel scheme are discussed.

2.1.1 Algorithm

We *replace* the classical intra- and inter- molecular force calculations in a classical MD program by the corresponding quantum mechanical force calculations. Forces are calculated as gradients of the variational energy expressions [185, 186].

Start from the classical MD method with periodic boundary conditions (PBC) and distances calculated from the minimum image criteria. For simplicity use the neighbor-list (NL) technique [16]. We reuse this algorithmic structure, but the classical pair-wise force calculations are substituted with quantum mechanical (QM) calculations.

For each molecule in the simulation we get one QM calculation with the molecule's current NL included and minimum image criteria applied. See Figure 2.1. Using this approach all participating molecules are treated in the same way; we get N "cluster" calculations per time-step in a simulation containing N molecules. The force calculations thus become many-body interaction calculations. It is important to note that to continue with PBC we must treat each central molecule in every NL in the same way.

The total energy is the sum of the potential energy of the nuclear positions and the electrons plus the kinetic energy of the electrons and the nuclei. The kinetic energy of the nuclei is computed using classical molecular dynamics. The QM calculations are iterated to the same accuracy at each time-step to ensure conservation of the energy. The approach described above leads to the following tasks at each time-step:

1. Generate input files for the quantum chemistry program from the MD program. Coordinates of molecules, kept in the respective neighbor-lists, are used. Molecules in each neighbor-list become clusters. They form the input to a QM calculation. With N molecules in the simulation cell, N quantum mechanical cluster calculations are carried out in each time-step.

2. From each cluster calculation we extract the forces acting on the central atom in the cluster. We also get energies, and atomic charges from a Mulliken population analysis. Loop through all N clusters and calculate the forces on each molecule.

3. Finally perform a numerical time integration and update the position and velocity of each atom.

The cost of a time-step will depend on the size of the system (N), the sophistication level of the QM method employed, and the number of functions (L) in the basis set. Since the clusters we calculate on can not be expected to contain any symmetries to reduce the number of operations, the computational complexity becomes $\mathcal{O}(NL^4)$.

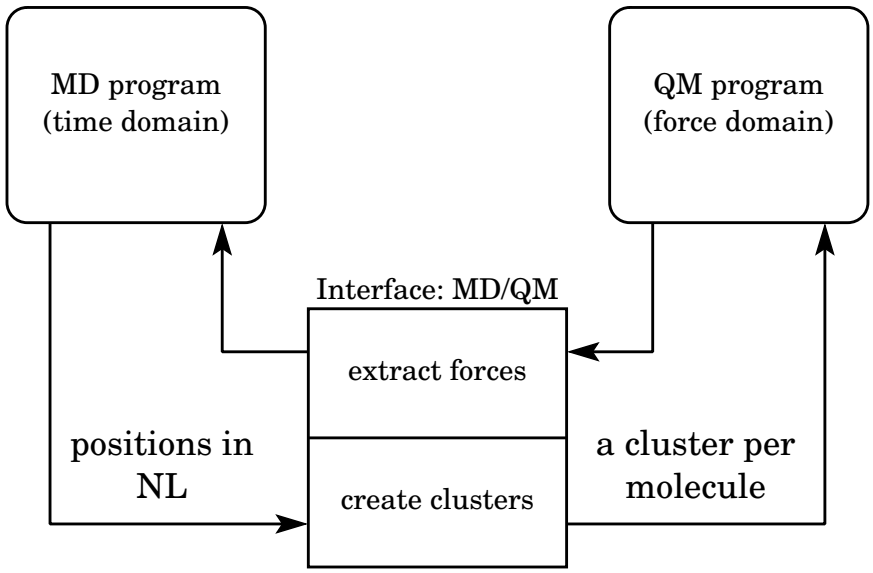


Figure 2.1. A quantum mechanical MD method that is classical in the time domain (CT) and quantum mechanical in the force domain (QF) – CTQF. The MD program generates input files for the quantum chemistry program. With the minimum image convention applied, the coordinates in the neighbor list of each molecule and the molecule itself result in a "cluster" and a corresponding input file. From each completed cluster calculation forces acting on the atoms of the central molecule in the cluster are extracted. With all forces returned to the MD program it can perform a numerical time integration step to update the position and velocity of each atom in the simulated system.

2.1.2 Method test

Comparing simulations of small and large classical systems of water, one finds minor differences. We assume that the same holds true for small and large QM simulations. A system of liquid water is prepared using classical MD. A cubic box is filled with 32 water molecules to a density of 1.0 g/cm^3, giving a box length of 9.86 Å. Water molecules in the classical simulation are described using the flexible SPC potential model [191, 192]. Periodic boundary conditions and minimum image convention are applied. The system is equilibrated classically at 300 K. Temperature is maintained using the Nóse-Hoover method [193, 194]. The time-step is chosen to 0.2 fs. This ensures that various types of fast motion of the water molecules are properly treated as well as all the internal degrees of freedom, including bond-stretching and angle bending.

The neighbor-list technique is used to calculate the intermolecular interactions. Let the cut-off be half the box length. For each of the 32 water molecules, there

is a spherical volume around its centre-of-mass (COM) containing 18-22 neighbors. The varying number molecules kept in the neighbor-list is due to local density fluctuations. After an initial classical simulation, the Newtonian forces are turned off and the corresponding quantum mechanical forces, supplied by the interfaced quantum chemistry software, are activated.

MD simulation at the AM1 level

After a classical MD simulation was carried out to equilibrate the system of liquid water, the AM1 force field was turned on and the system equilibrated during 2000 steps. It took about 800 steps for the water molecules to adjust into the new interactions, after which, temperature and energies were fluctuating around their mean values. The simulation was continued another 3000 steps. The total length of the simulation using the AM1 force field corresponds to 1.0 picoseconds. Use of the AM1 method also serves as a soft intermediate between the classical force field and the *ab initio* Hartree-Fock force field.

MD simulation at the Hartree-Fock level

The final configuration from the AM1 simulation was taken as an initial configuration for a run with a Hartree-Fock force field. Again, the water system was first equilibrated for 2000 steps in order to get it adapted into a new environment of interactions. In Hartree-Fock calculations, the Fock matrix is diagonalized iteratively until a specified threshold, normally based either on the energy or on the electron density, is reached. The energy of the whole system of water molecules in the simulation cell is assumed to be conserved by solving the HF equations to the same accuracy during each cluster calculation. This resulted in some fluctuations in the numbers of iterations to reach convergence. As in the case of the AM1 model, all runs had to be carried out without the use of symmetry. The basis set we used was the limited STO-3G. The number of time-steps of the simulation was the same as for the AM1 model. To the best of our knowledge, this is the first MD simulation of liquid water carried out at the Hartree-Fock level.

Liquid structure of water

The Figures in the paper show 3 sets of radial distribution functions. Compared to the experimental curves, all the simulations (both the classical and the two quantum mechanical) give the first maximum position shifted to a closer distance. Compared to the $g(r_{OO})$ function from a "full-scale" classical simulation with a larger number of molecules using the same potential model [192], there are no essential differences between the results. This would indicate that the liquid near structure of water can be described reasonably well using as few as 32 water molecules. The position of $g(r_{OO})$ from the AM1 simulation appears to be closer to the experimental curve than the STO-3G curve but having a lower peak height.

For the $g(r_{OH})$ curves there are some striking feature. The AM1 model completely fails to describe the hydrogen-bonds. The HF curve is much more shifted to closer distances than the classical $g(r)$, giving a too short H-bond distance. However, the height of the second peak is in better agreement with the experimental curve.

For the $g(r_{HH})$ curves the AM1 model fails to reproduce the characteristic shape of the H-H structure of water. The flexible SPC model and the HF STO-3G both give too high intensities in comparison with the experimental curve. Again, the HF curve is shifted too much to closer distances.

It is encouraging to observe that the structure and dynamics (see paper) of liquid water can be described reasonably well by our simulations. In summary, the results of our simulations is a strong indication that our method works.

2.1.3 Parallel platform

The parallel computer used was the IBM Power Parallel SP, commonly called the IBM SP-2. The SP-2 had a MIMD architecture: the interconnect was a high performance switch with each node being a standard processor with local memory.

We started from a conventional MD program, added code to evenly distribute the cluster calculations to different processors and also interfaced it to a standard quantum chemistry package. The parallel communication library used was IBM's MPL. We have used the User Space (US) communication subsystem and the high-performance switch. The parallel scheme is implemented in such a way that the program will automatically accommodate for the number of available nodes at program start-up. The Unix system call *system* is used to start the cluster calculations from the MD program. To be able to use both US communication and *system* we had to explicitly tell the compiler to use the extended memory model. Further details of the software and hardware configuration used can be found in the paper.

2.1.4 Efficiency

Parallel computers are an obvious alternative when solving very CPU intensive tasks [195]. MD simulations are naturally parallel since, during each time-step, interactions on molecules can be calculated and accumulated independently of other molecules. In the method under consideration, the number of tasks and processors are approximately the same and the character of force calculation is very much different than in a classical MD scheme. An overwhelmingly large fraction of the total CPU simulation time is spent on calculating these interactions.

During each time-step, the quantum chemical calculations of molecular clusters are, besides very CPU demanding, also completely independent of each other and can be done concurrently. Also the integration of forces can be done locally to obtain the new positions and velocities for the atoms. The amount of data to be communicated is limited to the atomic coordinates, velocities of each molecule and a few molecular properties. The inter-processor communication should practically

not take any time at all in comparison to the time the nodes are busy doing the force calculations. At first sight, this problem appears to be an ideal and straightforward problem to parallelize. We have a number of large-grain tasks that need to communicate very small amounts of data, and during each time-step we only need one synchronization point. This problem naturally fits the Replicated Data (RD) [196, 197] parallelization. See Figure 1.8.

However, we can not use more than N processors and load balancing becomes a problem if there are variations in the time it takes to complete a cluster calculation task. With only one task per processor the most time consuming task in every time-step sets the pace, and this means that efficiency will decrease. Since the problem complexity is $\mathcal{O}(NL^4)$ it is clear that in order to efficiently use more processors than there are particles we must also be able to parallelize cluster calculations (the L dimension).

We have calculated the speed-up by varying the number of nodes and processors. The speed-up achieved is respectable for almost all cases. The most outstanding exception comes from running the AM1 model on 32 processors. The explanation for the poor speed-up in this case comes from the fact that the calculation times of each cluster calculation are short and the variations in calculation time are relatively large. In general, since we are using a static assignment of molecules to processors, we can not expect speed-ups to be perfectly linear. Using a dynamic assignment based on previous calculation times for each molecule one could devise a load balancing scheme that would be slightly more efficient.

In the following paper (§2.2) we devise a method for using more than N processors as well as a number of different load-balancing strategies.

2.2 QMD: Improving Parallelization and Scaling

THE previous paper (§2.1) identified two main hurdles for parallel scalability: load-balancing and ability to use more processors than "clusters". In the present paper *"Parallel aspects of quantum molecular dynamics simulations of liquids"* we improve the computational approach by providing solutions to these problems.

Our approach to *ab initio* MD is built on the basic assumption that quantum mechanical interactions are short-range, particularly in comparison with Coulomb interactions between point charges, present in conventional molecular simulations. In our case the liquid is built from overlapping clusters where the effective radius of the cluster is set by the range of quantum mechanical interactions. The number of clusters is equal to the number of molecules, each molecule carrying its surrounding in a small spherical liquid droplet. The force field is calculated without any pre-determined parameters and without an assumption of pairwise additivity of interactions. No distinction is made between intermolecular and intramolecular forces and no constraints are imposed on the molecular geometry. This last feature dictates that a very short time-step is used.

The system simulated is the same as in the previous paper. Results from three different simulations are presented. The basis sets used were STO-3G, 6-31G, and 4-31G*. Further details of these aspects are given in the paper.

2.2.1 Parallel algorithm

During each time-step quantum chemical force calculations are performed for each of the N molecules in the system. In addition to the sophistication level of the QM method employed, the cost of a time-step will depend on the size of the system (N) and the number of functions (L) in the basis set used. The cluster of molecules on which we calculate can not be expected to contain any symmetries to reduce the number of operations. When using SCF Hartree-Fock the computational complexity becomes $\mathcal{O}(NL^4)$.

These calculations are independent of each-other. This gives a first level of parallelism. To be able to use more processors than there are molecules we must also parallelize the QM calculations (the L dimension). This second level of parallelism comes from distributing the force calculations of each molecule over a number of processors. On the first level, we thus have a number of large-grain tasks that only need to communicate very small amounts of data, and during each time-step we only need one synchronization point. The second level can be obtained by using a parallel QM program. This problem naturally fits the Replicated Data (RD) parallelization strategy [77, 167]. See Figure 2.2.

2.2.2 Parallel implementation

A simulation is made up of thousands of time-steps and each time-step consists of N force calculations. Consequently it is essential that each QM run starts quickly. GAMESS [198] is an example of a general QM simulation package that meets this requirement.

In constructing a parallel program that in turn can start several parallel instances of GAMESS we need to decide on how many processors to assign to each individual instance of GAMESS. In the general case, an appropriate partitioning of the system must be found. This can be done by measuring the wall-clock time it takes to perform the typical QM force calculations that are needed during the course of the simulation. For a realistic assignment we also measure how these wall-clock times vary when more processors are used.

In our simulations we have used a system which only contains one type of molecule and this makes it reasonable to use the same number of processors for each instance of the QM program. This can be achieved by arranging the available processors in an p $\times$ m grid topology. In the implementation we use MPI. See Figure 2.3. Each column is one parallel pool assigned to each instance of GAMESS and the processors in the first row act as masters for each pool [199]. For a production run we choose the number of processors ("pool size") that delivers the best

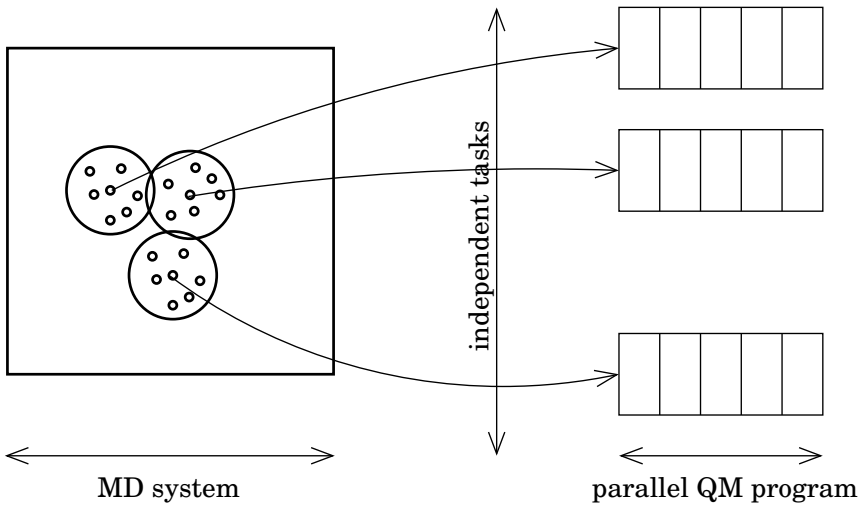


Figure 2.2. During each time-step, each of the N "cluster" calculations are independent. Each of these calculations can also be parallelized. On the first level, we thus have a number of large-grain tasks that only need to communicate very small amounts of data, and during each time-step we only need one synchronization point. The second level can be obtained by using a parallel QM program.

efficiency. When simulating an inhomogeneous system it is probable that other arrangements with pools of different sizes are better.

In implementing the above scheme on the IBM SP-2 we faced several practical issues. The details of how these were resolved are found in the paper. Benchmark results on water systems of different sizes are also reported for runs using from 16 up to 128 processors.

2.2.3 Better load-balancing gives improved scaling

Increasing the number of processors means that the number of tasks allocated to each pool goes down. The initial benchmark results show less than ideal scaling.

The fundamental reason behind this behavior is the lack of tasks. This happens when the last few tasks during each time-step is about to complete and all the other pools are left idle. Better load-balancing strategies are needed which can handle the varying force-calculation times and which are also reasonably simple to implement.

Starting from the simple static assignment of tasks to pools we have made several

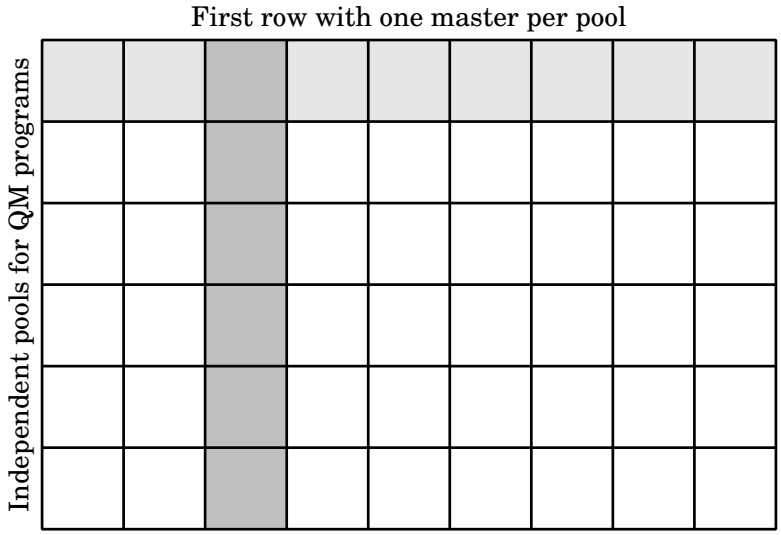


Figure 2.3. In simulations which only contain one type of molecule it is reasonable to use the same number of processors for each instance of the QM program. This can be achieved by arranging the available processors in an p × m grid topology. The processors in the first row participate in the RD part of the parallel program and also act as masters for each pool. Each column is one parallel pool assigned to each instance of the QM program.

step-wise improvements to find a better scheme. The most obvious candidate is a regular master-slave approach which will give out tasks to workers as soon as they request it. By also keeping track of the wall-clock time used by each of the N force-calculations during the previous time-step, the master can give out tasks in a time-sorted order, starting with the tasks that took the most time during the previous time-step. These two load-balancing schemes will distribute the available work-load more evenly, but they do not actively address the task starvation issue. One way of doing this is to change the number of pools allocated to a task once the number of tasks remaining between two time-steps goes below a fixed threshold. A natural threshold to use is the number of pools in the simulation. So, when there are less tasks remaining than the number of pools, then tasks are allocated to two pools which then combine forces on the allocated task. This algorithm keeps more processors busy and result in better load-balancing and scaling. In the paper we show the scaling behavior of these four load-balancing algorithms on a system of 64 water molecules. Benchmark runs on an IBM SP using from 16 up to 128

processors show that a load-balancing scheme based on the master-slave approach with time-sorted work allocation of tasks strikes a good balance between obtained speed-up and implementation effort.

2.3 CMmd: Data-Parallel Short-Range Interactions

THE paper *"Data Parallel Large-Scale Molecular Dynamics for Liquids"* develops a parallel algorithm suitable for simulating systems consisting of particles that interact via short-range interactions. The interaction used in the paper is the spherically symmetric Lennard-Jones potential: with a pair of atoms i and j located at $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$, separated by a distance $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$, the potential function used is

$$V(r_{ij}) = \begin{cases} 4\epsilon \left((\frac{\sigma}{r_{ij}})^{12} - (\frac{\sigma}{r_{ij}})^{6} \right) \text{ when } r_{ij} < r_{cut} = 2.8\sigma, \\ 0 \text{ when } r_{ij} \geq r_{cut}. \end{cases} \tag{2.1}$$

The parameter σ can be interpreted as the atomic diameter and the parameter ϵ controls the strength of the interaction.

The algorithm presented in the paper is a refinement of the serial link-cell algorithm combined with the neighbor-list algorithm and adapted for a SIMD parallel platform [19, 20, 87]. Furthermore, we show that it can be efficiently implemented on a SIMD massively parallel computer – the Connection Machine CM-2. The implementation uses CM Fortran, a high-level parallel language and an early precursor of High Performance Fortran (HPF) [200] – a dialect of Fortran 95[113].

2.3.1 Parallel platform

The Connection Machine platform from Thinking Machines Inc. (TMC) was a massively parallel (SIMD) computer that contained both 1-bit processors and floating point processors, as well as a high-performance parallel file system [201, 202]. The memory was local to the processor nodes. The clock frequency was 8 MHz for the CM-2 model and 10 MHz for the CM-200. The CM-2 hardware system was built in sizes of 4K, 8K, 16K, 32K and 64K. A fully configured system contained 8 Gbyte of memory, 64K 1-bit processors and 2K floating point processors. The theoretical peak performance was 40 Gflop/s. Each physical processor emulated a number of virtual processors via a combination of the run-time system and a high-level language. The virtual processors could be transparently used through high-level languages like CM Fortran. The interconnect topology was a hypercube, and it supported both structured and unstructured communication patterns as well as global logical operations and global numerical operations. The nearest-neighbor communication (NEWS) had a cost per floating point value that was roughly two to three times more expensive compared with a floating point operation. Unstructured communication could cost several hundred times more.

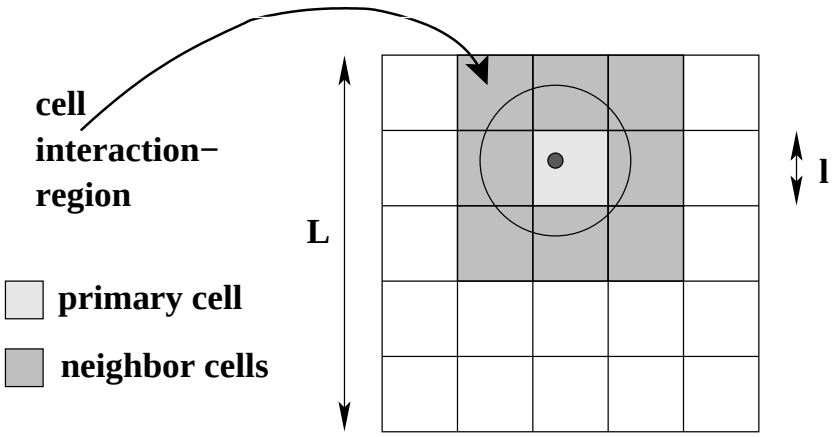


Figure 2.4. The simulation box with side length L, and the sub-cells with side length l. A particle's primary region ("home cell"), primary neighbor region and interaction sphere of radius $r_{cut} \leq l$.

2.3.2 Algorithm

In the link-cell algorithm a simulation box is divided into a number of equally sized cubic sub-cells. Let L be the simulation box-size. Choose the sub-cell size $l = \lfloor L/r_{cell} \rfloor$, where $r_{cell} = r_{cut} + r_{skin}$. The "skin" is a small fraction of potential the cut-off. This choice limits the search volume of possible interacting particles to the 26 neighboring cells and the particle's primary region ("home cell"). See Figure 2.4.

Using force symmetry, $\boldsymbol{F}_{ij} = -\boldsymbol{F}_{ji}$, we only need to consider 13 neighboring cells and the home cell. The 13 neighbors can be divided into 3 groups: 4 "point cells", 6 "line cells" and 3 "plane cells". The terms point, line and plane indicate what geometrical object the cell has in common with the home cell. Thus, the force acting on a particle consists of an *intra-cell* component and an *inter-cell* component.

The *inter-cell* component is calculated by a double do-loop over all the particles present in the cell as is shown in the following CM Fortran code example

```
c
c Zero local potential energy
c
      cm_pe = 0.0
      do i=1,imax-1
c
c        Compute inverse of distance squared between particle i and particle j.
c
         do j=i+1,imax
            delta1 = xc1(i,:,:,:) - xc1(j,:,:,:)
```

```
            delta2 = xc2(i,:,:,:) − xc2(j,:,:,:)
            delta3 = xc3(i,:,:,:) − xc3(j,:,:,:)
            r2     = 1.0 / (delta1 * delta1 + delta2 * delta2 + delta3 * delta3)
c
c           Within cut off radius.
c
            where (r2.lt.rssqi)
              r2 = 0.0
            end where
c
c           Calculate L-J potential; use result only within cut off radius.
c
            r6    = r2 * r2 * r2
            esr   = (b12 * r6 + a6) * r6
            fsr   = (b12_12 * r6 + a6_6) * r6 * r2
            cm_pe = cm_pe + esr
c
c           Accumulate scalar force in x,y and z direction.
c
            fc1(i,:,:,:) = fc1(i,:,:,:) + fsr * delta1
            fc1(j,:,:,:) = fc1(j,:,:,:) − fsr * delta1
            fc2(i,:,:,:) = fc2(i,:,:,:) + fsr * delta2
            fc2(j,:,:,:) = fc2(j,:,:,:) − fsr * delta2
            fc3(i,:,:,:) = fc3(i,:,:,:) + fsr * delta3
            fc3(j,:,:,:) = fc3(j,:,:,:) − fsr * delta3
          end do
        end do
c
c Total potential energy.
c
        pe    = pe + sum(cm_pe)
```

The *intra-cell* component of each particle consists of 13 components originating from the 13 neighbor cells of the home cell. For each of these 13 components the inter-cell calculation proceeds as in the intra-cell calculation, but with a few differences.

1. Before the double do-loop starts, the particles in each cell is sorted into two buffers: the green buffer and the red buffer.

2. Inside the first do-loop the position of a red particle is shifted to the current neighbor cell.

3. After the second do loop the force accumulated on the red particle is shifted back to its home cell.

4. Finally, when the double do-loop has been completed the forces on the red and green particles are accumulated.

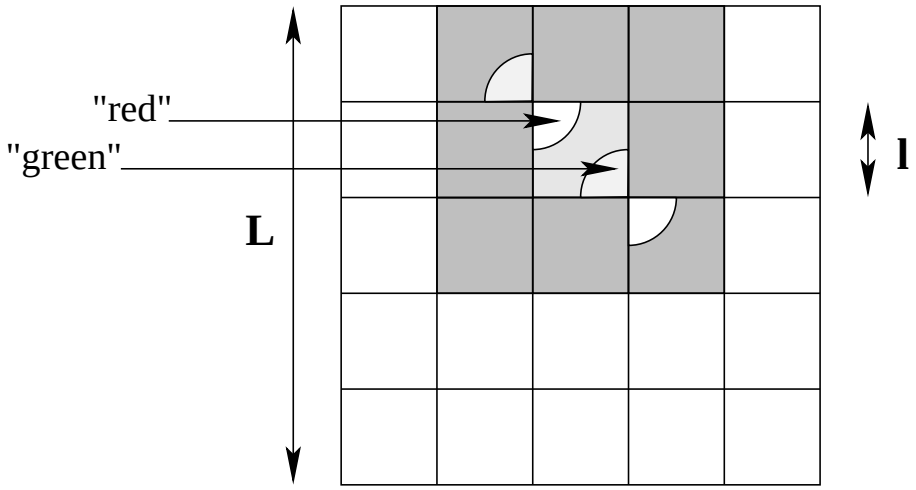


Figure 2.5. "Red" and "green" regions of the home cell when the neighbor cell is either a point cell or a line cell.

The sorting procedure is based only on the distance of each particle in its home cell to the geometrical object the home cell shares with the current neighbor cell: a point, a line or a plane. The particles which are within the cut-off radius of this object becomes red. The ones which are within the cut-off radius of the complementing cell becomes green. The complementing cell is the unique symmetrical counterpart found among the other 13 neighbor cells which are accounted for by force symmetry.

Thus the algorithm colors particles red or green using only local computations. Thereafter, red particles of the current neighbor are shifted to the home cell where they interact with the green particles in the home cell. Red forces are shifted back and accumulated. After all red and green particles have interacted, the total force on each particle has been calculated.

The speed and efficiency of our algorithm result from a consistent use of indirect addressing features (node-local gather and scatter), available directly via CM Fortran, in combination with an algorithm that can screen out interaction candidates with inter-particle distances longer than the cut-off of the interaction. The operation of screening out potential interaction candidates is formulated so that it does not require any inter-processor communication. In addition, it requires very little additional storage.

2.3.3 Efficiency

The timings in the paper show that the program scales linearly on a computer system with either 256 or 512 floating point processors and on system sizes up to about

5 million particles. This was the size of the hardware platform available to us at the time. With the combination of a scalable algorithm and the hardware architecture of the CM-2/CM-200, one could expect a fully configured system to be a factor of eight more powerful than the system we performed our calculations and timings on. In spite of what was later reported in the review article by Plimpton [80], it is clear from the timings reported in the paper, that at the time of publication this was probably one of the fastest MD programs for simulating simple particle systems with short-range interactions. It compared very favorably with a number of other algorithms implemented on similar type of SIMD hardware. In addition it performed on par or better with other algorithms implemented on MIMD hardware.

2.4 CMmd: Data-Parallel Long-Range Interactions

THE paper *"A Data-Parallel Molecular Dynamics Method for Liquids With Coulombic Interactions"* develops a parallel algorithm for simulating systems that include charges. The algorithm is an evolution of the overall approach for short-range interactions that we describe in §2.3. To include long-range interactions in our simulations we develop an efficient data-parallel formulation of the Ewald summation method. Moreover we demonstrate an efficient implementation.

We use fused salts as our model systems with the BHMFT[1] potential [203]: given two particles i and j, with respective charges q_i and q_j separated by a distance r_{ij}, the potential $V(r_{ij})$ is given by

$$V(r_{ij}) = \overbrace{q_i q_j \frac{e^2}{r_{ij}}}^{long-range} + \underbrace{A_{ij} \exp\left(B(\sigma_{ij} - r_{ij})\right) + \frac{C_{ij}}{r_{ij}^6} + \frac{D_{ij}}{r_{ij}^8}}_{short-range} \tag{2.2}$$

The first term in Equation (2.2) is the Coulombic interaction, the exponential term is the Born-Huggins repulsion and the third and fourth term represent, respectively, the dipole-dipole and dipole-quadrupole dispersion energies. Parameters are obtained from [204]. As noted in the above Equation, the Coulombic interaction is long-range, whereas the last three terms can be considered short-range.

2.4.1 Ewald summation

In periodic systems the Coulombic interaction is conditionally convergent. This means that the result of a direct summation of the interactions depend on the summation order. The Ewald summation method transforms the conditionally convergent sum into four parts, the real-space term, the reciprocal-space term, the self-interaction term, and the boundary term [58]. The Ewald sum for an electrically

[1] Born-Huggins-Mayer model parametrized by Fumi and Tosi.

neutral system of N charges with no external field can be written

$$V_{Coulomb} = V_R + V_F - V_S + V_B(\epsilon_r), \tag{2.3}$$

where

$$V_R = \frac{1}{4\pi\epsilon_0} \sum_{r_{ij}<r_{cut}} \frac{q_i q_j}{r_{ij}} \operatorname{erfc}(\alpha r_{ij}), \tag{2.4}$$

$$V_F = \frac{1}{2V\epsilon_0} \sum_{\boldsymbol{k}\neq 0} \frac{\exp\left(-|\boldsymbol{k}|^2/4\alpha^2\right)}{|\boldsymbol{k}|^2} Q_{\boldsymbol{k}} \overline{Q}_{\boldsymbol{k}}, \tag{2.5}$$

$$V_S = \frac{\alpha}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2. \tag{2.6}$$

In the above Equation, erfc is the complementary error function; $Q_{\boldsymbol{k}}$ denotes the charge density[2] at wave-vector $\boldsymbol{k}$

$$Q_{\boldsymbol{k}} = \sum_i q_i \exp(\imath \boldsymbol{k} \cdot \boldsymbol{r}_i), \qquad \boldsymbol{k} = \frac{2\pi}{L}(k_1, k_2, k_3),\ k_i \in Z,\ i = 1, 2, 3; \tag{2.7}$$

L is the simulation box length and $V = L^3$.

The last term in Equation (2.3) is a function of the dipole moment of the simulation cell and the dielectric constant of the surrounding media of the periodically replicated cells.

2.4.2 Algorithm

Except for the boundary term, the different terms in the Ewald method are functions of a free parameter, α—the Ewald convergence parameter. The self-interaction term, Equation (2.6), is a constant during a simulation and can be calculated at initialization.

Use the same accuracy requirement $\delta \ll 1$ for both the real-space term and reciprocal-space term. We employ the fact that $\operatorname{erfc}(x) \approx \exp(-x^2)$ when $x \gg 1$ and select a suitably large value for the Ewald convergence parameter. This makes the real-space term, Equation (2.4), sufficiently short-ranged, and it can be calculated together with the other short-range parts of the potential. The downside is that a large value of α forces the use of a large number of terms in the reciprocal-space term, Equation (2.5).

Adding in the real-space part of the Ewald summation only changes the routine which contains the force calculations. Real-space contributions are calculated with a data-parallel approach: it consists of a loop over the 13 neighboring[3] cells and the

[2] Also called the structure factor.

[3] Symmetry of force gives $(27-1)/2 = 13$.

"home-cell". First select particles which have a chance of interacting with particles in the neighbor cell, zero out force accumulators, change origin of selected particles and shift them to the neighboring cell. Now interactions can be calculated and the forces which have been found are sent back and accumulated. Calculated forces must be sent back and accumulated, because we only loop over half the 26 neighbor cells.

```
do group=1,NRNEIGH
   call color_group
   call zero_force_and_change_origo
   call shift_arrays_forward
   call real_space_force
   call shift_arrays_back
   call add_in_accumulated_force
enddo
```

Particles in all cells have their coordinates in relation to their current "home-cell". This arrangement allows for more uniform algorithms since particles in the interior will not have to be treated differently than particles belonging to a cell at the edge of the simulation box. In particular we can calculate distances directly, without having to use the minimum-image transformation.

The reciprocal-space part does not require any exchange of information between processors except for the global summation for each wave-vector, $\boldsymbol{k}$ (Equation (2.7)). For an efficient and memory conservative calculation of $Q_{\boldsymbol{k}}$ in Equation (2.5) we use an iterative scheme to find the values of $\exp(\imath \boldsymbol{k} \cdot \boldsymbol{r}_i)$ based on the formulas

$$\begin{aligned} \sin nx &= 2\sin(n-1)x\cos x - \sin(n-2)x, \\ \cos nx &= 2\cos(n-1)x\cos x - \cos(n-2)x. \end{aligned} \tag{2.8}$$

First, adjust global coordinates and calculate the initial terms of the iterative scheme in Equation (2.8). Calculate in one octant of reciprocal-space and use the symmetries of Equation (2.7) (internal, face, axis). End by switching back from a global to local coordinate system.

```
call ewald_setup
call ewald_internal
call ewald_face
call ewald_axis
call ewald_scale
```

2.4.3 Efficiency

To estimate the performance of our data-parallel formulation of the Ewald method we performed a number of benchmark runs on the CM-200. The system size ranged from $N = 32768$ to $N = 2097152$. The timings are reported in the paper and show

that we were attempting too large systems for realistic simulations. Additionally, the parameters we selected for the Ewald method were far from optimal [156].

Nevertheless, the performance of our algorithm compared quite well with the approach reported by Boltjes and de Leeuw [205]. In their paper several parallel algorithms for the BHMFT potential are presented; the largest system run was $N = 32768$ on a 32K CM-2 with a time per time-step of ≈ 570 seconds. For the same system size we reported a time of 28 seconds on a 4K CM-200. This is roughly a factor 20 faster on a computer with 1/4 the number of processors. Part of the difference in performance can be attributed to the faster clock speed of our Connection Machine and also much improved compilers. Still, at the time of publication this was probably one of the fastest MD programs for large simulations of simple particle systems with both long-range and short-range interactions.

2.5 *ENUF: an $\mathcal{O}(N \log N)$ Algorithm for Electrostatic Interactions*

REAL molecules in compounds are in continuous rapid motion under normal conditions. In liquids and gases they are colliding and interacting with each other at close distances and they also interact with external fields. In computer simulations of molecular systems, mechanistic ball and spring models are common to give a simple picture of atoms with specific sizes and masses bonded together with covalent bonds. Molecular equilibrium geometries and interactions are defined in so-called force fields where a somewhat arbitrary division between intramolecular and intermolecular interactions is made. Arbitrary because this partitioning is clearly a simplification of the more detailed and fundamental understanding furnished by a quantum mechanical description.

Intramolecular[4] interactions are normally described by bond-stretching, angle bending and torsional angle motion terms. These interactions involve the closest bonded atoms described by two-, three-, and four-body terms, respectively. The bond and angle terms are normally given as harmonic wells while the torsion term is most often expressed as a Fourier sum.

Intermolecular[5] interactions are those between separate molecules but also include all interactions within the same molecule beyond the bonded interactions. Non-bonded interactions are further divided between short-ranged and long-ranged interactions. The short-range interactions mimic the van der Waals type of forces. The long-range interactions are electrostatic interactions. These approximate the electron distributions around atoms by fixed point charges. Interactions are treated using Coulomb's law.

The short-range interactions are by definition such that their effective range is limited to within a specified cut-off. By assuming a uniform density on scales larger than the cut-off, error terms may be approximated and correction terms

[4]Often called *bonded interactios.*

[5]Often called *non-bonded* interactions.)

can be applied [33, 32, 29]. Thus the short-range interactions can be accurately approximated by truncation in most calculations.

Artificially collecting and dividing the diffuse and fluctuating electron densities inside and around molecules on single atomic sites is a crude but conceptually simple approximation, since Coulomb's law can be invoked. However, this simplification comes at a price since the interactions between point charges exist over very long distances. Furthermore, the long-range can not be truncated without introducing simulation artifacts [206, 207, 55, 56].

As the system size grows, calculating the electrostatic interactions becomes the major computational bottleneck. Methods based on Ewald summation [57, 58] are still considered as the most reliable choice and a large variety of schemes to compute them in computer simulations have been proposed [208, 209, 210, 211, 212, 213, 214]. There is also a multitude of alternative methods for representing electrostatic interactions. Examples are methods based on a cut-off [63, 64, 65], tree and multipole based methods [73, 215, 216, 217, 218, 219], multigrid methods [70, 220, 72], reaction field methods [221, 155, 222], the particle mesh method [1, 223], and the isotropic sum method [66].

In the paper *"Ewald Summation Based on Nonuniform Fast Fourier Transform"* we present a novel approach that scales as $\mathcal{O}(N \log N)$, where N is the number if electrostatic interaction sites in the system. Our method combines the traditional Ewald summation technique with the nonuniform Fast Fourier transform to calculate electrostatic energies and forces in molecular computer simulations.

An essential ingredient in any method is its name. We propose the acronym ENUF—Ewald summation using NonUniform fast Fourier transform. In the paper we show that ENUF is an easy-to-implement, practical, and efficient method for calculating electrostatic interactions. Energy and momentum is conserved to floating point accuracy. By a suitable choice of parameters, ENUF can be made to behave as traditional Ewald summation but at the same time give a computational complexity of $\mathcal{O}(N \log N)$. Weighing all these properties together, we believe that ENUF should be an attractive alternative in simulations where the high accuracy of Ewald summation is desired.

2.5.1 Ewald summation

We start by describing a model system of charged particles which captures the most salient features of electrostatic interactions in general MD systems. The electrostatic potential of a system with periodic boundary conditions (PBC) is first stated; we follow with the manipulation of the basic formulas to the form in which they are commonly written; this Section ends with a summary of expressions for both energy and forces.

The motivation for this recapitulation of known results is to prepare the ground for the next stage (§2.5.3) in which we show how ENUF can be developed.

Model system

Consider a cubic simulation box with edge length L, containing N charged particles, each with a charge q_i, located at $\boldsymbol{r}_i$. Periodic boundary conditions in a system without cut-off is represented by replicating the simulation box in all directions. The total electrostatic potential energy of the charge-charge interactions is then given by

$$\begin{aligned} U_{qq} &= \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{q_i q_j}{|\boldsymbol{r}_{ij}|} + \frac{1}{2} \sum_{\mathbf{n} \neq 0} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{|\boldsymbol{r}_{ij} + \mathbf{n}L|} \\ &= \frac{1}{2} \sum_{\mathbf{n}}^{\dagger} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{|\boldsymbol{r}_{ij} + \mathbf{n}L|}, \end{aligned} \tag{2.9}$$

where $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$, and $|\boldsymbol{r}|$ denotes the length (2-norm) of the vector $\boldsymbol{r}$. Because of the long-range nature of the electrostatic interactions, U_{qq} includes contributions from all replicas, but exclude self-interactions; this is expressed in the triple sum in Equation (2.9): the outer sum is taken over all integer vectors $\mathbf{n} = (n_1, n_2, n_3)$ such that each $n_i \in Z$; the $\dagger$ symbol on the first summation sign in Equation (2.9), indicates that the self-interaction terms should not be included, *i.e.*, when $\mathbf{n} = 0$ then the $i = j$ terms are omitted.

The sum in Equation (2.9) is not an absolutely convergent series, but rather conditionally convergent.[6] As a consequence, the order of summation affects the value of the series. In fact it was discovered by Riemann that any conditionally convergent series of real terms can be rearranged to yield a series which converges to any prescribed sum [224, Section 8.18]. In a sense, this is a situation very similar to the case when a linear equation has an infinite number of solutions because it is under-determined; by adding a set of conditions a unique solution may be defined. For the specific case of Equation (2.9), a physically relevant summation order has to be prescribed and the boundary conditions of the surrounding media have to be specified.

The lattice sum of Equation (2.9) can be calculated by a method that was first developed in 1921 by P.P Ewald [57] to calculate lattice potentials in solids. There are several different derivations of the Ewald summation method for use in the context of Molecular Dynamics that are more recent and easily accessible; a small selection is given by [57, 58, 225, 29]. In the discussion below I mainly follow the work of de Leeuw *et al.* [58, 156, 226].

In [58] de Leeuw *et al.* developed a technique using convergence factors that transforms the sum of a conditionally convergent series into a series with a well defined sum. Furthermore, they showed that applying a specific convergence factor is equivalent to a certain summation order. Assuming an overall charge neutral system, $\sum_i q_i = 0$, and summing the terms in Equation (2.9) over all integer vectors

[6] A series $\sum a_i$ is called absolutley convergent if $\sum |a_i|$ converges. When $\sum a_i$ converges and $\sum |a_i|$ diverges, the series is called conditionally convergent.

$\boldsymbol{n}$ in concentric spherical order, they showed that the electrostatic potential energy can be written as

$$U_{qq}(\epsilon_r = 1) = \sum_{1 \le i < j \le N} \frac{q_i q_j}{L} \Psi(\frac{r_{ij}}{L}) + \frac{\xi}{2L} \sum_{i=1}^{N} q_i^2 + \frac{2\pi}{(2\epsilon_r + 1)L^3} |\sum_i q_i \boldsymbol{r}_i|^2, \quad (2.10)$$

when the surrounding media of the periodically replicated cell is vacuum ($\epsilon_r = 1$) and distances are calculated with the minimum image convention. The function Ψ and number ξ are defined below in Equations (2.14) and (2.13), respectively. When the surrounding media is a conductor ($\epsilon_r = \infty$), the energy can be written as

$$U_{qq}(\epsilon_r = \infty) = \sum_{1 \le i < j \le N} \frac{q_i q_j}{L} \Psi(\frac{r_{ij}}{L}) + \frac{\xi}{2L} \sum_{i=1}^{N} q_i^2 \quad (2.11)$$

$$= U_{qq}(\epsilon_r = 1) - \frac{2\pi}{(2\epsilon_r + 1)L^3} |\sum_i q_i \boldsymbol{r}_i|^2. \quad (2.12)$$

From Equation (2.12) it is clear that the boundary conditions, vacuum or conductor, have an effect on the energy of the system. Depending on the simulated system and the properties of interest, the choice of boundary conditions can affect the results obtained, *e.g.*, [227, 228].

The number ξ used above in Equations (2.10) and (2.11) is defined as

$$\xi = \sum_{\boldsymbol{n} \neq 0} \frac{\text{erfc}(\alpha |\boldsymbol{n}|)}{|\boldsymbol{n}|} + \frac{1}{\pi} \sum_{\boldsymbol{n} \neq 0} \frac{e^{-\pi^2 |\boldsymbol{n}|^2 / \alpha^2}}{|\boldsymbol{n}|^2} - \frac{2\alpha}{\sqrt{\pi}} \quad (2.13)$$

and the function Ψ is given by

$$\Psi(\boldsymbol{r}) = \sum_{\boldsymbol{n}} \frac{\text{erfc}(\alpha |\boldsymbol{r} + \boldsymbol{n}|)}{|\boldsymbol{r} + \boldsymbol{n}|} + \frac{1}{\pi} \sum_{\boldsymbol{n} \neq 0} \frac{\exp(2\pi \imath \boldsymbol{n} \cdot \boldsymbol{r} - \pi^2 |\boldsymbol{n}|^2 / \alpha^2)}{|\boldsymbol{n}|^2}, \quad (2.14)$$

with the error functions $\text{erfc}(x)$ and $\text{erf}(x)$ defined as

$$\text{erfc}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$$
$$\text{erf}(x) = 1 - \text{erfc}(x)$$

Equations (2.10) and (2.11) are not in a form that is appropriate for efficient numerical calculations and in the case of Molecular Dynamics simulation we also need expressions for the forces. To arrive at a more suitable form we make the necessary anlysis for the electrostatic energy and forces in the following Sections.

Energies in Ewald summation

To rearrange and expand Equation (2.10) we first insert $\boldsymbol{r}_{ij}/L$ in Equation (2.14)

$$\begin{aligned}\Psi(\boldsymbol{r}_{ij}/L) &= \sum_{\boldsymbol{n}} \frac{\operatorname{erfc}(\alpha|\frac{\boldsymbol{r}_{ij}}{L}+\boldsymbol{n}|)}{|\frac{\boldsymbol{r}_{ij}}{L}+\boldsymbol{n}|} + \frac{1}{\pi}\sum_{\boldsymbol{n}\neq 0}\frac{\exp(2\pi\imath\boldsymbol{n}\cdot\frac{\boldsymbol{r}_{ij}}{L}-\pi^2|\boldsymbol{n}|^2/\alpha^2)}{|\boldsymbol{n}|^2} \\ &= L\sum_{\boldsymbol{n}}\frac{\operatorname{erfc}(\frac{\alpha}{L}|\boldsymbol{r}_{ij}+L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|} \\ &\quad + \frac{1}{\pi}\sum_{\boldsymbol{n}\neq 0}\frac{\exp(\frac{2\pi i}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}-\pi^2|\boldsymbol{n}|^2/\alpha^2)}{|\boldsymbol{n}|^2}.\end{aligned} \tag{2.15}$$

Next we rescale α by making the substitution $\alpha \to \alpha L$ in Equations (2.15) and (2.13) to get

$$\begin{aligned}\Psi(\boldsymbol{r}_{ij}/L) &= L\sum_{\boldsymbol{n}}\frac{\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}+L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|} \\ &\quad + \frac{1}{\pi}\sum_{\boldsymbol{n}\neq 0}\frac{\exp(\frac{2\pi i}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2)}{|\boldsymbol{n}|^2}\end{aligned} \tag{2.16}$$

and

$$\xi = \sum_{\boldsymbol{n}\neq 0}\frac{\operatorname{erfc}(\alpha L|\boldsymbol{n}|)}{|\boldsymbol{n}|} + \frac{1}{\pi}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} - \frac{2\alpha L}{\sqrt{\pi}}. \tag{2.17}$$

Inserting Equations (2.16) and (2.17) into Equation (2.10) we get

$$\begin{aligned}U_{qq}(\epsilon_r=1) &= \sum_{1\le i<j\le N} q_iq_j\Bigg\{\sum_{\boldsymbol{n}}\frac{\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}+L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|} \\ &\quad + \frac{1}{\pi L}\sum_{\boldsymbol{n}\neq 0}\frac{\exp(\frac{2\pi i}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2)}{|\boldsymbol{n}|^2}\Bigg\} \\ &\quad + \frac{1}{2L}\sum_i q_i^2\Bigg[\sum_{\boldsymbol{n}\neq 0}\frac{\operatorname{erfc}(\alpha L|\boldsymbol{n}|)}{|\boldsymbol{n}|} + \frac{1}{\pi}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} - \frac{2\alpha L}{\sqrt{\pi}}\Bigg] \\ &\quad + \frac{2\pi}{(2\epsilon_r+1)L^3}|\sum_i q_i\boldsymbol{r}_i|^2\end{aligned} \tag{2.18}$$

Note that the summation above is for $i<j$ in the first sum. We make further simplifications by studying the terms on the right hand side of Equation (2.18) for $n=0$ and $n\neq 0$.

When $n = 0$ we have the following terms

$$\sum_{1\le i<j\le N} q_i q_j \frac{\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}|)}{|\boldsymbol{r}_{ij}|} - \frac{\alpha}{\sqrt{\pi}}\sum_i {q_i}^2 + \frac{2\pi}{(2\epsilon_r+1)L^3}|\sum_i q_i \boldsymbol{r}_i|^2$$
$$= \frac{1}{2}\sum_{i,j}^{\dagger} \frac{q_i q_j}{|\boldsymbol{r}_{ij}|}\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}|) - \frac{\alpha}{\sqrt{\pi}}\sum_i {q_i}^2 + \frac{2\pi}{(2\epsilon_r+1)L^3}|\sum_i q_i \boldsymbol{r}_i|^2 \quad (2.19)$$

with the terms independent of $\boldsymbol{n}$ included; the † symbol indicates that the $i = j$ terms are excluded from the daggered sum.

When $n \neq 0$ we get

$$\sum_{1\le i<j\le N} q_i q_j \Bigg\{ \sum_{\boldsymbol{n}\neq 0} \frac{\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}+L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|}$$
$$+ \frac{1}{\pi L}\sum_{\boldsymbol{n}\neq 0} \frac{\exp(\frac{2\pi i}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij} - \pi^2|\boldsymbol{n}|^2/(\alpha L)^2)}{|\boldsymbol{n}|^2}\Bigg\}$$
$$+ \frac{1}{2L}\sum_i q_i^2 \Bigg\{ \sum_{\boldsymbol{n}\neq 0} \frac{\operatorname{erfc}(\alpha L|\boldsymbol{n}|)}{|\boldsymbol{n}|} + \frac{1}{\pi}\sum_{\boldsymbol{n}\neq 0} \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\Bigg\}$$
$$= \frac{1}{2}\sum_{i,j} q_i q_j \sum_{\boldsymbol{n}\neq 0}\Bigg\{ \frac{\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}+L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|} + \frac{1}{\pi L}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij})\Bigg\}. \quad (2.20)$$

The factor 1/2 in Equation (2.20) comes from changing the summation from $i < j$ to all pairs i and j, and using the symmetry induced by $\boldsymbol{r}_{ij} = -\boldsymbol{r}_{ji}$ and $\pm\boldsymbol{n}$.

By combining Equation (2.19) and (2.20) we identify the real-space term, U_{qq}^{real}; the reciprocal-space term, U_{qq}^{recip}; the self-interaction term, U_{qq}^{self}; and the boundary-condition term, U_{qq}^{bc}. The real-space term is given by

$$U_{qq}^{\text{real}} = \sum_{1\le i<j\le N} q_i q_j \frac{\operatorname{erfc}(\alpha|\boldsymbol{r}_i j|)}{|\boldsymbol{r}_i j|} + \sum_{i<j}\sum_{\boldsymbol{n}\neq 0} \frac{\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}+L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|}$$
$$+ \frac{1}{2}\sum_i q_i^2 \sum_{\boldsymbol{n}\neq 0} \frac{\operatorname{erfc}(\alpha|L\boldsymbol{n}|)}{|L\boldsymbol{n}|}$$
$$= \frac{1}{2}\sum_{\boldsymbol{n}}^{\dagger}\sum_{i,j} \frac{q_i q_j}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|}\operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}+L\boldsymbol{n}|), \quad (2.21)$$

and the reciprocal-space term is

$$U_{qq}^{\text{recip}} = \frac{1}{\pi L}\sum_{i<j}\sum_{\boldsymbol{n}\neq 0} q_i q_j \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}) + \frac{1}{2\pi L}\sum_{i}\sum_{\boldsymbol{n}\neq 0} q_i q_i \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}. \tag{2.22}$$

However, with the symmetries generated by $\boldsymbol{r}_{ij} = -\boldsymbol{r}_{ji}$ and $\pm\boldsymbol{n}$, we get

$$= \frac{1}{2\pi L}\sum_{i\neq j}\sum_{\boldsymbol{n}\neq 0} q_i q_j \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}) + \frac{1}{2\pi L}\sum_{i}\sum_{\boldsymbol{n}\neq 0} q_i q_i \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} \tag{2.23}$$

$$= \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\sum_{i,j} q_i q_j \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}). \tag{2.24}$$

Furthermore, we have

$$U_{qq}^{\text{self}} = \frac{\alpha}{\sqrt{\pi}}\sum_i {q_i}^2 \tag{2.25}$$

$$U_{qq}^{\text{bc}} = \frac{2\pi}{(2\epsilon_r+1)L^3}|\sum_i q_i\boldsymbol{r}_i|^2, \tag{2.26}$$

and finally

$$U_{qq}(\epsilon_r = 1) = U_{qq}^{\text{real}} + U_{qq}^{\text{recip}} - U_{qq}^{\text{self}} + U_{qq}^{\text{bc}}. \tag{2.27}$$

The reciprocal-space part, Equation (2.24), can be expanded in two different forms. The first form is in terms of the structure factor $S(\boldsymbol{n})$,

$$S(\boldsymbol{n}) = \sum_i q_i \exp(-\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i), \tag{2.28}$$

and is given by

$$U_{qq}^{\text{recip}} = \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\sum_{i,j} q_i q_j \exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot(\boldsymbol{r}_i-\boldsymbol{r}_j))$$

$$= \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\sum_i q_i \exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)\sum_j q_j \exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot(-\boldsymbol{r}_j))$$

$$= \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} S(-\boldsymbol{n})S(\boldsymbol{n}). \tag{2.29}$$

Now for a fixed $\boldsymbol{n}$, the structure factor $S(\boldsymbol{n})$ is just a complex number a, and the simple fact that $a\overline{a} = Re(a)^2 + Im(a)^2$, gives the real form of Equation (2.24)

$$\begin{aligned} U_{qq}^{\text{recip}} &= \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\sum_{i,j} q_i q_j \exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot(\boldsymbol{r}_i - \boldsymbol{r}_j)) \\ &= \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\Big[|\sum_i q_i \cos(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)|^2 \\ &\quad + |\sum_i q_i \sin(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)|^2\Big]. \end{aligned} \tag{2.30}$$

The last form is the most common point-of-departure when implementing the reciprocal-space part. The first form is used in our fast approach to calculating the reciprocal-space part.

Forces in Ewald summation

Now that we have calculated the electrostatic energy of the system we can easily compute the electrostatic forces $\boldsymbol{F}_i$ that act on each particle i. Splitting the forces in the same way as we have split the energy and using Equation (2.27) we get the total electrostatic force by finding the negative of the gradient of the electrostatic energy

$$\begin{aligned} F_i &= -\nabla_i U_{qq} = -\nabla_i\Big[U_{qq}^{\text{real}} + U_{qq}^{\text{recip}} - U_{qq}^{\text{self}} + U_{qq}^{\text{bc}}\Big] \\ &= F_i^{\text{real}} + F_i^{\text{recip}} - 0 + F_i^{\text{bc}}. \end{aligned} \tag{2.31}$$

Where the subscript i on the ∇ operator indicates that we take the partial derivatives with respect to the position of particle $\boldsymbol{r}_i$ and the 0 in Equation (2.31) comes from the self-interaction term (Equation (2.25)) being independent of $\boldsymbol{r}_i$. Before we do this calculation we note a couple basic, but helpful, formulas for calculating derivatives

$$\begin{aligned} \nabla_i \boldsymbol{n}\cdot\boldsymbol{r}_{ij} &= \nabla_i \boldsymbol{n}\cdot(\boldsymbol{r}_i - \boldsymbol{r}_j) = \boldsymbol{n} \\ \nabla_i \frac{1}{|\boldsymbol{r}_{ij}|} &= -\frac{\boldsymbol{r}_{ij}}{|\boldsymbol{r}_{ij}|^3} \\ \nabla_i |\boldsymbol{r}_{ij}| &= \frac{\boldsymbol{r}_{ij}}{|\boldsymbol{r}_{ij}|} \\ \nabla_i \operatorname{erfc}(\alpha|\boldsymbol{r}_{ij}|) &= -\frac{2\alpha}{\sqrt{\pi}}\frac{\boldsymbol{r}_{ij}}{|\boldsymbol{r}_{ij}|}\exp(-\alpha^2|\boldsymbol{r}_{ij}^2|) \\ \nabla_i \exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ji}) &= -\boldsymbol{n}\frac{2\pi\imath}{L}\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ji}) \\ \nabla_i \exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}) &= \boldsymbol{n}\frac{2\pi\imath}{L}\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}) \end{aligned} \tag{2.32}$$

With the formulas on the facing page, it is straightforward to find the different terms of F_i. The contribution from the real-space term, F_i^{real}, becomes

$$\begin{aligned}
F_i^{\text{real}} &= -\nabla_i U_{qq}^{\text{real}} \\
&= -\nabla_i \left[\frac{1}{2} \sum_{\boldsymbol{n}}^{\dagger} \sum_{i,j} \frac{q_i q_j}{|\boldsymbol{r}_{ij} + L\boldsymbol{n}|} \operatorname{erfc}(\alpha |\boldsymbol{r}_{ij} + L\boldsymbol{n}|) \right] \\
&= -\frac{2}{2} \sum_{\boldsymbol{n}}^{\dagger} \sum_{j} q_i q_j \Big[-\frac{(\boldsymbol{r}_{ij} + L\boldsymbol{n})}{|\boldsymbol{r}_{ij} + L\boldsymbol{n}|^3} \operatorname{erfc}(\alpha |\boldsymbol{r}_{ij} + L\boldsymbol{n}|) \\
&\quad - \frac{(\boldsymbol{r}_{ij} + L\boldsymbol{n})}{|\boldsymbol{r}_{ij} + L\boldsymbol{n}|^2} \frac{2\alpha}{\sqrt{\pi}} \exp(-\alpha^2 |\boldsymbol{r}_{ij} + L\boldsymbol{n}|^2) \Big] \\
&= \sum_{\boldsymbol{n}}^{\dagger} \sum_{j} q_i q_j \frac{(\boldsymbol{r}_{ij} + L\boldsymbol{n})}{|\boldsymbol{r}_{ij} + L\boldsymbol{n}|^2} \left[\frac{\operatorname{erfc}(\alpha |\boldsymbol{r}_{ij} + L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij} + L\boldsymbol{n}|} \right. \\
&\quad \left. + \frac{2\alpha}{\sqrt{\pi}} \exp(-\alpha^2 |\boldsymbol{r}_{ij} + L\boldsymbol{n}|^2) \right], \qquad (2.33)
\end{aligned}$$

such that when $n \neq 0$ include all j and otherwise only $j \neq i$ – "the daggered saviour". Equation (2.30) on the preceding page is convenient to use when calculating the reciprocal-space contribution because it is expressed in terms of charge locations $\boldsymbol{r}_i$ rather than relative distances $\boldsymbol{r}_{ij}$. Thus the reciprocal-space force is given by

$$\begin{aligned}
F_i^{\text{recip}} &= -\nabla_i U_{qq}^{\text{recip}} \\
&= -\nabla_i \left[\frac{1}{2\pi L} \sum_{\boldsymbol{n} \neq 0} \frac{e^{-\pi^2 |\boldsymbol{n}|^2 / (\alpha L)^2}}{|\boldsymbol{n}|^2} \left[|\sum_i q_i \cos(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i)|^2 \right. \right. \\
&\quad \left. \left. + |\sum_i q_i \sin(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i)|^2 \right] \right] \\
&= -\frac{1}{2\pi L} \sum_{\boldsymbol{n} \neq 0} \frac{e^{-\pi^2 |\boldsymbol{n}|^2 / (\alpha L)^2}}{|\boldsymbol{n}|^2} \frac{4\pi}{L} \boldsymbol{n} \left[-q_i \sin(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i) \sum_j q_j \cos(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_j) \right. \\
&\quad \left. + q_i \cos(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i) \sum_j q_j \sin(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_j) \right] \\
&= \frac{2q_i}{L^2} \sum_{\boldsymbol{n} \neq 0} \boldsymbol{n} \frac{e^{-\pi^2 |\boldsymbol{n}|^2 / (\alpha L)^2}}{|\boldsymbol{n}|^2} \left[\sin(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i) \sum_j q_j \cos(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_j) \right. \\
&\quad \left. - \cos(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i) \sum_j q_j \sin(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_j) \right]
\end{aligned}$$

$$= \frac{2q_i}{L^2}\sum_{\boldsymbol{n}\neq 0}\boldsymbol{n}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\Big[\sin(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)\,Re(S(\boldsymbol{n})) + \cos(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)\,Im(S(\boldsymbol{n}))\Big] \tag{2.34}$$

Finally, the contribution that depends on the boundary condition

$$\begin{aligned} F_i^{\mathrm{bc}} &= -\nabla_i \frac{2\pi}{(2\epsilon_r+1)L^3}|\sum_i q_i\boldsymbol{r}_i|^2 \\ &= -\frac{4\pi}{(2\epsilon_r+1)L^3}q_i\sum_j q_j\boldsymbol{r}_j. \end{aligned} \tag{2.35}$$

Summary of energy and forces in Ewald summation

Consider a periodically replicated system, with the central box consisting of N point charges q_i, such that

$$\sum_i q_i = 0. \tag{2.36}$$

Assume that the surrounding media at the boundary of the periodically replicated system is a uniform dielectric with dielectric constant ϵ_r. The cubic box has edge length L; each charge q_i is located at $\boldsymbol{r}_i$, and distances are calculated with the minimum image convention. After expansion and rearrangements of Equation (2.10), rescaling $\alpha \to \alpha L$ and using symmetries induced by $\boldsymbol{r}_{ij} = -\boldsymbol{r}_{ji}$ and $\pm\boldsymbol{n}$, the total electrostatic energy of the system can be written as

$$U_{qq}(\epsilon_r = 1) = U_{qq}^{\mathrm{real}} + U_{qq}^{\mathrm{recip}} - U_{qq}^{\mathrm{self}} + U_{qq}^{\mathrm{bc}} \tag{2.37}$$

with the different terms given by

$$U_{qq}^{\mathrm{real}} = \frac{1}{2}\sum_{\boldsymbol{n}}^{\dagger}\sum_{i,j}\frac{q_iq_j}{|\boldsymbol{r}_{ij}+L\boldsymbol{n}|}\,\mathrm{erfc}(\alpha|\boldsymbol{r}_{ij}+L\boldsymbol{n}|), \tag{2.38}$$

$$U_{qq}^{\mathrm{recip}} = \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\sum_{i,j}q_iq_j\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_{ij}) \tag{2.39}$$

$$= \frac{1}{2\pi L}\sum_{\boldsymbol{n}\neq 0}\frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2}\Big[|\sum_i q_i\cos(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)|^2 + |\sum_i q_i\sin(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)|^2\Big] \tag{2.40}$$

$$= \frac{1}{2\pi L} \sum_{\boldsymbol{n} \neq 0} \frac{e^{-\pi^2 |\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} S(-\boldsymbol{n}) S(\boldsymbol{n}), \tag{2.41}$$

$$U_{qq}^{\text{self}} = \frac{\alpha}{\sqrt{\pi}} \sum_i {q_i}^2, \tag{2.42}$$

$$U_{qq}^{\text{bc}} = \frac{2\pi}{(2\epsilon_r + 1)L^3} |\sum_i q_i \boldsymbol{r}_i|^2. \tag{2.43}$$

Note that $\alpha > 0$ is a free parameter. The structure factor $S(n)$ is defined as

$$S(\boldsymbol{n}) = \sum_i q_i \exp(-\frac{2\pi \imath}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i). \tag{2.44}$$

The total electrostatic force, F_i, on each particle is

$$F_i = F_i^{\text{real}} + F_i^{\text{recip}} + F_i^{\text{bc}}. \tag{2.45}$$

Each of the force terms given by

$$F_i^{\text{real}} = \sum_{\boldsymbol{n}}{}^{\dagger} \sum_j q_i q_j \frac{(\boldsymbol{r}_{ij} + L\boldsymbol{n})}{|\boldsymbol{r}_{ij} + L\boldsymbol{n}|^2} \left[\frac{\text{erfc}(\alpha |\boldsymbol{r}_{ij} + L\boldsymbol{n}|)}{|\boldsymbol{r}_{ij} + L\boldsymbol{n}|} + \frac{2\alpha}{\sqrt{\pi}} \exp(-\alpha^2 |\boldsymbol{r}_{ij} + L\boldsymbol{n}|^2) \right], \tag{2.46}$$

$$F_i^{\text{recip}} = \frac{2q_i}{L^2} \sum_{\boldsymbol{n} \neq 0} \boldsymbol{n} \frac{e^{-\pi^2 |\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} \left[\sin(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i) \sum_j q_j \cos(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_j) - \cos(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_i) \sum_j q_j \sin(\frac{2\pi}{L} \boldsymbol{n} \cdot \boldsymbol{r}_j) \right], \tag{2.47}$$

$$F_i^{\text{bc}} = -\frac{4\pi}{(2\epsilon_r + 1)L^3} q_i \sum_j q_j \boldsymbol{r}_j. \tag{2.48}$$

The positive number α, the so called Ewald convergence parameter, is chosen for computational convenience. Note that $\text{erfc}(x) \approx e^{-x^2}$ for large values of x. By choosing α large enough in Equation (2.38) on the facing page, we can ensure that the only terms that contribute in the real-space sum is when $\boldsymbol{n} = 0$. This may be expressed so that all terms with $|\boldsymbol{n}| < n_{cut}$ should be included.

Choose a cut-off in both real-space and reciprocal-space so that the neglected terms in the real-space and reciprocal-space parts are of the same order δ, or less. The truncation in real-space implies that a sufficient number of terms must be included in the reciprocal-space sums, Equation (2.41).

Given a required accuracy δ, n_{cut} is fixed by

$$e^{-\pi^2 {n_{cut}}^2/(\alpha L)^2} \leq \delta : n_{cut} \geq \frac{\alpha L}{\pi}\sqrt{-\log(\delta)}, \tag{2.49}$$

and r_{cut} is determined by

$$\operatorname{erfc}(\alpha r_{cut}) \approx e^{-\alpha^2 r_{cut}^2} \leq \delta : r_{cut} = \frac{\pi n_{cut}}{\alpha^2 L}. \tag{2.50}$$

Note that there are 2 conditions and 4 parameters. With a required δ we may just as well pick a suitable value for n_{cut} and let the above 2 Equations determine α and r_{cut}.

With an optimal choice of parameters the computational effort of the Ewald method becomes $\mathcal{O}(N^{3/2})$ [156, 229] giving a considerable improvement over the $\mathcal{O}(N^2)$ computational complexity implied by the "infinite" reach of the Coulomb interactions.

2.5.2 Discrete Fourier transforms for non-equispaced data

The fast Fourier transform for nonuniform data-points (NFFT) [230] is a generalization of the FFT [231]. Several similar approaches have been proposed; some examples are [232, 233, 234, 235, 236, 237, 238, 239, 240] with comparisons in [241, 237, 242].

The basic idea of NFFT is to combine the standard FFT and linear combinations of a window function that is well localized in both the spatial/time domain and the frequency domain. A controlled approximation using a cut-off in the frequency domain and a limited number of terms in the spatial/time domain results in an aliasing error and a truncation error, respectively. The aliasing errors is controlled by the oversampling factor σ_s, and the truncation error is controlled by the number of terms, m, in the spatial/time approximation. For a number of window functions (Gaussian, B-spline, Sinc-power, Kaiser-Bessel), it has been shown that for a fixed oversampling factor, $\sigma_s > 1$, the error decays exponentially with m [243].

Problem definition

We wish to calculate the discrete Fourier transform for nonequispaced data (NDFT). The problem can be stated as follows. For a finite number of given Fourier coefficients $\hat{\boldsymbol{f}}_{\boldsymbol{k}} \in \mathcal{C}$ with $\boldsymbol{k} \in I_M$ we want to evaluate the trigonometric polynomial

$$f(x) = \sum_{\boldsymbol{k} \in I_M} \hat{\boldsymbol{f}}_{\boldsymbol{k}} \exp(-2\pi\imath \boldsymbol{k}\boldsymbol{x})$$

at each of the given nonequispaced points $\boldsymbol{x}_j \in \mathcal{D}^d, j = 0, \ldots, N-1$. In the literature, *points* are often called *knots*. We use the two terms synonymously.

Obviously, the details of an NDFT depend on the definitions of a sampling set for knots, $\mathcal{D}^d$, and an index space I_M. More in-depth discussions and further details can be found in [243, 244]. The presentation that follows is mainly drawn from these sources.

Underlying concepts

Consider a d-dimensional domain $\mathcal{D}^d$ in which the set of nonequispaced knots, or data points, are located. Let

$$\mathcal{D}^d := \{\boldsymbol{x} = (x_t)_{t=0,\ldots,d-1} \in \mathcal{R}^d \ : \ -\frac{1}{2} \le x_t < \frac{1}{2},\ t = 0, \ldots, d-1\}, \qquad (2.51)$$

and the set of N data points

$$\mathcal{X} := \{\boldsymbol{x}_j \in \mathcal{D}^d \ : \ j = 0, \ldots, N-1\}.$$

For the application we have in mind d is usually 2 or 3. Let $\mathcal{F}$ be a function space of trigonometric polynomials with degree M_t $(t = 0, \ldots, d-1)$ in dimension t; the function space $\mathcal{F}$ can be defined as

$$\mathcal{F} := \{f : \mathcal{D}^d \mapsto \mathcal{C} \text{ such that } f \in span(e^{-2\pi\imath \boldsymbol{k}\cdot} \ : \ \boldsymbol{k} \in I_M)\}.$$

The dimension of this function space is $dim(\mathcal{F}) = M_\Pi$, where $M_\Pi = \prod_{t=0}^{d-1} M_t$. The frequencies $\boldsymbol{k} \in I_M$ with the index set I_M are such that

$$I_M := \{\boldsymbol{k} = (k_t)_{t=0,\ldots,d-1} \in \mathcal{Z}^d \ : \ -\frac{M_t}{2} \le k_t < \frac{M_t}{2},\ t = 0, \ldots, d-1\}. \qquad (2.52)$$

Matrix-vector formulation

With these preliminary defintions we carry on with the problem of calculating the discrete Fourier transform for nonequispaced data. For a finite number of given Fourier coefficients $\hat{\boldsymbol{f}}_{\boldsymbol{k}} \in \mathcal{C}$ with $\boldsymbol{k} \in I_M$ we want to evaluate the trigonometric polynomial

$$f(x) = \sum_{\boldsymbol{k} \in I_M} \hat{\boldsymbol{f}}_{\boldsymbol{k}} \exp(-2\pi\imath \boldsymbol{k}\boldsymbol{x}) \qquad (2.53)$$

at each of the given nonequispaced knots in $\mathcal{X}$. Where the product $\boldsymbol{k}\boldsymbol{x}$ is the usual scalar product of the two vectors $\boldsymbol{k}$ and $\boldsymbol{x}$, $\boldsymbol{k}\boldsymbol{x} := k_0 x_0 + \ldots + k_{d-1} x_{d-1}$. Consequently, for each $\boldsymbol{x}_j \in \mathcal{X}$, we evaluate

$$\boldsymbol{f}_j = f(\boldsymbol{x}_j) := \sum_{\boldsymbol{k} \in I_{\boldsymbol{k}}} \hat{\boldsymbol{f}}_{\boldsymbol{k}} \exp(-2\pi\imath \boldsymbol{k}\boldsymbol{x}_j). \qquad (2.54)$$

This may be reformulated in matrix-vector notation by setting

$$\boldsymbol{f} := (f_j)_{j=0,\ldots,N-1},\ \boldsymbol{A} := (e^{-2\pi\imath \boldsymbol{k}\boldsymbol{x}_j})_{j=0,\ldots,N-1\ ;\ \boldsymbol{k} \in I_M},\ \hat{\boldsymbol{f}} := (\hat{f}_{\boldsymbol{k}})_{\boldsymbol{k} \in I_M}$$

and writing

$$\boldsymbol{f} = \boldsymbol{A}\hat{\boldsymbol{f}}. \qquad (2.55)$$

Related matrix-vector products

A number of related NDFT matrix-vector products can also be defined. To write them down we let $\overline{\boldsymbol{A}}$ be the complex conjugate of the elements of the matrix $\boldsymbol{A}$ and $\boldsymbol{A}^H = \overline{\boldsymbol{A}}^T$ the transposed complex conjugate of the matrix $\boldsymbol{A}$. Using these conventions we can name and summarize the related NDFT matrix-vector products and their component representation as

regular

$$\begin{aligned} \boldsymbol{f} &= \boldsymbol{A}\hat{\boldsymbol{f}} \\ \boldsymbol{f}_j &= \sum_{\boldsymbol{k} \in I_M} \hat{\boldsymbol{f}}_{\boldsymbol{k}} \exp(-2\pi\imath \boldsymbol{k}\boldsymbol{x}_j) \end{aligned} \tag{2.56}$$

adjoint

$$\begin{aligned} \hat{\boldsymbol{f}} &= \boldsymbol{A}^H \boldsymbol{f} \\ \hat{\boldsymbol{f}}_{\boldsymbol{k}} &= \sum_{j=0}^{N-1} \hat{\boldsymbol{f}}_j \exp(2\pi\imath \boldsymbol{k}\boldsymbol{x}_j) \end{aligned} \tag{2.57}$$

conjugated

$$\begin{aligned} \boldsymbol{f} &= \overline{\boldsymbol{A}}\hat{\boldsymbol{f}} \\ \boldsymbol{f}_j &= \sum_{\boldsymbol{k} \in I_M} \hat{\boldsymbol{f}}_{\boldsymbol{k}} \exp(2\pi\imath \boldsymbol{k}\boldsymbol{x}_j) \end{aligned} \tag{2.58}$$

transposed

$$\begin{aligned} \hat{\boldsymbol{f}} &= \boldsymbol{A}^T \boldsymbol{f} \\ \hat{\boldsymbol{f}}_{\boldsymbol{k}} &= \sum_{j=0}^{N-1} \hat{\boldsymbol{f}}_j \exp(-2\pi\imath \boldsymbol{k}\boldsymbol{x}_j) \end{aligned} \tag{2.59}$$

NDFT, FFT and NFFT

From the different NDFT products written in matrix-vector form, as in Equations (2.56)–(2.59), it is clear that it takes $\mathcal{O}(NM_\Pi)$ arithmetic operations to transform between the Fourier-samples and the Fourier-coefficients. This is simply because the matrix $\boldsymbol{A}$ is $N \times M_\Pi$, with $M_\Pi = dim(\mathcal{F}) = \prod_{t=0}^{d-1} M_t$.

However, for the special case of $M_t = M$, $(t = 0, \ldots, d-1)$ and $N = M^d$ equispaced knots $\boldsymbol{x}_{\boldsymbol{k}}$ $(\boldsymbol{k} \in I_M)$, the Fourier-samples $f_{\boldsymbol{k}}$ can be calculated from the Fourier-coefficients $\hat{f}_{\boldsymbol{k}}$ by the fast Fourier transform (FFT) with $\mathcal{O}(N \log N)$ arithmetic operations.

The fast Fourier transform for nonequispaced knots (NFFT) is a generalization of the FFT. The essential idea is that of combining a window function with the standard FFT. The window function is a well localized function in both the space domain and frequency domain. Several different window functions and similar approaches have been proposed. The resulting algorithms are approximative and some of them have been shown to have a computational complexity of $\mathcal{O}(M_\Pi \log M_\Pi + \log(1/\epsilon)N)$, where ϵ is the desired accuracy [243].

2.5.3 Fast Ewald summation

Using optimal parameters in the Ewald summation method implies that the time to calculate the real-space part and the reciprocal-space part are approximately equal. As the number of particles in the system grows we would like to combine the calculation of the short-range part of the potential with the real-space part. This implies that we need to choose a real-space cut-off about the same size as the short-range cut-off. With this nonoptimal choice, the reciprocal-space parts of the Ewald summation method become the most time-consuming to calculate [157].

To show how a fast Ewald summation approach may be obtained from the regular Ewald method, described in §2.5.1, we focus on the reciprocal-space parts. In §2.5.2 we give the details of the discrete Fourier transform (DFT) for data that is nonuniformly spaced (NDFT). Based on these definitions we get a number of useful algorithmic primitives. First we reformulate the reciprocal-space part of the regular Ewald method in terms of the NDFT primitives. Then we show how the fast Fourier transform for nonequispaced (NFFT) can be applied, yielding an Ewald method based on the nonuniform fast Fourier transform.

Reciprocal space terms as DFT

We apply the generalized DFT, described in §2.5.2, to the calculation of the reciprocal-space energy and forces. This allows us to formulate the standard Ewald method for calculating the reciprocal energy and forces in terms of the NDFT primitives.

Reciprocal energy In the case of the electrostatic energy we have from Equation (2.41)

$$U_{qq}^{\text{recip}} = \frac{1}{2\pi L} \sum_{\boldsymbol{n} \neq 0} \frac{e^{-\pi^2 |\boldsymbol{n}|^2 / (\alpha L)^2}}{|\boldsymbol{n}|^2} S(-\boldsymbol{n}) S(\boldsymbol{n}), \tag{2.60}$$

with the structure factor $S(\boldsymbol{n})$ defined as

$$S(\boldsymbol{n}) = \sum_j q_j \exp(-\frac{2\pi\imath}{L} \boldsymbol{n} \cdot \boldsymbol{r}_j). \tag{2.61}$$

By comparing the definition of the transposed NDFT in Equation (2.59) and the structure factor in Equation (2.61) we note that they have the same structure; after a renumbering of the location indexes, the summation limits are also the same. In fact, by setting

- the normalized locations, $\boldsymbol{x}_j = \boldsymbol{r}_j/L$, and
- the samples, $\hat{\boldsymbol{f}}_j = q_j$,

we see by inspection that Equation (2.61) is a 3D instance of Equation (2.59) with $\hat{\boldsymbol{f}}_{\boldsymbol{n}} = S(\boldsymbol{n})$. Furthermore, assuming that the MD simulation box is centered around the origin, the normalized locations can be assumed to be in the domain $\mathcal{D}^d$ as defined in Equation (2.51).

Consequently, we can use the NDFT approach to calculate each of the components of the structure factor. From a computational point of view this means that we can also expect to utilize an NFFT based algorithm to calculate the components of the structure factor $S(\boldsymbol{n})$, rather than the straightforward summation normally used in the Ewald method.

Recasting Equation (2.60) in terms of Fourier-components

$$U_{qq}^{\text{recip}} = \frac{1}{2\pi L} \sum_{\boldsymbol{n}\neq 0} \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} S(-\boldsymbol{n})S(\boldsymbol{n}), \tag{2.62}$$

$$= \frac{1}{2\pi L} \sum_{\boldsymbol{n}\neq 0} \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} |\hat{\boldsymbol{f}}_{\boldsymbol{n}}|^2, \tag{2.63}$$

and using the symmetry of $S(\boldsymbol{n})$ around the origin

$$= \frac{1}{\pi L} \sum_{\boldsymbol{n}_z>0} \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} |\hat{\boldsymbol{f}}_{\boldsymbol{n}}|^2. \tag{2.64}$$

Calculating the energy, U_{qq}^{recip}, using Equation (2.64) means that we

1. calculate all $\hat{\boldsymbol{f}}_{\boldsymbol{n}}$ using the transposed NDFT,
2. scale each $|\hat{\boldsymbol{f}}_{\boldsymbol{n}}|^2$,
3. sum all the scaled components.

Reciprocal forces We calculate the contribution from the reciprocal-space forces using a similar approach as for the energy. In the formula on the next page, $Re(\bullet)$, and $Im(\bullet)$, denote the real and imaginary part of the arguments, respectively. From

Equation (2.34) we have that

$$F_i^{\text{recip}} = \frac{2q_i}{L^2} \sum_{\boldsymbol{n}\neq 0} \boldsymbol{n} \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} \Big[\sin(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)\, Re(S(\boldsymbol{n})) + \cos(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)\, Im(S(\boldsymbol{n})) \Big]. \tag{2.65}$$

Now, the structure factor $S(\boldsymbol{n})$ is just a complex number so the expression in the brackets above can be written as the imaginary part of a product

$$\sin(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)\, Re(S(\boldsymbol{n})) + \cos(\frac{2\pi}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i)\, Im(S(\boldsymbol{n})) = Im\Big[\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i) S(\boldsymbol{n})\Big] \tag{2.66}$$

Inserting this into Equation (2.65) gives

$$F_i^{\text{recip}} = \frac{2q_i}{L^2} \sum_{\boldsymbol{n}\neq 0} \boldsymbol{n} \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} Im\Big[\exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i) S(\boldsymbol{n})\Big] \tag{2.67}$$

$$= Im\Big\{ \frac{2q_i}{L^2} \sum_{\boldsymbol{n}\neq 0} \boldsymbol{n} \frac{e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2}}{|\boldsymbol{n}|^2} S(\boldsymbol{n}) \exp(\frac{2\pi\imath}{L}\boldsymbol{n}\cdot\boldsymbol{r}_i) \Big\} \tag{2.68}$$

Note that Equation (2.68) is a *vector equation*. Furthermore, each of the three components has the same structure as the conjugated NDFT of Equation (2.58). By setting the normalized locations, $\boldsymbol{x}_j = \boldsymbol{r}_j/L$, and the samples,

$$\hat{\boldsymbol{g}}_{\boldsymbol{n}} = \boldsymbol{n} \frac{\exp(-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2)}{|\boldsymbol{n}|^2} S(\boldsymbol{n}) \text{ for } \boldsymbol{n} \neq 0, \tag{2.69}$$

we see, again, by inspection that each component of Equation (2.68) is a 3D instance of Equation (2.58). Assuming that $\boldsymbol{n}$ is in the index set I_M of Equation (2.52), $\boldsymbol{n} \in I_M$, and setting $\hat{\boldsymbol{g}}_0 = 0$, we can formulate F_i^{recip} directly in Fourier-terms

$$F_i^{\text{recip}} = Im\Big\{ \frac{2q_i}{L^2} \sum_{\boldsymbol{n}\in I_M} \hat{\boldsymbol{g}}_{\boldsymbol{n}} \exp(2\pi\imath\boldsymbol{n}\cdot\boldsymbol{x}_i) \Big\} \tag{2.70}$$

$$= \frac{2q_i}{L^2} Im(\boldsymbol{g}_i). \tag{2.71}$$

Calculating the reciprocal-space force F_i^{recip} on particle i, using Equation (2.70), means that we

1. start with the structure factor components, $S(\boldsymbol{n})$, already obtained when we calulated U_{qq}^{recip},
2. scale each $S(\boldsymbol{n})$ using Equation (2.69),

3. giving a new set of Fourier-coefficients that are transformed back to real-space, via Equation (2.70), using the conjugated NDFT, and finally

4. with Equation (2.71), taking the imaginary part of coefficient $\boldsymbol{g}_i$ and scaling it with $\frac{2q_i}{L^2}$ gives the reciprocal force on particle i.

Thus we can use the NDFT approach to calculate each of the components of the reciprocal forces. Again, from a computational point of view this means we can expect to utilize an NFFT based algorithm to find the respective components.

Combining the Ewald method and NFFT

The reformulation of U_{qq}^{recip} and F_i^{recip} as in Equations (2.64) and (2.70) shows the central rôle of the transposed and conjugated NDFT in calculating the reciprocal-space energy and forces. Starting with the location of the charged particles, the structure factor $S(\boldsymbol{n})$ is calculated via a transposed NDFT. In the language of Ewald summation, we transform from real-space to reciprocal-space. Scaling the absolute value of the Fourier-components and summing gives U_{qq}^{recip}. To find F_i^{recip} we go back from the reciprocal-space to the real-space by first calculating the Fourier-components of the forces and then performing a conjugated NDFT.

An implementation of Ewald summation uses cut-offs, in reciprocal-space, n_{cut}, and real-space, r_{cut}; with α large enough and with a required accuracy, δ, truncate the sums Equation (2.38) and Equation (2.41) at the respective cut-offs so that the last term added $\leq \delta$, in each of the sums. When n_{cut} is fixed by

$$e^{-\pi^2|\boldsymbol{n}|^2/(\alpha L)^2} \leq \delta : n_{cut} \geq \frac{\alpha L}{\pi}\sqrt{-\log(\delta)} : n_{cut} \propto L \propto N^{1/3}. \tag{2.72}$$

Then r_{cut} is determined by

$$\operatorname{erfc}(\alpha r_{cut}) \approx e^{-\alpha^2 r_{cut}^2} \leq \delta : r_{cut} = \frac{\pi n_{cut}}{\alpha^2 L}. \tag{2.73}$$

We have recast U_{qq}^{recip} and F_i^{recip} in terms of Fourier-components and set $M_t = 2\sigma_s n_{cut}$, where σ_s is the oversampling factor. This gives $M_\Pi = \prod_{t=0}^{d-1} M_t$. In general the computational complexity of the NFFT method is $\mathcal{O}(M_\Pi \log M_\Pi + \log(1/\epsilon)N)$, where ϵ is the desired accuracy in the approximation used within NFFT [243]. Using Equation (2.72) and the above defintion of M_Π, we see that the complexity becomes $\mathcal{O}(N \log N + \log(1/\epsilon)N)$. Note that ϵ is a function of m, for a fixed oversampling factor. With a controlled approximation of the structure factor via the use of nonuniform fast Fourier transform, the original computational complexity of $\mathcal{O}(N^{3/2})$ becomes $\mathcal{O}(N \log N)$.

At this stage, the path to a fast Ewald method should now be clear. By specifying an accuracy δ, we replace the transposed and conjugated NDFT with the corresponding operations using the NFFT algorithm. Thus Equations (2.64) and (2.70) become a concise procedure to calculate approximations of $U_{qq}^{\text{recip}}$and F_i^{recip}.

Most of the mathematical details can be kept separate and hidden in a set of library routines and the remaining formulas pertain to the physics of the problem. Furthermore, with a library implementation based on a state-of-the-art FFT-library, we have good reason to expect it to be efficient.

Implementation and results

In our current implementation [245], we use the libraries FFTW [246] and NFFT [244]. Details of the accuracy and scaling properties can be found in the paper.

Basing the implementation on libraries has a number of advantages. It makes the implementation task easier and introduces a convenient division of labor in the program code: the mathematical aspects are mainly concentrated to the libraries while the physical aspects of the problem remain. Also, since the code becomes quite compact without becoming convoluted, it becomes easier to check, understand, and explain. Improvements and optimizations of the libraries can be easily included in the program, usually by just relinking the program. For example, the customization of the window function used in the NFFT algorithm—Gaussian functions, dilated cardinal B-splines, Sinc functions, or Kaiser-Bessel functions—is currently achieved by recompiling the NFFT library and relinking the application. Due to the comparatively small size of the NFFT library this is very quick. Furthermore, improvements in either theory or implementation of the used libraries will be easily accessible.

In summary we claim that the ENUF method is

- efficient and concise, and
- has a clear separation of concerns between mathematical and physical details.

In a sense it can be said that we get the best of two worlds: a concise and efficient algorithm. The separation of mathematical concerns is a bonus that has the potential to simplify implementation and further developments due to the fact that they may occur independently of each-other.

> The first general-purpose electronic digital computer, the ENIAC, was a highly parallel and highly decentralised machine. [...] The difficulty of programming parallel computers is a recurring theme that is still with us today, and it remains to be seen whether the second coming of the parallel computer in the 1980s will be more successful than the first!
>
> R. W. Hockney, C. R. Jesshope
> *Parallel Computers2*

Chapter 3

Outlook

MOLECULAR Dynamics as a discipline was enabled by the electronic computer and MD continues to evolve with the latest developments in computer technology. The computational horizon is steadily expanding and as the investigated problems become more complex, more efficient and general methods are needed.

3.1 Further Development of Results

THIS thesis has investigated methods for performing large-scale parallel Molecular Dynamics. The focus has been on methods that advance along the F and N axis (See Figure on page 32).

I have shown that it is possible to perform *ab initio* MD simulations that give results that are in good agreement with experimental results. The method I have demonstrated is computationally expensive and parallel computers are an essential component. However, given the modest communication demands of the method, it can be expected to run well even on parallel platforms that have an ordinary communication network, *i.e.*, clusters and Grid platforms. This implies that adequate computational resources are not too hard to find. Currently I am preparing a simulation study of electron transport in liquid water.

For almost a decade, the most common parallel platforms have been built using commodity processors, but with a large variation in the type of interconnect. The number of nodes have usually been rather modest. This is now changing and the massively parallel computers(MPP) are back again.[1] The algorithms for short-range and long-range interactions presented in this thesis were tailored for MPP platforms with thousands of processors and aimed at large-scale simulations. The specific platforms I used are long gone, but the algorithmic ideas remain and serve

[1] Some examples from http://top500.org dated November 2005: BlueGene/L from IBM with 131072 processors, SGI Altix with 10160 processors, Cray XT3 with 10880 processors.

as a remainder that algorithms for MPPs need to be designed from scratch to deliver expected performance.

One of the most challenging aspects of MD is the treatment of long-range interactions. I have proposed a method(ENUF) that is efficient and straightforward to implement in existing MD programs. The results so far show that ENUF preserves both energy and momentum and has a behaviour very similar to the Ewald summation method. Work is in progress to improve the efficiency of ENUF and also include a modest degree of parallelism.

3.2 Speculations and Hints for Walkabouts

THE epigraph of this Chapter is an indication that the problem of "programming parallel computers" is not new. The problem is not likely to disappear and can be regarded as one important characteristic of the whole field of Computationl Science. An approach that aims to control complexity at different levels by a divide-and-conquer strategy should be consciously adopted.

It is important to make a distinction between the particular implementation platform and the algorithm used. Hardware platforms age and disappear, but the algorithms remain. I have noted before that "The fastest and most cost-effective computers available today are the massively parallel computing systems based on 10^2–10^5 processors working in parallel". With the advent of the BLUE-GENE/L platform this is certainly true once again. At least for the performance aspect. Another recent development is the Grid [37]. To master the additional software complexity these platforms bring, advances in Computer Science can help. In particular I believe that software development approaches that combines a multi-paradigm approach [95] with "test-driven development" and Agile practices will be very powerful [97, 96].

From an MD point-of-view, the Grid can potentially offer an "ensemble" of computers that can be used in a simulation. It is no longer the case that we need to be limited by a few simulation runs in our investigations, but we can expect to routinely employ a large number of runs. Parameter spaces can be more fully sampled and different hypotheses can be compared against experimental data. To leverage this new capability more robust and general approaches to statistical analysis should adopted. I strongly advocate the assimilation of a Bayesian approach to data analysis [247, 248, 249, 250, 251]. Results along these lines are already starting to appear [252].

Going even further out on a limb, I have a strong hunch that an information theoretical approach to Statistical Mechanics can provide new insights that are relevant for the practice of Molecular Dynamics. For the inquiring mind I refer to some of the text books on the subject [253, 254, 255, 256, 257, 258].

3.3 Conclusion

To take full advantage of the increasing computational capabilities, influences and cross fertilization from other fields, such as Statistics, Computer Science, Numerical Analysis and Mathematics, are crucial. This work is an attempt along these lines. It also shows that it is very probable that there are many interesting opportunities that remain to be discovered and explored.

AVANCEZ!

Bibliography

[1] R. Hockney and C. Eastwood; *Computer Simulation Using Particles* (McGraw–Hill, New York, **1981**). 2, 25, 52

[2] V. E. Thoren; *The Lord of Uranienborg* (Cambridge University Press, **1990**). 2

[3] A. Koestler; *The Sleepwalkers* (Arkana Books, London, **1989**). 3

[4] C. Baumgardt; *Johannes Kepler: Life and Letters* (Gollancz, London, **1952**). 3

[5] S. Drake; *Gallileo: Pioneer Scientist* (University of Toronto Press, **1990**). 3

[6] M. Sharratt; *Gallileo: Decisive Innovator* (Cambridge University Press, **1994**). 3

[7] P. S. Marquis de Laplace; *A Philosophical Essay on Probabilities (1814)* (Dover Publications, **1951**). Translated from the 6th French edition. 3

[8] G. Dahlquist and A. Björck; *Numerical Methods*; Series in Automatic Computation (Prentice-Hall, **1974**). Translated by Ned Anderson. 3

[9] D. Adams; *The Hitchhiker's Guide to the Galaxy* (Harmony Books, New York, **1979**). 3

[10] B. J. Alder and T. E. Wainwright; *Phase transition for a hard sphere system*; J. Chem. Phys. **1957**:**27**:1208–1209. 4, 33

[11] W. W. Wood and J. D. Jacobson; *Preliminary results from a recalculation of the Monte Carlo equation of state of hard spheres*; J. Chem. Phys. **1957**:**27**:1207–1208. 4

[12] J. M. Haile and G. A. Mansoori (editors); *Molecular-Based Study of Fluids*; Advances in Chemistry Series; 204 (American Chemical Society, **1983**). 4, 76

[13] W. G. Hoover, A. J. C. Ladd, and V. N. Hoover; *Historical Development and Recent Applications of Molecular Dynamics Simulation*; in: Haile and Mansoori [12]. 4

[14] W. W. Wood; *Early history of computer simulations in statistical mechanics*; in: Ciccotti and Hoover [259] 3–14. 4

[15] A. Rahman; *Correlations in the motion of atoms in liquid argon*; Phys. Rev. **1964**:**136(2A)**:405–411. Also published in [179]. 4, 33

[16] L. Verlet; *Computer "Experiments" On Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules*; Phys. Rev. **1967**:**159(1)**:98–103. 4, 26, 36

[17] A. Rahman and F. H. Stillinger; *Molecular dynamics study of liquid water*; J. Chem. Phys. **1971**:**55(7)**:3336–3359. 4

[18] L. Woodcock and K. Singer; *Thermodynamic and structural properties of liquid ionic salts obtained by Monte Carlo computation. Part 1.—Potassium Chloride*; Trans. Faraday Soc. **1971**:**67(1)**:12–30. 4

[19] P. Schofield; *Computer Simulation Studies of the Liquid State*; Comput. Phys. Commun. **1973**:**5**:17–23. 4, 26, 44

[20] R. W. Hockney, S. P. Goel, and C. R. Eastwood; *A 10000 particle molecular dynamics model with long range forces*; Chem. Phys. Lett. **1973**:**21(3)**:589–591. 4, 26, 44

[21] B. Quentrec and C. Brot; *New Method for Searching for Neighbors in Molecular Dynamics Computations*; J. Comput. Phys. **1973**:**13**:430–432. 4, 26

[22] D. Ceperley and J. Tully (editors); *Stochastic Molecular Dynamcis*; *Proc. Nat. Resource Comp. Chem.*, volume 6 (**1979**). 4

[23] N. Grønbech-Jensen and S. Doniach; *Long-Time Overdamped Langevin Dynamics of Molecular Chains*; J. Comput. Chem. **1994**:**15(9)**:997–1012. 4

[24] P. J. Hoogerbrugge and J. M. V. A. Koelman; *Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics*; Europhys. Lett. **1992**:**19(3)**:155–160. 4

[25] R. Groot and P. Warren; *Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation*; J. Chem. Phys. **1997**:**107(11)**:4423–4435. 4

[26] A. Lemak and N. Balabaev; *Molecular dynamics simulation of a polymer chain in solution by collisional dynamics method*; J. Comput. Chem. **1996**:**17(15)**:1685–1695. 4

[27] J. Turner, P. Weiner, *et al.*; *Reduced Variable Molecular Dynamics*; J. Comput. Chem. **1995**:**16(10)**:1271–1290. 4

[28] D. C. Rapaport; *The Art of Molecular Dynamics Simulation*; 2nd edition (Cambridge University Press, **2004**); ISBN 0-521-82568-7. 5

[29] D. Frenkel and B. Smit; *Understanding Molecular Simulation: from Algorithms to Applications*; *Computational Science Series*, volume 1; 2nd edition (Academic Press, **2002**); ISBN 0-122-67351-4. 5, 52, 53

[30] T. Schlick; *Molecular Modeling: An Interdisciplinary Guide* (Springer-Verlag, New York, **2002**); ISBN 0-387-95404-X. 5

[31] A. R. Leach; *Molecular Modelling: Principles and Applications*; 2nd edition (Prentice-Hall, **2001**); ISBN 0-582-38210-6. 5

[32] J. M. Haile; *Molecular Dynamics Simulation: Elementary Methods* (John Wiley & Sons, New York, London, Sydney, **1992**). 5, 52

[33] M. Allen and D. Tildesley; *Computer Simulation of Liquids* (Clarendon Press, Oxford, **1987**). 5, 26, 52

[34] B. Leimkuler and S. Reich; *Simulating Hamiltonian Dynamics* (Cambridge University Press, **2004**); ISBN 0-521-77290-7. 5

[35] T. C. Beutler and W. F. van Gunsteren; *Molecular dynamics free energy calculation in four dimensions*; J. Chem. Phys. **1994**: **101(2)**:1417–1422. 6

[36] B. Plale, G. Eisenhauer, *et al.*; *From interactive applications to distributed laboratories*; IEEE Concurr. **1998**:**6(2)**:78–90. 6

[37] P. V. Coveney; *Scientific Grid computing*; Phil. Trans. R. Soc. A-Math. Phys. Eng. Sci **2005**:**363(1833)**:1707–1713; DOI. 6, 72

[38] J. Phillips, R. Braun, *et al.*; *Scalable molecular dynamics with NAMD*; J. Comput. Chem. **2005**:**26(16)**:1781–1802; DOI. 6

[39] F. Gagliardi, B. Jones, *et al.*; *Building an infrastructure for scientific Grid computing: status and goals of the EGEE project*; Phil. Trans. R. Soc. A-Math. Phys. Eng. Sci **2005**:**363(1833)**:1729–1742; DOI. 6

[40] M. Born and T. von Karman; *Über Schwingungen in Raumgittern*; Physikalische Zeitschrift **1912**:**13**:297–309. 7

[41] D. Fincham and D. M. Heyes; *Recent Advances in Molecular Dynamics Computer Simulation*; in: M. W. Evans (editor) *Dynamical Processes in Condensed Matter*, *Advances in Chemical Physics*, volume LXIII; 493–575 (John Wiley & Sons, New York, London, Sydney, **1985**). 7, 8

[42] B. L. Holian and D. J. Evans; *Shear Viscosities away from the Melting Line*; J. Chem. Phys. **1983**:**78(8)**:5147–5150. 8

[43] D. J. Adams; *Alternatives to the periodic cube in computer simulation*; CCP5 Information Quarterly **1983**:**10**:30–36. 8

[44] W. Dzwinel, J. Kitowski, and J. Mościński; *"Checker Board" Periodic Boundary Conditions in Molecular Dynamics Codes*; Mol. Simul. **1991**: **7**:171–179. 8

[45] H. Bekker; *Unification of box shapes in molecular simulations*; J. Comput. Chem. **1997**:**18(15)**:1930–1942. 8

[46] ———; *Molecular dynamics simulation methods revised*; Ph.D. thesis; Rijksuniversiteit Groningen **1996**. http://www.cs.rug.nl/~bekker/. 8

[47] M. S. C. Reed and K. M. Flurchick; *Investigation of artifacts due to periodic boundary conditions*; Comput. Phys. Commun. **1996**:**95(1)**:39–46. 8

[48] V. Teboul and S. Chaussedent; *Cutoff effect in molecular dynamics simulations of interaction induced light scattering spectra*; Comput. Phys. Commun. **1997**:**105(2-3)**:151–158. 8

[49] G. C. Maitland, M. Rigby, *et al.*; *Intermolecular Forces: their Origin and Determination*; *International Series of Monographs on Chemistry*, volume 3 (Clarendon Press, Oxford, **1981**). 9

[50] J. P. Hansen and I. R. McDonald; *Theory of Simple Liquids*; 2nd edition (Academic Press, **1986**). 9

[51] A. J. Stone; *The Theory of Intermolecular Forces*; *International Series of Monographs on Chemistry*, volume 32 (Clarendon Press, Oxford, **1996**). 9, 22

[52] P. J. Steinbach and B. R. Brooks; *New Spherical-Cutoff Methods for Long-Range Forces in Macromolecular Simulation*; J. Comput. Chem. **1994**: **15(7)**:667–683. 10, 25

[53] H. Schreiber and O. Steinhauser; *Taming Cutoff Induced Artifacts in Molecular-Dynamics Studies of Solvated Polypeptides - the Reaction Field Method*; Journal of Molecular Biology **1992**:**228(3)**:909–923. 10

[54] J. Roberts and J. Schnitker; *Boundary-Conditions in Simulations of Aqueous Ionic-Solutions—a Systematic Study*; J. Phys. Chem. **1995**: **99(4)**:1322–1331. 10

[55] C. Anézo, A. H. de Vries, *et al.*; *Methodological Issues in Lipid Bilayer Simulations*; J. Phys. Chem. B **2003**:**107(35)**:9424–9433; DOI. 10, 52

[56] M. Patra, M. Karttunen, *et al.*; *Molecular Dynamics Simulations of Lipid Bilayers: Major Artifacts Due to Truncating Electrostatic Interactions*; Biophys. J. **2003**:**84(6)**:3636–3645. 10, 52

[57] P. P. Ewald; *Die Berechnung optischer und elektrostatischer Gitterpotentiale*; Ann. Phys. (Leipzig) **1921**:**64(3)**:253–287. 10, 26, 52, 53

[58] S. W. de Leeuw, J. W. Perram, and E. R. Smith; *Simulation of electrostatic systems in periodic boundary conditions I. Lattice sums and dielectric constants*; Proc. R. Soc. London **1980**:**A 373**:27–56. 10, 25, 26, 48, 52, 53

[59] B. A. Luty, I. G. Tironi, and W. F. van Gunsteren; *Lattice-sum methods for calculating electrostatic interactions in molecular simulations*; J. Chem. Phys. **1995**:**103(8)**:3014–3021; DOI. 10

[60] P. F. Batcho and T. Schlick; *New splitting formulations for lattice summations*; J. Chem. Phys. **2001**:**115(18)**:8312–8326; DOI. 10

[61] D. R. Wheeler and J. Newman; *A less expensive Ewald lattice sum*; Chem. Phys. Lett. **2002**:**366(5–6)**:537– 543; DOI. 10

[62] I. G. Tironi, R. Sperb, *et al.*; *A generalized reaction field method for molecular dynamics simulations*; J. Chem. Phys. **1995**:**102(13)**:5451–5459; DOI. 10

[63] P. J. Steinbach and B. R. Brooks; *New spherical-cutoff methods for long-range forces in macromolecular simulation*; J. Comput. Chem. **1994**: **15(7)**:667–683; DOI. 10, 52

[64] C. L. Brooks III, B. M. Pettitt, and M. Karplus; *Structural and energetic effects of truncating long ranged interactions in ionic and polar fluids*; J. Phys. Chem. **1985**:**83(11)**:5897–5908; DOI. 10, 52

[65] K. S. Kim; *On effective methods to treat solvent effects in macromolecular mechanics and simulations*; Chem. Phys. Lett. **1989**:**156(2–3)**:261–268; DOI. 10, 52

[66] X. Wu and B. R. Brooks; *Isotropic periodic sum: A method for the calculation of long-range interactions*; J. Chem. Phys. **2005**:**122**:044107. 10, 25, 52

[67] S. Pfalzner and P. Gibbon; *Many-Body Tree Methods in Physics* (Cambridge University Press, **1996**). 10, 25

[68] A. Brandt and A. A. Lubrecht; *Multilevel matrix multiplication and fast solution of integral equations*; J. Comput. Phys. **1990**:**90**:348–370. 10, 25

[69] L. Y. Zaslavsky and T. Schlick; *An adaptive multigrid technique for evaluating long-range forces in biomolecular simulations*; Appl. Math. Comput. **1998**:**97(2–3)**:237–250; DOI. 10, 25, 26

[70] C. Sagui and T. Darden; *Multigrid methods for classical molecular dynamics simulations of biomolecules*; J. Chem. Phys. **2001**: **114(15)**:6578–6591; DOI. 10, 25, 26, 52

[71] R. D. Skeel, I. Tezcan, and D. J. Hardy; *Multiple grid methods for classical molecular dynamics*; J. Comput. Chem. **2002**:**23(6)**:673–684; DOI. 10, 25, 26

[72] J. A. Izaguirre, S. S. Hampton, and T. Matthey; *Parallel multigrid summation for the N-body problem*; J. Para. Dist. Comput. **2005**: **65(8)**:949–962. 10, 25, 26, 52

[73] L. Greengard and V. Rokhlin; *A Fast Algorithm for Particle Simulations*; J. Comput. Phys. **1987**:**73(2)**:325–348; DOI. 10, 52

[74] P. Koehl; *Electrostatics calculations: latest methodological advances*; Current Opinion in Structural Biology **2006**:**16**:1–10; DOI. 10, 25

[75] C. Kittel; *Introduction to Solid State Physics*; 6th edition (John Wiley & Sons, **1986**). 10

[76] I.-C. Yeh and G. Hummer; *System-Size Dependence of Diffusion Coefficients and Viscosities from Molecular Dynamics Simulations with Periodic Boundary Conditions*; J. Phys. Chem. B **2004**: **108(40)**:15873–15879; DOI. 10

[77] D. Fincham; *Parallel Computers and Molecular Simulation*; Mol. Simul. **1987**:**1**:1–45. 11, 26, 31, 41

[78] D. Rapaport; *Large-scale molecular dynamics simulation using vector and parallel computers*; Computer Physics Reports **1988**:**9(1)**:1–53. 11

[79] S. Gupta; *Computing aspects of molecular dynamics simulation*; Comput. Phys. Commun. **1992**:**70**:243–270. 11

[80] S. Plimpton; *Fast parallel algorithms for short-range molecular-dynamics*; J. Comput. Phys. **1995**:**117(1)**:1–19. 11, 27, 30, 48

[81] F. Hedman and A. Laaksonen; *Large scale parallel molecular dynamics simulations*; chapter 7, 231–280 (Elsevier Science Publishers, **1999**). 11

[82] G. S. Heffelfinger; *Parallel atomistic simulations*; Comput. Phys. Commun. **2000**:**128(1–2)**:219–237. 11

[83] K. Kholmurodov, M. Altaisky, *et al.*; *Methods of molecular dynamics for simulation of physical and biological processes*; Phys. Part. Nuc. **2003**:**34(2)**:244–262. 11

[84] G. Csanyi, T. Albaret, *et al.*; *Multiscale hybrid simulation methods for material systems*; J. Phys.-Cond. Mat. **2005**:**17(27)**:R691–R703. 11

[85] D. A. Patterson and J. L. Hennessy; *Computer Architecture A Quantitative Approach*; 2nd edition (Morgan Kaufmann Publishers Inc, **1996**). 11, 19

[86] R. Hockney and J. Jesshope; *Parallel Computers 2*; 2nd edition (Adam Hilger, **1988**). 11, 31

[87] D. Auerbach, W. Paul, *et al.*; *A Special Purpose Parallel Computer for Molecular Dynamics: Motivation, Design, Implementation and Application*; J. Phys. Chem. **1987**:**91(19)**:4881–4890. 11, 44

[88] Y. Komeiji, M. U. H. Yokoyama, *et al.*; *A high performance system for molecular dynamics simulation of biomolecules using special purpose computer*; in: *Proceedings of Biocomputing '96* (**1996**) . 11

[89] G. LaPenna, V. Minicozzi, *et al.*; *Molecular Dynamics with the massively parallel APE computers*; Comput. Phys. Commun. **1997**:**106(1-2)**:53–68. 11

[90] F. Allen, G. Almasi, *et al.*; *Blue Gene: A vision for protein science using a petaflop supercomputer*; IBM Syst. J. **2001**:**40(2)**:310–327. 12

[91] G. F. Pfister; *In Search of Clusters: the coming battle in lowly parallel computing* (Prentice-Hall, **1995**). 12, 14

[92] LINUX Journal **1998**. 12

[93] T. L. Sterling, J. Salmon, *et al.*; *How to build a Beowulf: a guide to the implementation and application of PC clusters* (MIT Press, **1999**); ISBN 0-262-69218-X. Second printing. 12

[94] M. V. Wilkes; *The Lure of Parallelism and Its Problems*; in: *Computing Perspectives* [102] 75–84. 12

[95] J. O. Coplien; *Multi-paradigm design for C++* (Addison-Wesley, **1998**); ISBN 0-201-82467-1. 12, 16, 72

[96] R. C. Martin; *Agile Software Development: Principles, Patterns, and Practices* (Pearson Education, **2003**); ISBN 0-13-597444-5. 12, 72

[97] K. Beck; *Test-driven development: by example* (Pearson Education, **2003**); ISBN 0-321-14653-0. 12, 72

[98] K. Hwang; *Advanced Computer Architecture: Parallelism, Scalability, Programmability* (McGraw-Hill, **1993**). 12, 15

[99] G. C. Fox, R. D. Williams, and P. C. Messina (editors) *Parallel Computing Works!* (Morgan Kaufmann, **1994**). http://www.npac.syr.edu/copywrite/pcw/. 12, 31

[100] I. Foster; *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering* (Addison-Wesley, **1994**). See also URL http://www-unix.mcs.anl.gov/dbpp/. 12, 15, 17, 18, 27

[101] P. Hammarlund; *Techniques for Efficient Parallel Scientific Computing*; Ph.D. thesis; Royal Institute of Technology; Stockholm, Sweden **1996**. 12, 16

[102] M. V. Wilkes; *Computing Perspectives* (Morgan Kaufmann Publishers Inc, **1995**). 13, 81

[103] M. J. Flynn; *Very high-speed computing systems*; Proceedings of the IEEE **1966**:**54(12)**:1901–1909. 14

[104] ———; *Some Computer Organizations and Their Effectiveness*; IEEE Trans. Comp. **1972**:**C–21(9)**:948–960. 14

[105] G. V. Wilson; *Practical Parallel Programming* (MIT Press, **1995**). 15

[106] A. Geist, A. Beguelin, *et al.*; *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing*; Scientific and engineering computation (MIT Press, Cambridge, MA, **1994**); ISBN 0-262-57108-0 (paperback). 16

[107] W. Gropp, E. Lusk, and A. Skjellum; *Using MPI: portable parallel programming with the message-passing interface*; 2nd edition (MIT Press, **1995**); ISBN 0-262-57134-X. 16

[108] W. Gropp, E. Lusk, and R. Thakur; *Using MPI-2: advanced features of the message-passing interface* (MIT Press, **1999**); ISBN 0-262-57133-1. 16

[109] M. Snir, S. W. Otto, *et al.*; *MPI—the complete reference. The MPI core*; volume 1 (MIT Press, **1998**); ISBN 0-262-69215-5. Third printing, 2000. 16

[110] W. Gropp, S. Huss-Lederman, *et al.*; *MPI—the complete reference. The MPI extensions*; volume 2 (MIT Press, **1998**); ISBN 0-262-69215-5. Third printing, 2000. 16

[111] W. Gropp and E. Lusk; *Goals Guiding Design: PVM and MPI*; in: *IEEE Cluster* (**2002**) To appear. 16

[112] W. D. Gropp; *Learing from the Success of MPI*; in: *9th EuroPVM/MPI* (**2002**) . 16

[113] J. C. Adams, W. S. Brainerd, *et al.*; *Fortran 95 handbook: complete ISO/ANSI reference* (MIT Press, **1997**). 16, 44

[114] B. W. Kernighan and D. M. Ritchie; *The C Programming Language*; 2nd edition (Prentice-Hall, **1988**). 16

[115] B. Stroustrup; *The C++ programming language*; Special edition (Addison-Wesley, **2000**); ISBN 0-201-70073-5. http://www.research.att.com/~bs/. 16

[116] ———; *The design and evolution of C++* (Addison-Wesley, **1994**); ISBN 0-201-54330-3. The so called D&E book. 16

[117] *Information Technology—Programming languages—C*; ISO/IEC 9899:1999. Second edition 1999-12-01. Available from www.iso.ch. 16

[118] *Information Technology—Programming languages—C++*; ISO/IEC 14882:1998. First edition 1998-09-01. Available from www.iso.ch. 16

[119] B. Stroustrup; *A brief look at C++0x*; Artima **2006**: http://www.artima.com/cppsource. 16

[120] ———; *The design of C++0x*; C/C++ User Journal **2005**: http://www.research.att.com/~bs. 16

[121] G. Booch; *Object-oriented analysis and design with applications*; 2nd edition (Addison-Wesley, **1994**). 16

[122] V. K. Decyk, C. D. Norton, and B. K. Szymanski; *How to support inheritance and run-time polymorphism in Fortran 90*; Comput. Phys. Commun. **1998**:**115(1)**:9–17. 16

[123] T. L. Veldhuizen and M. E. Jernigan; *Will C++ be faster than Fortran?*; in: *Proceedings of the 1st International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'97)*; Lecture Notes in Computer Science (Springer-Verlag, **1997**) . 16

[124] T. L. Veldhuizen; *Techniques for Scientific C++*; Technical Report 542; Indiana University Computer Science **2000**. Version 0.4. 16

[125] T. Matthey, T. Cickovski, *et al.*; *ProtoMol, an object-oriented framework for prototyping novel algorithms for molecular dynamics*; ACM T Math Software **2004**:**30(3)**:237–265; DOI. 16, 28

[126] E. F. van de Velde; *Concurrent Scientific Computing*; *Texts in Applied Mathematics*, volume 16 (Springer-Verlag, **1994**). 17

[127] M. Schütz and R. Lindh; *An integral direct, distributed-data, parallel MP2 algorithm*; Theoretica Chimica Acta **1997**:**95(1–2)**:13–34. 17

[128] G. M. Amdahl; *Validity of the single-processor approach to achieving large scale computing capabilities*; in: *AFIPS Conference Proceedings*, volume 30 (**1967**) 483–485. 17

[129] J. L. Gustafson; *Reevaluating Amdahls's Law*; Communications of the ACM **1988**:**31(5)**:532–533. 17

[130] A. V. Aho, J. E. Hopcroft, and J. D. Ullman; *Data Structures and Algorithms* (Addison-Wesley, **1985**); ISBN 0-201-00023-7. 18

[131] W. Gropp and E. Lusk; *Reproducible Measurements of MPI Performance Characteristics*; in: *PVMMPI'99* (**1999**) . 18

[132] F. A. Smith; *Modeling, Predicting, and Optimizing Parallel Performance of Grid Structured Problems*; Ph.D. thesis; North Carolina State University **2004**. 18

[133] M. Martonosi, A. Gupta, and T. E. Anderson; *Tuning Memory Performance of Sequential and Parallel Programs*; Computer **1995**: **28(4)**:32–40. 19

[134] B. B. Fraguela, R. Doallo, and E. L. Zapata; *Probabilistic miss equations: Evaluating memory hierarchy performance*; IEEE Trans. Comp. **2003**:**52(3)**:321–336. 19

[135] E. Berg and E. Hagersten; *SIP: Performance tuning through source code interdependence*; in: *EURO-PAR 2002 Parallel Processing, Lecture Notes in Computer Science*, volume 2400 (**2002**) 177–186. 19

[136] P. Lomdahl, P. Tamayo, *et al.*; *50 GFlops Molecular Dynamics on the CM-5*; in: *Supercomputing 93* (**1993**) 520–527. 21

[137] A. P. Rendell, A. Bliznyuk, *et al.*; *Computationalchemistry on Fujitsu vector-parallel processors: Development and performance of applications software*; Parallel Comput. **2000**:**26(7–8)**:887–911; DOI. 21

[138] U. Andersson; *Time-Domain Methods for the Maxwell Equations*; Ph.D. thesis; KTH; NADA **2001**. 21

[139] J. Hein, F. Reid, *et al.*; *On the performance of molecular dynamics applications on current high-end systems*; Phil. Trans. R. Soc. A-Math. Phys. Eng. Sci **2005**:**363(1833)**:1987–1991; DOI. 21

[140] J. Comput. Chem. **2005**:**26(16)**:1667–1803; DOI. Thematic issue with articles describing Amber, BOSS, MCPRO, GROMACS, GROMOS, IMPACT, and NAMD. 21

[141] P. S. Lomdahl and D. M. Beazly; *Multi-Million Particle Molecular Dynamics in MPPs*; in: J. Dongarra, K. Madsen, and J. Waśniewski (editors) *Applied Parallel Computing, Computations in Physics, Chemistry and Engineering Science*, *Lecture Notes in Computer Science*, volume 1041 (Springer-Verlag, **1996**) 391–407. 21

[142] U. Essmann, L. Perera, *et al.*; *A Smooth Particle Mesh Ewald Method*; J. Comput. Phys. **1995**:**103(19)**:8577–8593. 22, 25

[143] A. Y. Toukmaji and J. A. Board Jr.; *Ewald summation techniques in perspective: a survey*; Comput. Phys. Commun. **1996**:**95**:73–92. 22, 26

[144] T. A. Darden, A. Toukmaji, and L. G. Pedersen; *Long-range electrostatic effects in biomolecular simulations*; Journal de Chimie Physique et de Physico-Chimie Biologique **1997**:**94(7–8)**:1346–1364. 22

[145] F. Hedman and A. Laaksonen; *Data Parallel Large-Scale Molecular Dynamics for Liquids*; Int. J. of Quantum Chem. **1993**:**46(1)**:27–38; DOI. 25

[146] J. H. Dunn and S. G. Lambrakos; *Calculating Complex Interactions in Molecular Dynamics Simulations Employing Lagrangian Particle Tracking Schemes*; J. Comput. Phys. **1994**:**111(1)**:15–23. 25, 26

[147] R. Everaers and K. Kremer; *A fast grid search algorithm for molecular dynamics simulation with short-range interactions*; Comput. Phys. Commun. **1994**:**81(1–2)**:19–55. 25, 26

[148] J. Lekner; *Summation of Coulomb Fields in Computer-Simulated Disordered Systems*; Physica A **1991**:**176**:485–498. 25

[149] ———; *Coulomb Forces and Potentials in Systems with an Orthorhombic Unit Cell*; Mol. Simul. **1998**:**20(6)**:357–368. 25

[150] R. Strebel and R. Sperb; *An alternative to Ewald sums. Part 3: Implementation and results*; Mol. Simul. **2001**:**27(1)**:61–74. 25

[151] J. Barnes and P. Hut; *A hierarchical $O(N \log N)$ force-calculation algorithm*; Nature **1986**:**324**:446–449. 25

[152] L. Greengard and V. Rokhlin; *The Rapid Evaluation of Potential Fields in Three Dimensions*; in: C. Andersson and C. Greengard (editors) *Vortex Methods, Lecture Notes in Mathematics 1360*; 121–141 (Springer Verlag, New York, **1988**). 25

[153] C. R. Anderson; *An Implementation of the Fast Multipole Method Without Multipoles*; Technical Report CAM 90–14; Department of Mathematics, UCLA; Los Angeles, CA. 90024-1555 **1990**. 25

[154] H.-Q. Ding, N. Karasawa, and W. A. Goddard III; *Atomic level simulations on a million particles: The cell multipole method for Coulomb and London nonbond interactions*; J. Chem. Phys. **1992**:**97(6)**:4309–4315. 25

[155] I. G. Tironi, R. Sperb, *et al.*; *A generalized reaction field method for molecular dynamics simulations*; J. Chem. Phys. **1995**:**102(13)**:5451–5459. 26, 52

[156] 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*; Mol. Phys. **1988**:**65**:875–893. 26, 51, 53, 62

[157] F. Hedman and A. Laaksonen; *A Data-parallel Molecular Dynamics Method for Liquids with Coulombic Interactions*; Mol. Simul. **1995**: **14(4–5)**:235–244. 26, 65

[158] B. A. Luty, M. E. Davis, *et al.*; *A Comparison of Particle-Particle Particle-Mesh and Ewald Methods for Calculating Electrostatic Interactions in Periodic Molecular Systems*; Mol. Simul. **1994**:**14(1)**:11–20. 26

[159] K. Esselink; *A comparison of algorithms for long-range interactions*; Comput. Phys. Commun. **1995**:**87(3)**:375–395. 26

[160] E. L. Pollock and J. Glosli; *Comments on P^3M, FMM and Ewald method for large periodic systems*; Comput. Phys. Commun. **1996**: **95**:93–110. 26

[161] J. V. L. Beckers, C. P. Lowe, and S. W. de Leeuw; *An Iterative PPPM Method for Simulating Coulombic Systems on Distributed Memory Parallel Computers*; Mol. Simul. **1998**:**20(6)**:369–383. 26

[162] H. Bekker, E. J. Dijkstra, *et al.*; *An Efficient, Box Shape Independent Non-Bonded Force and Virial Algorithm for Molecular Dynamics*; Mol. Simul. **1995**:**14(3)**:137–151. 26

[163] E. Clementi, G. Corongiu, *et al.*; *Parallelism in Computational Chemistry: Applications in Quantum and Statistical Mechanics*; Physica **1985**:**131B**:74–102. 26

[164] S. Plimpton and B. Hendrickson; *Parallel Molecular Dynamics Algorithms for Simulation of Molecular Systems*; in: Mattson [260] 114–132. 27

[165] T. Kerdcharoen, K. R. Liedl, and B. M. Rode; *Bidirectional Molecular Dynamics: Interpretation in Terms of a Modern Formulation of Classical Mechanics*; J. Comput. Phys. **1996**:**17(13)**:1564–1570. 27

[166] W. Smith; *Molecular dynamics on hypercube parallel computers*; Comput. Phys. Commun. **1991**:**62**:229–248. 27, 28

[167] ———; *Molecular dynamics on distributed memory (MIMD) parallel computers*; Theor. Chim. Acta **1993**:**84**:385–398. 27, 29, 41

[168] M. R. S. Pinches, D. J. Tildesley, and W. Smith; *Large Scale Molecular Dynamics on Parallel Computers using the Link-Cell Algorithm*; Mol. Simul. **1991**:**6**:51–87. 29, 30

[169] B. Hendrickson and S. Plimpton; *Parallel many-body simulations without all-to-all communication*; J. Para. Dist. Comput. **1995**:**27(1)**:15–25. 28

[170] A. Raine; *Systolic Loop Methods for Molecular Dynamics Simulation, Generalised for Macromolecules*; Mol. Simul. **1991**:**7**:59–69. 28

[171] C. Craven and G. Pawley; *Molecular dynamics of rigid polyatomic molecules on transputer arrays*; Comput. Phys. Commun. **1991**: **62(2-3)**:169–178. 28

[172] M. Surridge, D. J. Tildesley, *et al.*; *A parallel molecular dynamics simulation code for dialkyl cationic surfactants*; Parallel Comput. **1996**: **22(8)**:1053–1071. 28

[173] P. S. Lomdahl, D. M. Beazley, *et al.*; *Multimillion Particle Molecular-Dynamics on the CM00-5*; Int. J. Mod. Phys. C **1993**: **4(6)**:1075–1084. 30, 31

[174] D. Brown, J. H. Clarke, *et al.*; *A domain decomposition parallelization strategy for molecular dynamics simulations on distributed memory machines*; Comput. Phys. Commun. **1993**:**74(1)**:67–80. 30

[175] D. Brown, H. Minoux, and B. Maigret; *A domain decomposition parallel processing algorithm for molecular dynamics simulations of systems of arbitrary connectivity*; Comput. Phys. Commun. **1997**: **103(2–3)**:170–186; DOI. 31

[176] A. Windemuth; *Advanced Algorithms for Molecular Dynamics Simulation: The Program PMD*; in: Mattson [260] 151–169. 31

[177] T. Sterling, P. Messina, and P. H. Smith; *Enabling Technologies for Petaflops Computing* (MIT Press, **1995**); ISBN 0-262-69176-0. See also www.hq.nasa.gov/hpcc/petaflops/ and www.cacr.caltech.edu/pflops2/. 31

[178] A. H. L. Emmen; *A survey of Vector and Parallel Processors*; in: H. J. J. te Riele, T. J. Dekker, and H. A. van der Vorst (editors) *Algorithms and Applications on Vector and Parallel Computers*, *Special Topics in Supercomputing*, volume 3 (North Holland-Elsevier Science Publishers, **1987**) . 31

[179] G. Ciccotti, D. Frenkel, and I. R. McDonald (editors) *Simulation of liquids and solids: molecular dynamics and Monte Carlo methods in statistical mechanics* (North-Holland Physics Publishing, **1987**); ISBN 0-444-87062-8. 76

[180] F. F. Abraham; *How fast can cracks move? A research adventure in materials failure using millions of atoms and big computers*; Advances in Physics **2003**:**52(8)**:727–790. 33

[181] D. Rapaport; *Multibillion-atom molecular dynamics simulation: Design considerations for vector-parallel processing*; Comput. Phys. Commun. **2006**:**174(7)**:521–529; DOI. 33

[182] W. Wang and R. D. Skeel; *Fast evaluation of polarizable forces*; J. Chem. Phys. **2005**:**123(16)**:164107; DOI. 33

[183] F. Hedman and A. Laaksonen; *A Parallel Quantum Mechanical MD Simulation of Liquids*; Mol. Simul. **1998**:**20(5)**:265–284. 33

[184] T. Odagaki, J. Matsui, *et al.*; *The Role of Molecular Dynamics Simulations for the Study of Slow Dynamics*; Mol. Simul. **1994**: **12(3–6)**:299–304. 33

[185] P. Pulay; Ab initio *calculation of force constants and equilibrium geometries in polyatomic molecules. I Theory*; Mol. Phys. **1969**:**17**:197–204. 35, 36

[186] ———; *Analytical derivative methods in quantum chemistry*; in: K. Lawley (editor) *Ab initio methods in Quantum Chemistry, II* (John Wiley & Sons, New York, London, Sydney, **1987**) 241–287. 35, 36

[187] A. Laaksonen; *Computer simulation package for liquids and solids with polar interactions. I. McMOLDYN/$H_2$0: Aqueous systems*; Comput. Phys. Commun. **1986**:**42**:271. 35

[188] M. Frisch, G. Trucks, *et al.*; *Gaussian 94, (Revision B2)* **1995**. Gaussian, Inc., Pittsburgh, PA. 35

[189] K. Laasonen, M. Sprik, *et al.*; Ab-Initio *Liquid Water*; J. Chem. Phys. **1993**:**99(11)**:9080–9089. 35

[190] A. Soper and M. Phillips; *A determination of the structure of water at 25° C*; J. Chem. Phys. **1986**:**107**:47–60. 35

[191] H. Berendsen, J. Postma, *et al.*; *Interaction models for water in relation to protein hydration*; in: B. Pullman (editor) *Intermolecular Forces*; 331–342 (Reidel, Dordrecht, **1981**). 37

[192] K. Toukan and A. Rahman; *Molecular-dynamics of atomic motions in water*; Phys. Rev. B **1981**:**31**:2643–2648. 37, 38

[193] S. Nóse; *A molecular dynamics method for simulations in the canonical ensemble*; Mol. Phys. **1984**:**52**:255–268. 37

[194] W. Hoover; *Canonical dynamics: equilibrium phase-space distributions*; Phys. Rev. A **1985**:**31**:1695–1697. 37

[195] I. T. Forster; *Designing and Building Parallel Programs* (Addison-Wesley, **1995**). 39

[196] D. Fincham; *Parallel Computers and Molecular Simulation*; Mol. Simul. **1987**:**1**:1–45. 40

[197] W. Smith; *Molecular dynamics on hypercube parallel computers*; Comput. Phys. Commun. **1991**:**62**:229–248. 40

[198] M. Schmidt, K. Baldridge, *et al.*; *The General Atomic and Molecular Electronic Structure System*; J. Comput. Chem. **1993**:**14**:1347–1363. 41

[199] F. Hedman and A. Laaksonen; *An Embarrassingly Parallel* ab initio *MD Method for Liquids*; in: B. Kågström, E. Elmroth, *et al.* (editors) *Applied Parallel Computing, 4th International Workshop, PARA'98*, *Lecture Notes in Computer Science*, volume 1541; 224–229 (Springer-Verlag, **1998**). 41

[200] C. H. Koelbel, D. B. Loveman, *et al.*; *The High Performance Fortran Handbook* (MIT Press, **1994**). 44

[201] *Connection Machine Model CM-2 Technical Summary*; Technical Report HA87-4; Thinking Machines **1987**. 44

[202] TMC; Thinking Machines Corporation, Cambridge, Massachusetts; *Connection Machine CM-200 Technical Summary* **1992**. 44

[203] F. Fumi and M. Tosi; *Ionic sizes and Born repulsive parameters in the NaCl-type alkali halides. I. The Huggins-Mayer and Pauling forms*; J. Phys. Chem. Solids **1964**:**25**:31–43. 48

[204] J. W. Lewis and K. Singer; *Thermodynamic properties and self-diffusion of molten sodium chloride*; J. Chem. Soc. Faraday II **1975**:**71**:41–53. 48

[205] B. Boltjes and S. W. de Leeuw; *Molecular Dynamics on the Connection Machine using FORTRAN*; Mol. Simul. **1991**:**7**:1–42. 51

[206] D. M. York, T. A. Darden, and L. G. Pedersen; *The effect of long-range electrostatic interactions in simulations of macromolecular crystals - a comparison of the Ewald and truncated list methods*; J. Chem. Phys. **1993**:**99(10)**:8345–8348. 52

[207] M. Bergdorf, C. Peter, and P. H. Hunenberger; *Influence of cut-off truncation and artificial periodicity of electrostatic interactions in molecular simulations of solvated ions: A continuum electrostatics study*; J. Chem. Phys. **2003**:**119(17)**:9129–9144; DOI. 52

[208] D. M. Heyes; *Electrostatic potentials and fields in infinite point charge lattices*; J. Chem. Phys. **1981**:**74(3)**:1924–1929. 52

[209] T. Darden, D. York, and L. Pedersen; *Particle mesh Ewald: An $N\log(N)$ method for Ewald sums in large systems*; J. Chem. Phys. **1993**: **98(12)**:10089–10092; DOI. 52

[210] W. Y. Darrin York; *The fast Fourier Poisson method for calculating Ewald sums*; J. Chem. Phys. **1994**:**101(4)**:3298–3300. 52

[211] U. Essmann, L. Perera, *et al.*; *A smooth particle mesh Ewald method*; J. Chem. Phys. **1995**:**103(19)**:8577–8593. 52

[212] P. F. Batcho and T. Schlick; *New splitting formulations for lattice summations*; J. Chem. Phys. **2001**:**115(18)**:8312–8326. 52

[213] D. R. Wheeler and J. Newman; *A less expensive Ewald lattice sum*; Chem. Phys. Lett. **2002**:**366(5–6)**:537–543. 52

[214] Y. Shan, J. L. Klepeis, *et al.*; *Gaussian split Ewald: A fast Ewald mesh method for molecular simulation*; J. Chem. Phys. **2005**:**122(5)**:054101. 52

[215] J. E. Barnes and P. Hut; *A Hierarchical $O(N\log N)$ Force Calculation Algorithm*; Nature (London) **1986**:**324(4)**:446–449. 52

[216] J. Pérez-Jordá and W. Yang; *A simple $O(N\log N)$ algorithm for the rapid evaluation of particle-particle interactions*; Chem. Phys. Lett. **1995**: **247(4–6)**:484–490; DOI. 52

[217] Z.-H. Duan and R. Krasny; *An adaptive treecode for computing nonbonded potential energy in classical molecular systems*; J. Comput. Chem. **2001**:**22(2)**:184–195; DOI. 52

[218] I. Tsukerman; *Efficient computation of long-range electromagnetic interactions without Fourier transforms*; IEEE Trans. Magn. **2004**: **40(4)**:2158–2160. Part 2. 52

[219] E. T. Ong, K. M. Lim, *et al.*; *A fast algorithm for three-dimensional potential fields calculation: fast Fourier transform on multipoles*; J. Comput. Phys. **2003**:**192(1)**:244–261; DOI. 52

[220] R. D. Skeel, I. Tezcan, and D. J. Hardy; *Multiple grid methods for classical molecular dynamics*; J. Comput. Chem. **2002**:**23(6)**:673–684; DOI. 52

[221] J. A. Baker and R. O. Watts; *Monte Carlo studies of the dielectric properties of water-like models*; Mol. Phys. **1973**:**26(3)**:789–792. 52

[222] T. N. Heinz and P. H. Hunenberger; *Combining the lattice-sum and reaction-field approaches for evaluating long-range electrostatic interactions in molecular simulations*; J. Chem. Phys. **2005**:**123(3)**:034107; DOI. 52

[223] B. A. Luty, M. E. Davis, *et al.*; *A Comparison of Particle-Particle Particle-Mesh and Ewald Methods for Calculating Electrostatic Interactions in Periodic Molecular Systems*; Mol. Simul. **1994**:**14(1)**:11–20. 52

[224] T. M. Apostol; *Mathematical Analysis*; 2nd edition (Addison-Wesley, **1979**). 53

[225] J. P. Hansen; *Molecular-Dynamics Simulation of Coulomb Systems in Two and Three Dimensions*; in: Ciccotti and Hoover [259] 89–129. 53

[226] J. Kolafa and J. W. Perram; *Cutoff Errors in the Ewald Summation Formulae for Point Charge Systems*; Mol. Simul. **1992**:**9**:351–368. 53

[227] S. W. de Leeuw, J. W. Perram, and E. R. Smith; *Simulation of electrostatic systems in periodic boundary conditions II. Equivalence of boundary conditions*; Proc. R. Soc. London **1980**:**A 373**:57–66. 54

[228] ———; *Simulation of electrostatic systems in periodic boundary conditions III. Further theory and applications*; Proc. R. Soc. London **1983**:**A 388**:177–193. 54

[229] D. Fincham; *Optimisation of the Ewald Sum*; CCP5 Information Quarterly **1993**:**38**:17–24. 62

[230] W. H. Press and G. B. Rybicki; *Fast Algorithm for spectral-analysis of unevenly sampled data*; Astrophys J. **1989**:**338(1)**:277–280. Part 1. 62

[231] J. W. Cooley and J. W. Tukey; *An Algorithm for the Machine Calculation of Complex Fourier Series*; Math. Comp. **1965**:**19(90)**:297–301. 62

[232] A. Brandt; *Multilevel computations of integral transforms and particle interactions with oscillatory kernels*; Comput. Phys. Commun. **1991**: **65(1–3)**:24–38; DOI. 62

[233] A. Dutt and V. Rokhlin; *Fast Fourier Transforms for Nonequispaced Data*; SIAM J. Sci. Stat. Comput. **1993**:**14(6)**:1368–1393. 62

[234] G. Beylkin; *On the Fast Fourier Transform of Functions With Singularities*; Appl. Comput. Harmon. A **1995**:**2**:363–381. 62

[235] Q. Liu and N. Nguyen; *An accurate algorithm for nonuniform fast Fourier transforms (NUFFTs)*; IEEE Microw. G. W. Lett. **1998**:**8(1)**:18–20. 62

[236] G. Steidl; *A note on fast Fourier transforms for nonequispaced grids*; Adv. Comput. Math. **1998**:**9(3–4)**:337–352. 62

[237] J. Fessler and B. Sutton; *Nonuniform fast Fourier transforms using min-max interpolation*; IEEE Trans. Sign. Proc. **2003**:**51(2)**:560–574. 62

[238] C. Anderson and M. D. Dahleh; *Rapid computation of the discrete Fourier transform*; SIAM J. Sci. Comput. **1996**:**17(4)**:913–919. 62

[239] A. Duijndam and M. Schonewille; *Nonuniform fast Fourier transform*; Geophysics **1999**:**64(2)**:539–551. 62

[240] J. P. Boyd; *A fast algorithm for Chebyshev, Fourier, and sinc interpolation onto an irregular grid*; J. Comput. Phys. **1992**:**103(2)**:243–257. 62

[241] A. Ware; *Fast approximate Fourier transforms for irregularly spaced data*; SIAM Rev. **1998**:**40(4)**:838–856. 62

[242] A. Nieslony and G. Steidl; *Approximate factorizations of Fourier matrices with nonequispaced knots*; Lin. Alg. Appl. **2003**:**366**:337–351; DOI. 62

[243] D. Potts, G. Steidl, and M. Tasche; *Fast Fourier transforms for nonequispaced data: A tutorial*; in: J. Benedetto and P. Ferreira (editors) *Modern Sampling Theory: Mathematics and Applications*; Applied and Numerical Harmonic Analysis Series; chapter 12, 249–274 (Birkhauser Boston, **2001**). 62, 65, 68

[244] S. Kunis and D. Potts; *NFFT2.0*; University of Lübeck; Institute of Mathematics, D-23560 Lübeck, Germany **2004**. http://www.math.uni-luebeck.de/potts/nfft/. 62, 69

[245] F. Hedman and A. Laaksonen; *Ewald summation based on nonuniform fast Fourier transform*; Chem. Phys. Lett. **2006**:DOI. Article in press. 69

[246] M. Frigo and S. G. Johnson; *The Design and Implementation of FFTW3*; Proceedings of the IEEE **2005**:**93(2)**:216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation". 69

[247] H. Jeffreys; *Scientific Inference*; 3rd edition (Cambridge University Press, **1973**); ISBN 0-521-08446-6. 72

[248] E. T. Jaynes; *Probability theory: the logic of science* (Cambridge University Press, **2003**); ISBN 0-521-59271-2. 72

[249] G. L. Bretthorst; *Bayesian Spectrum Analysis and Parameter Estimation*; *Lecture Notes in Statistics*, volume 48 (Springer-Verlag, **1988**). 72

[250] G. D'Agostini; *Bayesian Reasoning in Data Analysis: a critical introduction* (World Scientific, **2003**); ISBN 981-238-356-5. 72

[251] P. C. Gregory; *Bayesian logical data analysis for the physical sciences: a comparitive approach with Mathematica©support* (Cambridge University Press, **2005**); ISBN 0-521-84150-X. 72

[252] G. Hummer; *Position-dependent diffusion coefficients and free energies from Bayesian analysis of equilibrium and replica molecular dynamics simulations*; New Journal of Physics **2005**:**7**:34; DOI. 72

[253] A. Katz; *Principles of Statistical Mechanics: the information theory approach* (W.H. Freeman, **1967**). 72

[254] A. Hobson; *Concepts in Statistical Mechanics* (Gordon and Breach, Science Publishers, Inc., **1971**); ISBN 0-677-03240-4. 72

[255] R. Baierlein; *Atoms and Information Theory. An Introduction to Statistical Mechanics* (W.H. Freeman, **1971**); ISBN 0-7167-0332-7. 72

[256] W. T. Grandy, Jr.; *Foundations of Statistical Mechanics. Volume I: Equilibrium Theory* (D. Reidel Publishing Company, **1987**); ISBN 90-277-2489-X. 72

[257] ———; *Foundations of Statistical Mechanics. Volume II: Nonequilibrium Phenomena* (D. Reidel Publishing Company, **1987**); ISBN 90-277-2469-3. 72

[258] D. S. Betts and R. E. Turner; *Introductory Statistical Mechanics* (Addison-Wesley, **1992**); ISBN 0-201-54421-0. 72

[259] G. Ciccotti and W. G. Hoover (editors); *Proccedings of the International School of Physics "Enrico Fermi". Course XCVII. Molecular-Dynamics Simulation of Statistical-Mechanical Systems* (**1986**). 76, 91

[260] T. G. Mattson (editor); *Parallel Computing in Computational Chemistry*; number 592 in ACS Symposium Series (**1995**). 86, 87