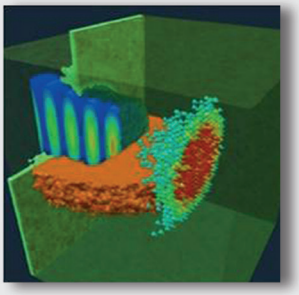
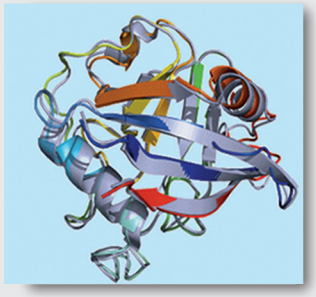
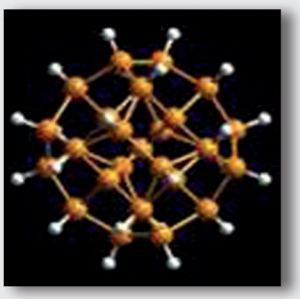
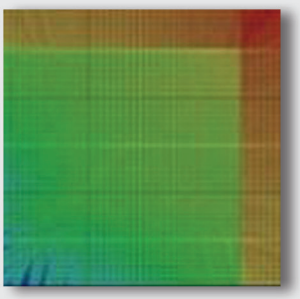
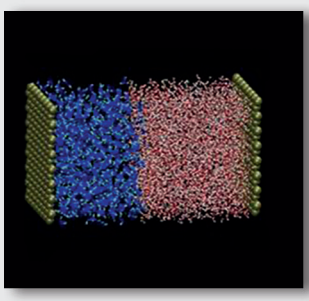










Fast Methods for Long-Range Interactions in Complex Systems

edited by Godehard Sutmann, Paul Gibbon, Thomas Lippert

JÜLICH
FORSCHUNGSZENTRUM

Mitglied der Helmholtz-Gemeinschaft

Schriften des Forschungszentrums Jülich
IAS Series Volume 6

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

Fast Methods for Long-Range Interactions in Complex Systems

edited by Godehard Sutmann, Paul Gibbon, Thomas Lippert

Summer School, 6 – 10 September 2010
Forschungszentrum Jülich GmbH
Lecture Notes

Schriften des Forschungszentrums Jülich
IAS Series Volume 6

ISSN 1868-8489 ISBN 978-3-89336-714-6

Bibliographic information published by the Deutsche Nationalbibliothek.
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Publisher and Distributor:	Forschungszentrum Jülich GmbH Zentralbibliothek 52425 Jülich Phone +49 (0) 24 61 61-53 68 · Fax +49 (0) 24 61 61-61 03 e-mail: zb-publikation@fz-juelich.de Internet: http://www.fz-juelich.de/zb
Cover Design:	Grafische Medien, Forschungszentrum Jülich GmbH
Printer:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2011

Schriften des Forschungszentrums Jülich
IAS Series Volume 6

ISSN 1868-8489
ISBN 978-3-89336-714-6

The complete volume ist freely available on the Internet on the Jülicher Open Access Server (JUWEL) at http://www.fz-juelich.de/zb/juwel

Persistent Identifier: urn:nbn:de:0001-2011051907
Resolving URL: http://www.persistent-identifier.de/?link=610

Preface

Computer simulations of complex particle systems have a still increasing impact in a broad field of physics, e.g. astrophysics, statistical physics, plasma physics, material sciences, physical chemistry or biophysics, to name a few. Along with the development of computer hardware, which today shows a performance in the range of PFlop/s, it is essential to develop efficient and scalable algorithms which solve the physical problem. Since with more powerful computer systems usually also the problem size is increased, it is important to implement optimally scaling algorithms, which increase the computational effort proportionally to the number of particles.

Especially in fields, where long-range interactions between particles have to be considered the numerical effort is usually very large. Since most of interesting physical phenomena involve electrostatic, gravitational or hydrodynamic effects, the proper inclusion of long range interactions is essential for the correct description of systems of interest. Since in principle, long range interactions are $O(N^2)$ for open systems or include infinite lattice sums in periodic systems, fast implementations rely on approximations. Although, in principle, various methods might be considered as *exact representations* of the problem, approximations with controllable error thresholds are developed. Since different boundary conditions or dielectric properties require the application of appropriate methods, there is not only one method, but various classes of methods developed. E.g. the inclusion of different symmetries in the system (1d- ,2d- or 3d-periodic systems), the presence of interfaces or including inhomogeneous dielectric properties, require the implementation of different electrostatic methods. Furthermore, the interdisciplinary character of the problem led to the fact that either very similar methods or complementary methods were developed independently in parallel in different disciplines or were *discovered* in other research areas and adopted to other fields.

Therefore the present school does not only focus on one method, but introduces a spectrum of different fast algorithms:

- Fourier transform based methods
  - Particle-particle particle-mesh method (P^3M)
  - MMM-methods (MMM1d, MMM2d)
  - Fast summation based on non-equidistant Fast Fourier Transform (NFFT)
- Hierarchical methods
  - Fast Multipole Method (FMM)
  - Barnes-Hut Tree method
  - Multigrid based methods
- Local cutoff-approximations
  - Wolf summation

In addition to the mathematical description of the methods, focus is given to the parallelization and implementation on parallel computers. Therefore, also a special session is

devoted to an introduction to parallel programming and various parallelization paradigms (MPI, OpenMP, PThreads). Practical sessions complement the talks on theoretical foundations and implementation issues of different algorithms.

This first summer school on *Fast Methods for Long-Range Interactions in Complex Systems* brings together a number of German experts in the fields of mathematical methods and development of algorithms. The presented methods and their efficient parallel implementation are part of the German network project *ScaFaCoS* (Scalable Fast Coulomb Solvers), sponsored by the German Ministry for Science and Education (BMBF), which aims to build a publicly accessible parallel library.

Financial support of this school came from the Wilhelm and Else Heraeus Foundation, which is gratefully acknowledged. This preface also gives the opportunity to thank all the speakers, having prepared the lectures and practical sessions. Also we would like to express most cordial gratitudes to Monika Marx, who has put lots of effort in the realization of the present poster abstracts, lecture notes, WEB pages and lots of plannings. We are also most grateful to Elke Bielitza who was indispensable for this school by taking care of logistical details, transports, registration, catering and a lot more. Further thanks are expressed to Johannes Grotendorst who gave valuable advice and expertise of organizational details and also to Oliver Bücker, René Halver, Thomas Plaga and Marga Vaeßen for technical and administrational support.

Jülich, September 2010

Godehard Sutmann
Paul Gibbon
Thomas Lippert

Contents

Classical Particle Simulations

Godehard Sutmann

Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)
Forschungszentrum Jülich, 52425 Jülich, Germany
E-mail: g.sutmann@fz-juelich.de

An introduction to classical particle dynamics simulation is presented. In addition to some historical notes, an overview is given over particle models and integrators and pathways are shown how to build more abstract coarse grain models of particles or groups of particles in order to reduce the number of degrees of freedom, thereby increasing efficiency and performance.

1 Introduction

Computer simulation methods have become a powerful tool to solve many-body problems in statistical physics[1], physical chemistry[2] and biophysics[3]. Although both the theoretical description of complex systems in the framework of statistical physics as well as the experimental techniques for detailed microscopic information are rather well developed it is often only possible to study specific aspects of those systems in great detail via simulation. On the other hand, simulations need specific input parameters that characterize the system in question, and which come either from theoretical considerations or are provided by experimental data. Having characterized a physical system in terms of model parameters, simulations are often used both to solve theoretical models beyond certain approximations and to provide a hint to experimentalists for further investigations. In the case of big experimental facilities it is often even required to prove the potential outcome of an experiment by computer simulations. In this sense it can be stated that the field of computer simulations has developed into a very important branch of science, which on the one hand helps theorists and experimentalists to go beyond their *inherent limitations* and on the other hand is a scientific field on its own. Therefore, simulation science has often been called the *third pillar* of science, complementing theory and experiment.

The traditional simulation methods for many-body systems can be divided into two classes, i.e. stochastic and deterministic simulations, which are largely represented by the Monte Carlo (MC) method[1,4] and the molecular dynamics[5,6] (MD) method, respectively. Monte Carlo simulations probe the configuration space by trial moves of particles. Within the so-called Metropolis algorithm, the energy change from step n to $n+1$ is used as a trigger to accept or reject a new configuration. Paths towards lower energy are always accepted, those to higher energy are accepted with a probability governed by Boltzmann statistics. This algorithm ensures the correct limiting distribution and properties of a given system can be calculated by averaging over all Monte Carlo moves within a given statistical ensemble (where one move means that every degree of freedom is probed once on average). In contrast, MD methods are governed by the system Hamiltonian and consequently Hamilton's equations of motion[7,8]

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

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

In order to characterise a given system and to simulate its complex behavior, a model for interactions between system constituents is required. This model has to be tested against experimental results, i.e. it should reproduce or approximate experimental findings like distribution functions or phase diagrams, and theoretical constraints, i.e. it should obey certain fundamental or limiting laws like energy or momentum conservation.

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

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

An important issue of simulation studies are the accessible time- and length-scales which can be covered by particle simulations. At this point it is important to consider different concepts of particles:

- Ab-initio models: particles are described consistently together with the electronic structure. Interactions are not restricted to a fixed force field description but the interactions are dependent on the configuration of particles in the system. The calculation of interactions is therefore a many-body problem. Depending on the approximations,

used for the solution of electronic configurations, the computational complexity is large[9,10].

- All-atom model: molecules are resolved on an atomistic level and each atom is represented with a force-field description. Molecular degrees of freedom may contain flexible bonds or rigid body constraints.

- Unified-atom model: molecules are again described on an atomistic level, but some intra-molecular atom-groups are described as single interaction center. This technique often applies to light atoms, e.g. hydrogens, in order to allow for a larger timestep in the equations of motion.

- Unified-molecule model: whole molecules or even groups of molecules are considered as one particle. Interactions are e.g. described by multipole moments, introducing a non-isotropic interaction between molecules. This description is used e.g. in the simulation of liquid crystals[11] or polar liquids[12,97].

- Coarse-grain field representation: particles represent large groups of particles, which exhibit dynamics on mesoscopic time- and length-scales. Fluctuations in particle motion thereby reflects thermal behavior. Interactions are modeled such to represent macroscopic behavior, governed by Navier-Stokes equations, as a limiting case.

Since in this review, classical types of particle simulations are considered, *ab initio* methods are discarted here. Usually they have such a large complexity that number of particles and consequently, time and length scales are strongly limited with respect to classical methods. However, some multiscale methods, being developed during the last years, allow to couple high resolution methods (*ab initio*) with atomistic and coarse-grain models, thereby treating only a small subset of degrees of freedom on a very detailed level.

Fig.1 shows a schematic representation for different types of simulations. It is clear that the more detailed a simulation technique operates, the smaller is the accessibility of long times and large length scales. Therefore quantum simulations, where electronic fluctuations are taken into account, are located in the part of the diagram of very short time and length scales which are typically of the order of $\mathring{A}$ and ps. Classical molecular dynamics approximates electronic distributions in a rather coarse-grained fashion by putting either fixed partial charges on interaction sites or by adding an approximate model for polarization effects. In both cases, the time scale of the system is not dominated by the motion of electrons, but the time of intermolecular collision events, rotational motions or intramolecular vibrations, which are orders of magnitude slower than those of electron motions. Consequently, the time step of integration is larger and trajectory lengths are of order ns and accessible lengths of order $10 - 100$ $\mathring{A}$. If one considers tracer particles in a solvent medium, where one is not interested in a detailed description of the solvent, one can apply Brownian dynamics, where the effect of the solvent is hidden in average quantities. Since collision times between tracer particles is very long, one may apply larger timesteps. Furthermore, since the solvent is not simulated explicitly, the lengthscales may be increased considerably. Finally, if one is interested not in a microscopic picture of the simulated system but in macroscopic quantities, the concepts of hydrodynamics may be applied, where the system properties are hidden in effective numbers, e.g. density, viscosity or sound velocity.

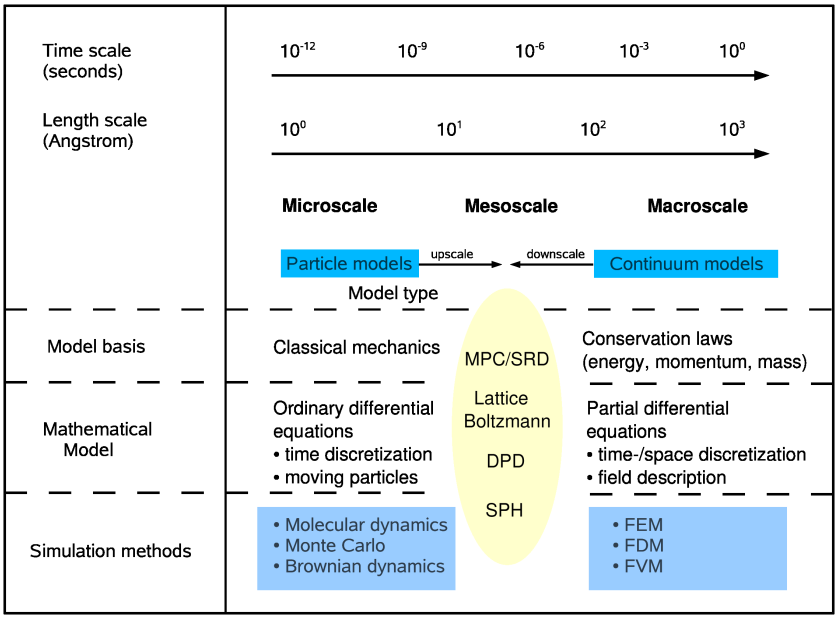


Figure 1: Schematic of different time- and length-scales, occurring from microscopic to macroscopic dimensions. Due to recent developments of techniques like Stochastic Rotation Dynamics (SRD) or Lattice Boltzmann techniques, which are designed to simulate the mesoscopic scales, there is the potential to combine different methods in a multiscale approach to cover a broad spectrum of times and lengths.

It is clear that the performance of particle simulations strongly depends on the computer facilities at hand. The first studies using MD simulation techniques were performed in 1957 by B. J. Alder and T. E. Wainright[13] who simulated the phase transition of a system of hard spheres. The general method, however, was presented only two years later[14]. In these early simulations, which were run on an IBM-704, up to 500 particles could be simulated, for which 500 collisions per hour were calculated. Taking into account 200000 collisions for a production run, these simulations lasted for more than two weeks. Since the propagation of hard spheres in a simulation is event driven, i.e. it is determined by the collision times between two particles, the propagation is not based on an integration of the equations of motion, but rather the calculation of the time of the next collision, which results in a variable time step in the calculations.

The first MD simulation which was applied to atoms interacting via a continuous potential was performed by A. Rahman in 1964. In this case, a model system for Argon was simulated and not only binary collisions were taken into account but the interactions were modeled by a Lennard-Jones potential and the equations of motion were integrated with a finite difference scheme. This work may be considered as seminal for dynamical calculations. It was the first work where a numerical method was used to calculate dynamical quantities like autocorrelation functions and transport coefficients like the diffusion coefficient for a realistic system. In addition, more involved characteristic functions like the dynamic van Hove function and non-Gaussian corrections to diffusion were evaluated. The

calculations were performed for 864 particles on a CDC 3600, where the propagation of all particles for one time step took $\approx 45\ s$. The calculation of 50000 timesteps then took more than three weeks! [a]

With the development of faster and bigger massively parallel architectures the accessible time and length scales are increasing for all-atom simulations. In the case of classical MD simulations it is a kind of competition to break new world records by carrying out demonstration runs of larger and larger particle systems[15–18]. In a recent publication, it was reported by Germann and Kadau[19] that a trillion-atom (10^{12} particles!) simulation was run on an IBM BlueGene/L machine at Lawrence Livermore National Laboratory with 212992 PowerPC 440 processors with a total of 72 TB memory. This run was performed with the memory optimised program SPaSM[20,21] (Scalable Parallel Short-range Molecular dynamics) which, in single-precision mode, only used 44 Bytes/particle. With these conditions a simulation of a Lennard-Jones system of $N = (10000)^3$ was simulated for 40 time steps, where each time step used $\approx 50secs$ wall clock time.

Concerning the accessible time scales of all-atom simulations, a numerical study, carried out by Y. Duan and P. A. Kollman in 1998 still may be considered as a milestone in simulation science. In this work the protein folding process of the subdomain HP-36 from the villin headpiece[22,23] was simulated up to $1\ \mu s$. The protein was modelled with a 596 interaction site model dissolved in a system of 3000 water molecules. Using a timestep of integration of $2 \times 10^{-15} s$, the program was run for 5×10^8 steps. In order to perform this type of calculation, it was necessary to run the program several months on a CRAY T3D and CRAY T3E with 256 processors. It is clear that such kind of a simulation is exceptional due to the large amount of computer resources needed, but it was nonetheless a kind of milestone pointing to future simulation practices, which are nowadays still not standard, but nevertheless exceptionally applied[24].

Classical molecular dynamics methods are nowadays applied to a huge class of problems, e.g. properties of liquids, defects in solids, fracture, surface properties, friction, molecular clusters, polyelectrolytes and biomolecules. Due to the large area of applicability, simulation codes for molecular dynamics were developed by many groups. On the internet homepage of the Collaborative Computational Project No.5 (CCP5)[25] a number of computer codes are assembled for condensed phase dynamics. During the last years several programs were designed for parallel computers. Among them, which are partly available free of charge, are, e.g., Amber/Sander[26], CHARMM[27], NAMD[28], NWCHEM[29], GROMACS[30] and LAMMPS[31].

Although, with the development of massively parallel architectures and highly scalable molecular dynamics codes, it has become feasible to extend the time and length scales to *relatively* large scales, a lot of processes are still beyond technical capabilities. In addition, the time and effort for running these simulations is enormous and it is certainly still far beyond of standard. A way out of this dilemma is the invention of new simulation of methodological approaches. A method which has attracted a lot of interest recently is to coarse grain all-atom simulations and to approximate interactions between individual atoms by interactions between whole groups of atoms, which leads to a smaller number of degrees of freedom and at the same time to a smoother energy surface, which on the one hand side increases the computation between particle interactions and on the other

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

hand side allows for a larger time step, which opens the way for simulations on larger time and length scales of physical processes[32]. Using this approach, time scales of more than 1 $\mu secs$ can now be accessed in a fast way[33,34], although it has to be pointed out that coarse grained force fields have a very much more limited range of application than all-atom force fields. In principle, the coarse graining procedure has to be outlined for every different thermodynamic state point, which is to be considered in a simulation and from that point of view coarse grain potentials are not transferable in a straight forward way as it is the case for a wide range of all-atom force field parameters.

2 Models for Particle Interactions

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

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

where $\mathbf{p}$ is the momentum, m the mass of the particles and u and $u^{(3)}$ are pair and three-body interaction potentials. $\mathcal{H}_1$ is an external part, which can include time dependent effects or external sources for a force. All simulated objects are defined within a model description. Often a precise knowledge of the interaction between atoms, molecules or surfaces are not known and the model is constructed in order to describe the main features of some observables. Besides boundary conditions, which are imposed, it is the model which completely determines the system from the physical point of view. In classical simulations the *objects* are most often described by point-like centers which interact through pair- or multibody interaction potentials. In that way the highly complex description of electron dynamics is abandoned and an effective picture is adopted where the main features like the hard core of a particle, electric multipoles or internal degrees of freedom of a molecules are modeled by a set of parameters and (most often) analytical functions which depend on the mutual position of particles in the configuration. Since the parameters and functions give a complete information of the system's energy as well as the force acting on each particle through $\mathbf{F} = -\nabla U$, the combination of parameters and functions is also called a *force field*[35]. Different types of force field were developed during the last ten years. Among them are e.g. MM3[36], MM4[37], Dreiding[38], SHARP[39], VALBON[40], UFF[41], CFF95[42], AMBER[43] CHARMM[44], OPLS[45] and MMFF[46].

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

A second important difference between interactions is the spatial extent of the potential, classifying it into short and long range interactions. If the potential drops down to zero faster than r^{-d}, where r is the separation between two particles and d the dimension of

the problem, it is called short ranged, otherwise it is long ranged. This becomes clear by considering the integral

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

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

There may be different terms contributing to the interaction potential between particles, i.e. there is no universal expression, as one can imagine for first principles calculations. In fact, contributions to interactions depend on the model which is used and this is the result of collecting various contributions into different terms, coarse graining interactions or imposing constraints, to name a few. Generally one can distinguish between bonded and non-bonded terms, or intra- and inter-molecular terms. The first class denotes all contributions originating between particles which are closely related to each other by constraints or potentials which guaranty defined particles as close neighbors. The second class denotes interactions between particles which can *freely* move, i.e. there are no defined neighbors, but interactions simply depend on distances.

A typical form for a (so called) force field (e.g. AMBER[26]) looks as follows

$$\begin{aligned} \mathcal{U} = & \sum_{\text{bonds}} K_r(r - r_{eq})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{eq})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + \cos(n\phi - \gamma)] \\ & + \sum_{i<j} \left[\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right] + \sum_{\text{H-bonds}} \left[\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right] + \sum_{i<j} \frac{q_i q_j}{r_{ij}} \end{aligned} \tag{4}$$

In the following, short- and long-range interaction potentials and methods are briefly described in order to show differences in their algorithmical treatment.

In the following two examples shall illustrate the different treatment of short- and long range interactions.

2.1 Short Range Interactions

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

$$U = \sum_{i<j}^{N} u(r_{ij} | r_{ij} < R_c) + U_{lrc} \tag{5}$$

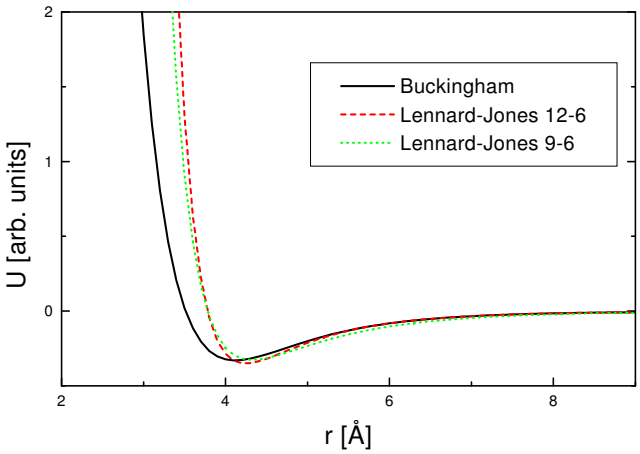


Figure 2: Comparison between a Buckingham-, Lennard-Jones (12-6) and Lennard-Jones (9-6) potential.

The long-range correction is thereby given as

$$U_{lrc} = 2\pi N \rho_0 \int_{R_c}^{\infty} dr\, r^2 g(r) u(r) \tag{6}$$

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

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

$$u^B_{\alpha\beta}(r_{ij}) = A_{\alpha\beta} e^{-B_{\alpha\beta} r_{ij}} - \frac{D_{\alpha\beta}}{r^6_{ij}} \tag{7}$$

and the Lennard-Jones potential

$$u^{LJ}_{\alpha\beta}(r_{ij}) = 4\epsilon_{\alpha\beta} \left(\left(\frac{\sigma_{\alpha\beta}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{\alpha\beta}}{r_{ij}} \right)^{6} \right) \tag{8}$$

which are compared in Fig.2. In Eqs.7,8 the indices α, β indicate the species of the particles, i.e. there are parameters A, B, D in Eq.7 and ϵ, σ in Eq.8 for intra-species interactions ($\alpha = \beta$) and cross species interactions ($\alpha \neq \beta$). For the Lennard-Jones potential

the parameters have a simple physical interpretation: ϵ is the minimum potential energy, located at $r = 2^{1/6}\sigma$ and σ is the diameter of the particle, since for $r < \sigma$ the potential becomes repulsive. Often the Lennard-Jones potential gives a reasonable approximation of a *true* potential. However, from exact quantum ab initio calculations an exponential type repulsive potential is often more appropriate. Especially for dense systems the too steep repulsive part often leeds to an overestimation of the pressure in the system. Since computationally the Lennard-Jones interaction is quite attractive the repulsive part is sometimes replaced by a weaker repulsive term, like $\propto r^{-9}$. The Lennard-Jones potential has another advantage over the Buckingham potential, since there are combining rules for the parameters. A common choice are the Lorentz-Berelot combining rules

$$\sigma_{\alpha\beta} = \frac{\sigma_{\alpha\alpha} + \sigma_{\beta\beta}}{2} \qquad , \qquad \epsilon_{\alpha\beta} = \sqrt{\epsilon_{\alpha\alpha}\epsilon_{\beta\beta}} \tag{9}$$

This combining rule is, however, known to overestimate the well depth parameter. Two other commonly known combining rules try to correct this effect, which are Kong[47] rules

$$\sigma_{\alpha\beta} = \left[\frac{1}{2^{13}} \frac{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{12}}{\sqrt{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{6}\epsilon_{\beta\beta}\sigma_{\beta\beta}^{6}}} \left(1 + \left(\frac{\epsilon_{\beta\beta}\sigma_{\beta\beta}^{12}}{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{12}} \right)^{1/13} \right)^{13} \right]^{1/6} \tag{10}$$

$$\epsilon_{\alpha\beta} = \frac{\sqrt{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{6}\epsilon_{\beta\beta}\sigma_{\beta\beta}^{6}}}{\sigma_{\alpha\beta}^{6}} \tag{11}$$

and the Waldman-Kagler[48] rule

$$\sigma_{\alpha\beta} = \left(\frac{\sigma_{\alpha\alpha}^{6} + \sigma_{\beta\beta}^{6}}{2} \right)^{1/6} \qquad , \qquad \epsilon_{\alpha\beta} = \frac{\sqrt{\epsilon_{\alpha\alpha}\sigma_{\alpha\alpha}^{6}\epsilon_{\beta\beta}\sigma_{\beta\beta}^{6}}}{\sigma_{\alpha\beta}^{6}} \tag{12}$$

In a recent study[49] of Ar-Kr and Ar-Ne mixtures, these combining rules were tested and it was found that the Kong rules give the best agreement between simulated and experimental pressure-density curves. An illustration of the different combining rules is shown in Fig.3 for the case of an Ar-Ne mixture.

Since there are only relatively few particles which have to be considered for the interaction with a tagged particle (i.e. those particles within the cutoff range), it would be a computational bottleneck if in any time step all particle pairs would have to be checked whether they lie inside or outside the interaction range. This becomes more and more a problem as the number of particles increases. A way to overcome this bottleneck is to introduce list techniques. The first implementation dates back to the early days of molecular dynamics simulations. In 1967, Verlet introduced a list[50], where at a given time step all particle pairs were stored within a range $R_c + R_s$, where R_s is called the skin radius and which serves as a reservoir of particles, in order not to update the list in each time step (which would make the list redundant). Therefore, in a force routine, not all particles have to tested, whether they are in a range $r_{ij} < R_c$, but only those particle pairs, stored in the list. Since particles are moving during the simulation, it is necessary to update the list from time to time. A criterion to update the list could be, e.g.

$$\max_i |\mathbf{r}_i(t) - \mathbf{r}_i(t_0)| \geq \frac{R_s}{2} \tag{13}$$

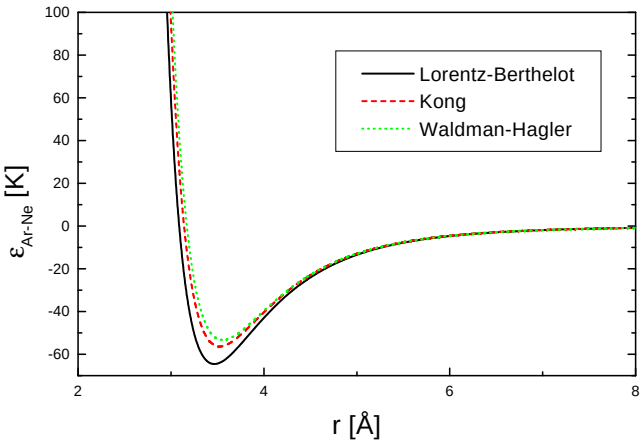


Figure 3: Resulting cross-terms of the Lennard-Jones potential for an Ar-Ne mixture. Shown is the effect of different combining rules (Eqs.9-12). Parameters used are $\sigma_{Ar} = 3.406\ \mathring{A}, \epsilon_{Ar} = 119.4\ K$ and $\sigma_{Ne} = 2.75\ \mathring{A}, \epsilon_{Ne} = 35.7\ K$.

where t_0 is the time from the last list update. This ensures that particles cannot move from the outside region into the cutoff sphere without being recognized. This technique, though efficient, has still complexity $\mathcal{O}(N^2)$, since at an update step, *all* particle pairs have to be checked for their mutual distances. Another problem arises when simulating many particles, since the memory requirements are relatively large (size of the list is $4\pi(R_c + R_s)^3 \rho\, N/3$). There is, of course also the question, how large the skin radius should be chosen. Often, it is chosen as $R_s = 1.5\sigma$. In Ref.[51] it was shown that an optimal choice strongly depends on the number of particles in the system and an optimization procedure was outlined.

An alternative list technique, which scales linearly with the number of particles is the linked-cell method[52,53]. The linked-cell method starts with subdividing the whole system into cubic cells and sorting all particles into these cells according to their position. The size of the cells, L_c, is chosen to be $L_c \leq L_{Box}/\text{floor}(L_{Box}/R_c)$, where L_{Box} is the length of the simulation box. All particles are then sorted into a list array of length N. The list is organized in a way that particles, belonging to the same cell are linked together, i.e. the entry in the list referring to a particle points directly to the entry of a next particle inside the same cell. A zero entry in the list stops the search in the cell and a next cell is checked for entries. This technique not only has computational complexity of $O(N)$, since the sorting into the cells and into the N-dimensional array is of $O(N)$, but also has memory requirements which only grow linearly with the number of particles. These features make this technique very appealing. However, the technique is not well vectorizable and also the addressing of next neighbors in the cells require indirect access (e.g. `i=index(i)`), which may lead to cache misses. In order not to miss any particle pair in the interactions every box has to have a neighbor region in each direction which extends to R_c. In the case, where $L_c \geq R_c$, every cell is surrounded by 26 neighbor cells in three dimensional systems. This gives rise to the fact that the method gives only efficiency gains if $L_{Box} \geq 4R_c$, i.e. subdividing each box direction into more than 3 cells. In order to approximate the

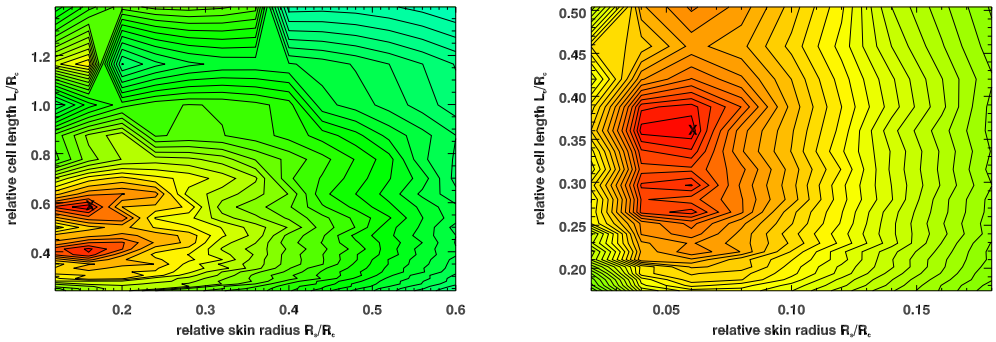


Figure 4: Contour plots of the performance for the combination of linked-cell and Verlet list as a function of the cell length and the size of the skin radius. Crosses mark the positions predicted from an optimization procedure[54]. Test systems were composed of 4000 Lennard-Jones particles with $R_c = 2.5\,\sigma$ at temperature $T = 1.4\,\epsilon/k_B$. Left: $\rho = 0.75/\sigma^3$. Right: $\rho = 2.0/\sigma^3$.

cutoff sphere in a better way by cubic cells, one may reduce the cell size and simultaneously increasing the total number of cells. In an optimization procedure[51], it was found that a reduction of cell sizes to $L_c = R_c/2$ or even smaller often gives very much better results.

It is, of course, possible to combine these list techniques, i.e. using the linked-cell technique in the update step of the Verlet list. This reduces the computational complexity of the Verlet list to $\mathcal{O}(N)$ while fully preserving the efficiency of the list technique. It is also possible to model the performance of this list combination and to optimize the length of the cells and the size of the skin radius. Figure 4 shows the result of a parameter study, where the performance of the list was measured as a function of (L_c, R_s). Also shown is the prediction of parameters coming out of an optimization procedure[54].

2.2 Long Range Interactions

Long range interactions essentially require to take all particle pairs into account for a proper treatment of interactions. This may become a problem, if periodic boundary conditions are imposed to the system, i.e. formally simulating an infinite number of particles (no explicit boundaries imply infinite extend of the system). Therefore one has to devise special techniques to treat this situation. On the other hand one also has to apply fast techniques to overcome the inherent $\mathcal{O}(N^2)$ complexity of the problem, since for large numbers of particles this would imply an intractable computational bottleneck. In general one can classify algorithms for long range interactions into the following system:

- Periodic boundary conditions
  - Grid free algorithms, e.g. Ewald summation method[55–57]
  - Grid based algorithms, e.g. Smoothed Particle Mesh Ewald[58,59], Particle-Particle Particle-Mesh method[60–62]
- Open boundary conditions

- Grid free algorithms, e.g. Fast Multipole Method[63–68] (FMM), Barnes-Hut Tree method[69,70]
- Grid based algorithms, e.g. Particle-Particle Particle-Multigrid method[71] (P^3Mg), Particle Mesh Wavelet method[72] (PMW)

In the following two important members of these classes will be described, the Ewald summation method and the Fast Multipole Method.

2.2.1 Ewald Summation Method

The Ewald summation method originates from crystal physics, where the problem was to determine the Madelung constant[73], describing a factor for an effective electrostatic energy in a perfect periodic crystal. Considering the electrostatic energy of a system of N particles in a cubic box and imposing periodic boundary conditions, leads to an equivalent problem. At position $\mathbf{r}_i$ of particle i, the electrostatic potential, $\phi(\mathbf{r}_i)$, can be written down as a lattice sum

$$\phi(\mathbf{r}_i) = \sum_{\mathbf{n}}{}^{\dagger} \sum_{j=1}^{N} \frac{q_j}{\|\mathbf{r}_{ij} + \mathbf{n}L\|} \tag{14}$$

where $\mathbf{n} = (n_x, n_y, n_z), n_\alpha \in \mathbb{Z}$ is a vector along cartesian coordinates and L is the length of the simulation box. The sign "†" means that $i \neq j$ for $\|\mathbf{n}\| = 0$.

Eq. (14) is conditionally convergent, i.e. the result of the outcome depends on the order of summation. Also the sum extends over infinite number of lattice vectors, which means that one has to modify the procedure in order to get an absolute convergent sum and to get it fast converging. The original method of Ewald consisted in introducing a convergence factor e^{-ns}, which makes the sum absolute convergent; then transforming it into different fast converging terms and then putting s in the convergence factor to zero. The final result of the calculation can be easier understood from a physical picture. If every charge in the system is screened by a counter charge of opposite sign, which is smeared out, then the potential of this composite charge distribution becomes short ranged (it is similar in electrolytic solutions, where ionic charges are screened by counter charges - the result is an exponentially decaying function, the Debye potential[74]). In order to compensate for the added charge distribution it has to be subtracted again. The far field of a localized charge distribution is, however, again a Coulomb potential. Therefore this term will be long ranged. There would be nothing gained if one would simply sum up these different terms. The efficiency gain shows up, when one calculates the short range interactions as direct particle-particle contributions in real space, while summing up the long range part of the smeared charge cloud in reciprocal Fourier space. Choosing as the smeared charge distribution a Gaussian charge cloud of half width $1/\alpha$ the corresponding expression for the energy becomes

$$\begin{aligned} \phi(\mathbf{r}_i) = &\sum_{\mathbf{n}}{}^{\dagger} \sum_{j=1}^{N} q_j \frac{\operatorname{erfc}(\alpha\|\mathbf{r}_{ij} + \mathbf{n}L\|)}{\|\mathbf{r}_{ij} + \mathbf{n}L\|} \\ &+ \frac{4\pi}{L^3} \sum_{\mathbf{k}\neq 0} \sum_{j=1}^{N} \frac{q_j}{\|\mathbf{k}\|^2} e^{-\|\mathbf{k}\|^2/4\alpha^2} e^{i\mathbf{k}\mathbf{r}_{ij}} - q_i \frac{2\alpha}{\sqrt{\pi}} \end{aligned} \tag{15}$$

The last term corresponds to a self energy contribution which has to be subtracted, as it is considered in the Fourier part. Eq. (15) is an exact equivalent of Eq. (14), with the difference that it is an absolute converging expression. Therefore nothing would be gained without further approximation. Since the complimentary error function can be approximated for large arguments by a Gaussian function and the k-space parts decreases like a Gaussian, both terms can be approximated by stopping the sums at a certain lattice vector $\mathbf{n}$ and a maximal k-value k_{max}. The choice of parameters depends on the error, $\epsilon = \exp(-p^2)$, which one accepts to tolerate. Setting the error tolerance p and choosing the width of the counter charge distribution, one gets

$$R_c^2 + \frac{\log(R_c)}{\alpha^2} = \frac{1}{\alpha^2}(p^2 - \log(2)) \tag{16}$$

$$k_{max}^2 + 8\alpha^2 \log(k_{max}) = 4\alpha^2 p^2 + \log\left(\frac{4\pi}{L^3}\right) \tag{17}$$

This can be solved iteratively or if one is only interested in an approximate estimate for the error, i.e. neglecting logarithmic terms, one gets

$$R_c = \frac{p}{\alpha} \tag{18}$$

$$k_{max} = 2\alpha p \tag{19}$$

Using this error estimate and furthermore introducing execution times, spent for the real- and reciprocal-space part, it is possible to show that parameters R_c, α and k_{max} can be chosen to get a complexity of $\mathcal{O}(N^{3/2})$ for the Ewald sum[75,76]. In this case, parameters are

$$\frac{R_c}{L} \approx \sqrt{\frac{\pi}{N^{1/3}}} \quad , \quad \alpha L \approx \frac{L k_{max}}{2\pi} = \sqrt{\pi N^{1/3}} \tag{20}$$

Figure 5 shows the contributions of real- and reciprocal parts in Eq. (15), as a function of the spreading parameter α, where an upper limit in both the real- and reciprocal-contributions was applied. In the real-space part usually one restricts the sum to $|\mathbf{n}| = 0$ and applies a spherical cutoff radius, R_c. For fixed values of R_c and k_{max} there is a broad plateau region, where the two terms add up to a constant value. Within this plateau region, a value for α should be chosen. Often it is chosen according to $\alpha = 5/L$. Also shown is the potential energy of a particle, calculated with the Ewald sum. It is well observed that due to the periodicity of the system the potential energy surface is not radial symmetric, which may cause problems for small numbers of particles in the system.

The present form of the Ewald sum gives an exact representation of the potential energy of point like charges in a system with periodic boundary conditions. Sometimes the charge distribution in a molecule is approximated by a point dipole or higher multipole moments. A more general form of the Ewald sum, taking into account arbitrary point multipoles was given in Ref.[77]. The case, where also electronic polarizabilities are considered is given in Ref.[78].

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

$$U = \frac{1}{2} \sum_{i,j=1}^{N} q_i q_j \frac{e^{-\kappa \|\mathbf{r}_{ij}\|}}{\|\mathbf{r}_{ij}\|} \tag{21}$$

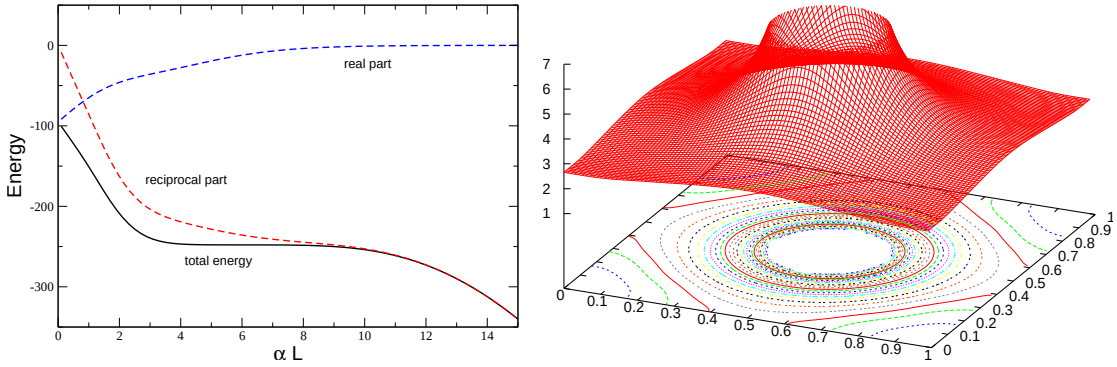


Figure 5: Left: Dependence of the calculated potential on the choice of the scaled inverse width, $\alpha\, L$, of the smeared counter charge distribution. Parameters for this test were $N = 152$, $R_c = 0.5\, L$ and $k_{max} L/2\pi = 6$. Right: Surface plot and contours for the electrostatic potential of a charge, located in the center of the simulation volume. Picture shows the xy-plane for $z = L/2$. Parameters were $R_c = 0.25\, L$, $\alpha\, L = 12.2$ and $k_{max} L/2\pi = 6$.

The parameter κ is the inverse Debye length, which gives a measure of screening strength in the system. If $\kappa < 1/L$ the potential is short ranged and usual cut-off methods may be used. Instead, if $\kappa > 1/L$, or generally if $u(r = L/2)$ is larger than the prescribed uncertainties in the energy, the minimum image convention in combination with truncation methods fails and the potential must be treated in a more rigorous way, which was proposed in Ref.[79], where an extension of the Ewald sum for such Yukawa type potentials was developed.

2.2.2 The Fast Multipole Method

In open geometries there is no lattice summation, but only the sum over all particle pairs in the whole system. The electrostatic energy at a particle's position is therefore simply calculated as

$$\phi(\mathbf{r}_i) = \sum_{j=1}^{N} \frac{q_j}{\|\mathbf{r}_i - \mathbf{r}_j\|} \tag{22}$$

Without further approximation this is always an $\mathcal{O}(N^2)$ algorithm since there are $N(N-1)/2$ interactions to consider in the system (here Newton's third law was taken into account). The idea of a multipole method is to group particles which are far away from a tagged particle together and to consider an effective interaction of a particle with this particle group[80–82]. The physical space is therefore subdivided in a hierarchical way, where the whole system is considered as level 0. Each further level is constructed by dividing the length in each direction by a factor of two. The whole system is therefore subdivided into a hierarchy of boxes where each *parent box* contains eight *children boxes*. This subdivision is performed at maximum until the level, where each particle is located in an individual box. Often it is enough to stop the subdivision already at a lower level.

In the following it is convenient to work in spherical coordinates. The main principle of the method is that the interaction between two particles, located at $\mathbf{r} = r, \theta, \varphi$ and

$\mathbf{a} = (a, \alpha, \beta)$ can be written as a multipole expansion[83]

$$\frac{1}{\|\mathbf{r} - \mathbf{a}\|} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{(l-|m|)!}{(l+|m|)!} \frac{a^l}{r^{l+1}} P_{lm}(\cos\alpha)\, P_{lm}(\cos\theta)\, e^{-im(\beta-\varphi)} \qquad (23)$$

where $P_{lm}(x)$ are associated Legendre polynomials[84]. This expression requires that $a/r < 1$ and this gives a lower limit for the so called *well separated* boxes. This makes it necessary to have at least one box between a tagged box and the zone, where contributions can be expanded into multipoles. Defining the operators

$$O_{lm}(\mathbf{a}) = a^l\, (l-|m|)!\, P_{lm}(\cos\alpha)\, e^{-im\beta} \qquad (24)$$

$$M_{lm}(\mathbf{r}) = \frac{1}{r^{l+1}} \frac{1}{(l+|m|)!}\, P_{lm}(\cos\theta)\, e^{im\varphi} \qquad (25)$$

with which Eq. (23) may simply be rewritten in a more compact way, it is possible to write further three operators, which are needed, in a compact scheme, i.e.

1.) a translation operator, which relates the multipole expansion of a point located at $\mathbf{a}$ to a multipole expansion of a point located at $\mathbf{a} + \mathbf{b}$

$$O_{lm}(\mathbf{a}+\mathbf{b}) = \sum_{j=0}^{l} \sum_{k=-l}^{j} A_{jk}^{lm}(\mathbf{b})\, O_{jk}(\mathbf{a}) \qquad , \qquad A_{jk}^{lm}(\mathbf{b}) = O_{l-j,m-k}(\mathbf{b}) \qquad (26)$$

2.) a transformation operator, which transforms a multipole expansion centered at the origin into a Taylor expansion centered at location $\mathbf{b}$

$$M_{lm}(\mathbf{a}-\mathbf{b}) = \sum_{j=0}^{l} \sum_{k=-l}^{j} B_{jk}^{lm}(\mathbf{b})\, O_{jk}(\mathbf{a}) \qquad , \qquad B_{jk}^{lm}(\mathbf{b}) = M_{l+j,m+k}(\mathbf{b}) \qquad (27)$$

3.) a translation operator, which translates a Taylor expansion of a point $\mathbf{r}$ about the origin into a Taylor expansion of $\mathbf{r}$ about a point $\mathbf{b}$

$$M_{lm}(\mathbf{r}-\mathbf{b}) = \sum_{j=0}^{l} \sum_{k=-l}^{j} C_{jk}^{lm}(\mathbf{b})\, M_{jk}(\mathbf{r}) \qquad , \qquad C_{jk}^{lm}(\mathbf{b}) = A_{jk}^{lm}(\mathbf{b}) \qquad (28)$$

The procedure to calculate interactions between particles is then subdivided into five passes. Figure 6 illustrates four of them. The first pass consists of calculating the multipole expansions in the lowest level boxes (finest subdivision). Using the translation operator $O_{lm}(\mathbf{a}+\mathbf{b})$, the multipole expansions are translated into the center of their parent boxes and summed up. This procedure is repeated then subsequently for each level, until level 2 is reached, from where no further information is passed to a coarser level. In pass 2, using operator $M_{lm}(\mathbf{a}-\mathbf{b})$, multipole expansions are translated into Taylor expansions in a box from well separated boxes, whose parent boxes are nearest neighbor boxes. Well separated means, that for all particles in a given box the multipole expansion in a separated box is valid. Since the applicability of Eq. (23) implies $r > a$, well separateness means on level l that boxes should be separated by a distance 2^{-l}. This also explains, why there is no need to transfer information higher than level 2, since from there on it is not possible to have well separated boxes anymore, i.e. multipole expansions are not valid any more. In

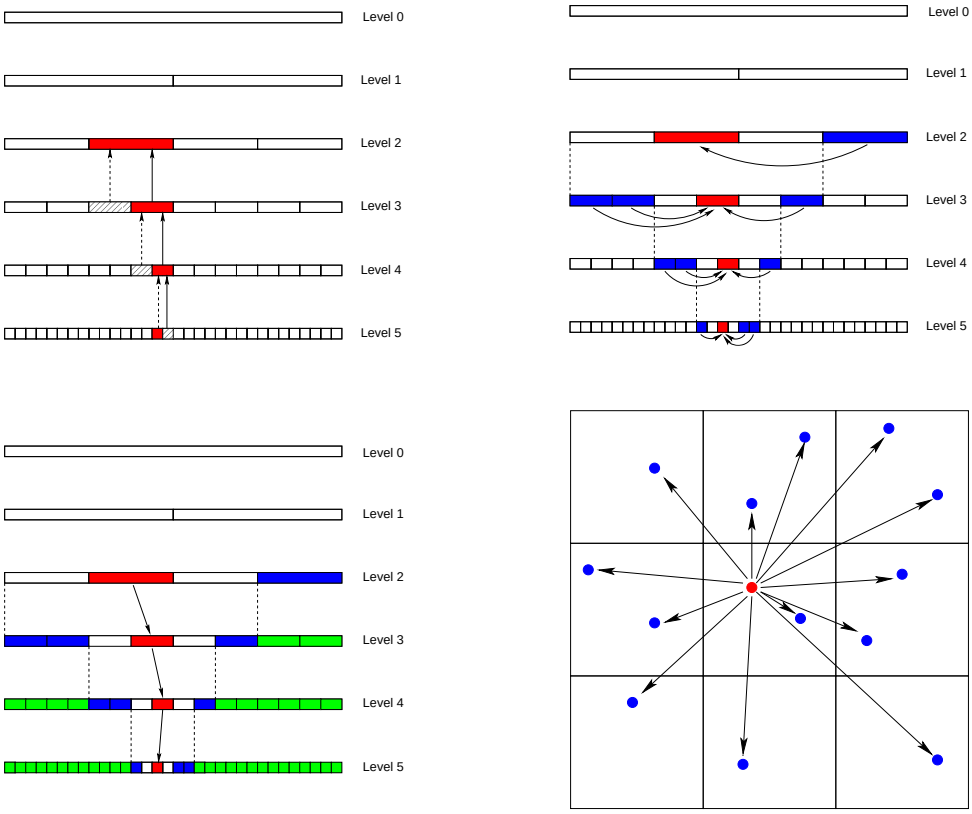


Figure 6: Schematic of different passes in the Fast Multipole Method. Upper left: Pass 1, evaluation of multipole terms in finest subdivision and translating information upwards the tree. Upper right: Pass 2, transforming multipole expansions in well separated boxes into local Taylor expansions. Lower left: Pass 3, transferring multipole expansions downwards the tree, thus collecting information of the whole system, except nearest neighbor boxes. Lower right: Pass 5, direct calculation of particle-particle interactions in local and nearest neighbor boxes.

pass 3, using the operator $M_{lm}(\mathbf{a}-\mathbf{b})$, this information is then translated downwards the tree, so that finally on the finest level all multipole information is known in order to interact individual particles with expansions, originating from all other particles in the system which are located in well separated boxes of the finest level. In pass 4 this interaction between individual particles and multipoles is performed. Finally in pass 5, explicit pair-pair interactions are calculated between particles in a lowest level box and those which are in nearest neighbor boxes, i.e. those boxes which are not called well separated.

It can be shown[65] that each of the steps performed in this algorithm is of order $\mathcal{O}(N)$, making it an optimal method. Also the error made by this method can be controlled rather reliably[68]. A very conservative error estimate is thereby given as[80,65,85]

$$\left|\phi(r)-\frac{q}{\|\mathbf{r}-\mathbf{a}\|}\right| \leq \frac{|q|}{r-a}\left(\frac{a}{r}\right)^{p+1} \tag{29}$$

At the current description the evaluation of multipole terms scales as $\mathcal{O}(l_{max}^4)$, when l_{max} is the largest value of l in the multipole expansion, Eq.(23). A faster version which scales as $\mathcal{O}(l_{max}^3)$ and therefore strongly reducing the prefactor of the overall scheme, was proposed in Ref.[66], where multipoles are evaluated in a rotated coordinate frame, which makes it possible to reduce calculations to Legendre polynomials and not requiring associated Legendre polynomials.

Also to mention is that there are approaches to extend the Fast Multipole Method to periodic systems[86,87].

2.3 Coarse Grain Methods

The force field methods mentioned so far treat molecules on the atomic level, i.e. resolving heavy atoms, in most cases also hydrogens, explicitly. In the case, where flexible molecular bonds, described e.g. by harmonic potentials, are considered the applied time step is of the order of $\delta t \approx 10^{-15}\, secs$. Considering physical phenomena like self assembling of lipid molecules[88,89], protein folding or structure formation in macromolecular systems[90–92], which take place on time scales of microseconds to seconds or even longer, the number of timesteps would exceed the current computational capacities. Although these phenomena all have an underlying microscopic background, the fast dynamics of e.g. hydrogen vibrations are not directly reflected in the overall process. This lead to the idea to either freeze certain degrees of freedom, as it is done for e.g. rigid water models[93–96], or to take several degrees of freedom only into account effectively via a pseudo potential, which reflects the behavior of whole groups of atoms. It is the latter approach which is now known as coarse graining[32,12,97] of molecular potentials and which opens the accessibility of a larger time and length scale. Mapping groups of atoms to one pseudo atom, or interaction site, leads already to an effective increase of the specific volume of the degrees of freedom. Therefore, the same number of degrees of freedom of a coarse grain model, compared with a conventional force field model, would directly lead to larger spatial scale, due to the increase of volume of each degree of freedom. On the other hand, comparing a conventional system before and after coarse graining, the coarse grained system could cover time scales longer by a factor of 100-1000 or even longer compared with a conventional force field all-atom model (the concrete factor certainly depends on the level of coarse graining).

Methodologies for obtaining coarse grain models of a system often start from an atomistic all-atom model, which adequately describes phase diagrams or other physical properties of interest. On a next level, groups of atoms are collected and an effective non-bonded interaction potential may be obtained by calculating potential energy surfaces of these groups and to parametrize these potentials to obtain analytical descriptions. Therefore, distribution functions of small atomic groups are taken into account (at least implicitly) which in general depend on the thermodynamic state point. For bonded potentials between groups of atoms, a normal mode analysis may be performed in order to get the most important contributions to vibrational-, bending- or torsional-modes.

In principle, one is interested in reducing the number of degrees of freedom by separating the problem space into coordinates which are *important* and those which are *unimportant*. Formally, this may be expressed through a set of coordinates $\{\mathbf{r}\} \in \mathbb{R}^{n_i}$ and $\{\tilde{\mathbf{r}}\} \in \mathbb{R}^{n_u}$, where n_i and n_u are the number of degrees of important and unimportant

degrees of freedom, respectively. Consequently, the system Hamiltonian may be written as $H = H(r_1, \ldots, r_{n_i}, \tilde{r}_1, \ldots, \tilde{r}_{n_u})$. From these considerations one may define a *reduced* partition function, which results from integrating out all unimportant degrees of freedom

$$Z = \int dr_1 \ldots dr_{n_i} d\tilde{r}_1 \ldots d\tilde{r}_{n_i} \exp\left\{-\beta H(r_1, \ldots, r_{n_i}, \tilde{r}_1, \ldots, \tilde{r}_{n_u})\right\} \tag{30}$$

$$= \int dr_1 \ldots dr_{n_i} \exp\left\{-\beta H^{CG}(r_1, \ldots, r_{n_i})\right\} \tag{31}$$

where a coarse grain Hamiltonian has been defined

$$H^{CG}(r_1, \ldots, r_{n_i}) = -\log \int d\tilde{r}_1 \ldots d\tilde{r}_{n_u} \exp\left\{-\beta H(r_1, \ldots, r_{n_i}, \tilde{r}_1, \ldots, \tilde{r}_{n_u})\right\} \tag{32}$$

which corresponds to the potential of mean force and which is the free energy of the non-important degrees of freedom. Since the Hamiltonian describes only a subset of degrees of freedom, thermodynamic properties, derived from this Hamiltonian will be different than obtained from the full Hamiltonian description (e.g. pressure will correspond to the osmotic pressure and not to the thermodynamic pressure). This has to be taken into account when simulating in different ensembles or if experimental thermodynamic properties should be reproduced by simulation.

The coarse grained Hamiltonian is still a multi-body description of the system, which is hard to obtain numerically. Therefore, it is often approximated by a pair-potential, which is considered to contribute the most important terms

$$H^{CG}(r_1, \ldots, r_{n_i}) \approx \sum_{i>j} V_{ij}(r_{ij}) \qquad , \qquad r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\| \tag{33}$$

According to the uniqueness theorem of Henderson[98], in a liquid where particles interact only through pair interactions, the pair distribution function $g(r)$ determines up to a constant uniquely the pair interaction potential V_{ij}. Therefore, V_{ij} may be obtained pointwise by reverting the radial pair distribution function[99–101], e.g. by reverse Monte Carlo techniques[102] or dynamic iterative refinement[103]. This approach directly confirms what was stated in Sec. 1 about the limited applicability of coarse grained potentials. It is clear that for different temperatures, pressures or densities the radial distribution functions of e.g. cation-cation, cation-anion and anion-anion distributions in electrolytic solutions will be different. If one wants to simulate ions in an effective medium (continuum solvent), the potential, which is applied in the simulation will depend on the thermodynamic state point and therefore has to be re-parametrized for every different state point.

3 The Integrator

The propagation of a classical particle system can be described by the temporal evolution of the phase space variables $(\mathbf{p}, \mathbf{q})$, where the phase space $\Gamma(\mathbf{p}, \mathbf{q}) \in \mathbb{R}^{6N}$ contains all possible combinations of momenta and coordinates of the system. The exact time evolution of the system is thereby given by a flow map

$$\Phi_{\delta t, \mathcal{H}} : \mathbb{R}^{6N} \to \mathbb{R}^{6N} \tag{34}$$

which means

$$\Phi_{\delta t,\mathcal{H}}(\mathbf{p}(t),\mathbf{q}(t)) = (\mathbf{p}(t)+\delta\mathbf{p},\mathbf{q}(t)+\delta\mathbf{q}) \tag{35}$$

where

$$\mathbf{p}+\delta\mathbf{p} = \mathbf{p}(t+\delta t) \qquad , \qquad \mathbf{q}+\delta\mathbf{q} = \mathbf{q}(t+\delta t) \tag{36}$$

For a nonlinear many-body system, the equations of motion cannot be integrated exactly and one has to rely on numerical integration of a certain order. Propagating the coordinates by a constant step size h, a number of different finite difference schemes may be used for the integration. But there are a number of requirements, which have to be fulfilled in order to be useful for molecular dynamics simulations. An integrator, suitable for many-body simulations should fulfill the following requirements:

- Accuracy, i.e. the solution of an analytically solvable test problem should be as close as possible to the numerical one.
- Stability, i.e. very long simulation runs should produce physically relevant trajectories, which are not governed by numerical artifacts
- Conservativity, there should be no drift or divergence in conserved quantities, like energy, momentum or angular momentum
- Reversibility, i.e. it should have the same temporal structure as the underlying equations
- Effectiveness, i.e. it should allow for large time steps without entering instability and should require a minimum of force evaluations, which usually need about 95 % of CPU time per time step
- Symplecticity, i.e. the geometrical structure of the phase space should be conserved

It is obvious that the numerical flow, $\phi_{\delta t,\mathcal{H}}$, of a finite difference scheme will not be fully equivalent to $\Phi_{\delta t,\mathcal{H}}$, but the system dynamics will be described correctly if the items above will be fulfilled.

In the following the mentioned points will be discussed and a number of different integrators will be compared.

3.1 Basic Methods

The most simple integration scheme is the Euler method, which may be constructed by a first order difference approximation to the time derivative of the phase space variables

$$\mathbf{p}_{n+1} = \mathbf{p}_n - \delta t\frac{\partial}{\partial\mathbf{q}}\mathcal{H}(\mathbf{p}_n,\mathbf{q}_n) \tag{37}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \delta t\frac{\partial}{\partial\mathbf{p}}\mathcal{H}(\mathbf{p}_n,\mathbf{q}_n) \tag{38}$$

where δt is the step size of integration. This is equivalent to a Taylor expansion which is truncated after the first derivative. Therefore, it is obvious that it is of first order. Knowing

all variables at step n, this scheme has all relevant information to perform the integration. Since only information from one time step is required to do the integration, this scheme is called the one-step explicit Euler scheme. The basic scheme, Eqs. (37,38) may also be written in different forms.

The implicit Euler method

$$\mathbf{p}_{n+1} = \mathbf{p}_n - \delta t \frac{\partial}{\partial \mathbf{q}} \mathcal{H}(\mathbf{p}_{n+1}, \mathbf{q}_{n+1}) \tag{39}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \delta t \frac{\partial}{\partial \mathbf{p}} \mathcal{H}(\mathbf{p}_{n+1}, \mathbf{q}_{n+1}) \tag{40}$$

can only be solved iteratively, since the derivative on the right-hand-side (*rhs*) is evaluated at the coordinate positions on the left-hand-side (*lhs*).

An example for a so called partitioned Runge-Kutta method is the *velocity implicit method*

$$\mathbf{p}_{n+1} = \mathbf{p}_n - \delta t \frac{\partial}{\partial \mathbf{q}} \mathcal{H}(\mathbf{p}_{n+1}, \mathbf{q}_n) \tag{41}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \delta t \frac{\partial}{\partial \mathbf{p}} \mathcal{H}(\mathbf{p}_{n+1}, \mathbf{q}_n) \tag{42}$$

Since the Hamiltonian usually splits into kinetic $\mathcal{K}$ and potential $\mathcal{U}$ parts, which only depend on one phase space variable, i.e.

$$\mathcal{H}(\mathbf{p}, \mathbf{q}) = \frac{1}{2} \mathbf{p}^T \, \mathbf{M}^{-1} \, \mathbf{p} + \mathcal{U}(\mathbf{q}) \tag{43}$$

where $\mathbf{M}^{-1}$ is the inverse of the diagonal mass matrix, this scheme may also be written as

$$\mathbf{p}_{n+1} = \mathbf{p}_n - \delta t \frac{\partial}{\partial \mathbf{q}} \mathcal{U}(\mathbf{q}_n) \tag{44}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \frac{\delta t}{m} \mathbf{p}_{n+1} \tag{45}$$

showing that it is not necessary to solve it iteratively.

Obviously this may be written as a *position implicit method*

$$\mathbf{p}_{n+1} = \mathbf{p}_n - \delta t \frac{\partial}{\partial \mathbf{q}} \mathcal{U}(\mathbf{q}_{n+1}) \tag{46}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \frac{\delta t}{m} \, \mathbf{p}_n \tag{47}$$

Applying first Eq. (47) and afterwards Eq. (46) also this variant does not require an iterative procedure.

All of these schemes are first order accurate but have different properties, as will be shown below. Before discussing these schemes it will be interesting to show a higher order

scheme, which is also based on a Taylor expansion. First write down expansions

$$\mathbf{q}(t+\delta t) = \mathbf{q}(t) + \delta t\,\dot{\mathbf{q}}(t) + \frac{1}{2}\,\delta t^2\,\ddot{\mathbf{q}}(t) + O(\delta t^3) \tag{48}$$

$$= \mathbf{q}(t) + \frac{\delta t}{m}\,\mathbf{p}(t) + \frac{1}{2m}\,\delta t^2\,\dot{\mathbf{p}}(t) + O(\delta t^3) \tag{49}$$

$$\mathbf{p}(t+\delta t) = \mathbf{p}(t) + \delta t\,\dot{\mathbf{p}}(t) + \frac{1}{2}\,\delta t^2\,\ddot{\mathbf{p}}(t) + O(\delta t^3) \tag{50}$$

$$= \mathbf{p}(t) + \frac{\delta t}{2}\,(\dot{\mathbf{p}}(t) + \dot{\mathbf{p}}(t+\delta t)) + O(\delta t^3) \tag{51}$$

where in Eq. (49), the relation $\dot{\mathbf{q}} = \mathbf{p}/m$ was used and in Eq. (51) a first order Taylor expansion for $\dot{\mathbf{p}}$ was inserted. From these expansions a simple second order, one-step splitting scheme may be written as

$$\mathbf{p}_{n+1/2} = \mathbf{p}_n + \frac{\delta t}{2}\,\mathbf{F}(\mathbf{q}_n) \tag{52}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \frac{\delta t}{m}\,\mathbf{p}_{n+1/2} \tag{53}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_{n+1/2} + \frac{\delta t}{2}\,\mathbf{F}(\mathbf{q}_{n+1}) \tag{54}$$

where the relation $\dot{\mathbf{p}} = -\partial\mathcal{H}/\partial\mathbf{q} = \mathbf{F}$ was used. This scheme is called the *Velocity Verlet* scheme. In a pictorial way it is sometimes described as half-kick, drift, half-kick, since the first step consists in applying forces for half a time step, second step consists in free flight of a particle with momentum $\mathbf{p}_{n+1/2}$ and the last step applies again a force for half a time step. In practice, forces only need to be evaluated once in each time step. After having calculated the new positions, $\mathbf{q}_{n+1}$, forces are calculated for the last integration step. They are, however, stored to be used in the first integration step as *old* forces in the next time step of the simulation.

This algorithm comes also in another flavor, called the *Position Verlet* scheme. It can be expressed as

$$\mathbf{q}_{n+1/2} = \mathbf{q}_n + \frac{\delta t}{2m}\,\mathbf{p}_n \tag{55}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \delta t\,\mathbf{F}(\mathbf{q}_{n+1/2}) \tag{56}$$

$$\mathbf{q}_{n+1/2} = \mathbf{q}_{n+1/2} + \frac{\delta t}{2m}\,\mathbf{p}_{n+1} \tag{57}$$

In analogy to the description above this is sometimes described as half-drift, kick, half-drift. Using the relation $\mathbf{p} = \dot{\mathbf{q}}/m$ and expressing this as a first order expansion, it is obvious that $\mathbf{F}(\mathbf{q}_{n+1/2}) = \mathbf{F}((\mathbf{q}_n + \mathbf{q}_{n+1})/2)$ which corresponds to an implicit midpoint rule.

3.2 Operator Splitting Methods

A more rigorous derivation, which in addition leads to the possibility of splitting the propagator of the phase space trajectory into several time scales, is based on the phase space

description of a classical system. The time evolution of a point in the $6N$ dimensional phase space is given by the Liouville equation

$$\mathbf{\Gamma}(t) = e^{i\mathcal{L}t}\,\mathbf{\Gamma}(0) \tag{58}$$

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

$$i\mathcal{L} = \{\ldots, \mathcal{H}\} = \sum_{j=1}^{N} \left(\frac{\partial \mathbf{q}_j}{\partial t}\frac{\partial}{\partial \mathbf{q}_j} + \frac{\partial \mathbf{p}_j}{\partial t}\frac{\partial}{\partial \mathbf{p}_j} \right) \tag{59}$$

In order to construct a discrete timestep integrator, the Liouville operator is split into two parts, $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$, and a Trotter expansion[104] is performed

$$e^{i\mathcal{L}\delta t} = e^{i(\mathcal{L}_1+\mathcal{L}_2)\delta t} \tag{60}$$

$$= e^{i\mathcal{L}_1\delta t/2} e^{i\mathcal{L}_2\delta t} e^{i\mathcal{L}_1\delta t/2} + \mathcal{O}(\delta t^3) \tag{61}$$

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

$$i\mathcal{L}_1 = \sum_{j=1}^{N} \mathbf{F}_j \frac{\partial}{\partial \mathbf{p}_j} \tag{62}$$

$$i\mathcal{L}_2 = \sum_{j=1}^{N} \mathbf{v}_j \frac{\partial}{\partial \mathbf{r}_j} \tag{63}$$

Applying Eq.60 to the phase space vector $\mathbf{\Gamma}$ and using the property $e^{a\partial/\partial x} f(x) = f(x+a)$ for any function f, where a is independent of x, gives

$$\mathbf{v}_i(t + \delta t/2) = \mathbf{v}(t) + \frac{\mathbf{F}_i(t)}{m_i}\frac{\delta t}{2} \tag{64}$$

$$\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t + \delta t/2)\delta t \tag{65}$$

$$\mathbf{v}_i(t + \delta t) = \mathbf{v}_i(t + \delta t/2) + \frac{\mathbf{F}_i(t + \delta t)}{m_i}\frac{\delta t}{2} \tag{66}$$

which is the velocity Verlet algorithm, Eqs. 52-54. In the same spirit, another algorithm may be derived by simply changing the definitions for $\mathcal{L}_1 \to \mathcal{L}_2$ and $\mathcal{L}_2 \to \mathcal{L}_1$. This gives the so called *position Verlet algorithm*

$$\mathbf{r}_i(t + \delta t/2) = \mathbf{r}_i(t) + \mathbf{v}(t)\frac{\delta t}{2} \tag{67}$$

$$\mathbf{v}_i(t + \delta t) = \mathbf{v}(t) + \frac{\mathbf{F}_i(t + \delta t/2)}{m_i} \tag{68}$$

$$\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t + \delta t/2) + (\mathbf{v}(t) + \mathbf{v}_i(t + \delta t))\frac{\delta t}{2} \tag{69}$$

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

for the velocity Verlet and correspondingly *half-drift/kick/half-drift* for the position Verlet algorithm.

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

$$\begin{pmatrix} f_x & f_p \\ g_x & g_p \end{pmatrix} \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \begin{pmatrix} f_x & f_p \\ g_x & g_p \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \tag{70}$$

Any method which is based on the splitting of the Hamiltonian, is symplectic. This does not yet, however, guarantee that the method is also time reversible, which may be also be considered as a strong requirement for the integrator. This property is guaranteed by symmetric methods, which also provide a better numerical stability[105]. Methods, which try to enhance the accuracy by taking into account the particles' history (multi-step methods) tend to be incompatible with symplecticness[106,107], which makes symplectic schemes attractive from the point of view of data storage requirements. Another strong argument for symplectic schemes is the so called *backward error analysis*[108–110]. This means that the trajectory produced by a discrete integration scheme, may be expressed as the solution of a perturbed ordinary differential equation whose *rhs* can formally be expressed as a power series in δt. It could be shown that the system, described by the ordinary differential equation is Hamiltonian, if the integrator is symplectic[111,112]. In general, the power series in δt diverges. However, if the series is truncated, the trajectory will differ only as $\mathcal{O}(\delta t^p)$ of the trajectory, generated by the symplectic integrator on timescales $\mathcal{O}(1/\delta t)$[113].

3.3 Multiple Time Step Methods

It was already mentioned that the rigorous approach of the decomposition of the Liouville operator offers the opportunity for a decomposition of time scales in the system. Supposing that there are different time scales present in the system, e.g. fast intramolecular vibrations and slow domain motions of molecules, then the factorization of Eq.60 may be written in a more general way

$$e^{i\mathcal{L}\Delta t} = e^{i\mathcal{L}_1^{(s)}\Delta t/2} e^{i\mathcal{L}_1^{(f)}\Delta t/2} e^{i\mathcal{L}_2\delta t} e^{i\mathcal{L}_1^{(f)}\Delta t/2} e^{i\mathcal{L}_1^{(s)}\Delta t/2} \tag{71}$$

$$= e^{i\mathcal{L}_1^{(s)}\Delta t/2} \left\{ e^{i\mathcal{L}_1^{(f)}\delta t/2} e^{i\mathcal{L}_2\delta t} e^{i\mathcal{L}_1^{(f)}\delta t/2} \right\}^p e^{i\mathcal{L}_1^{(s)}\Delta t/2} \tag{72}$$

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

$$i\mathcal{L}_1^{(s)} = \mathbf{F}_i^{(s)} \frac{\partial}{\partial \mathbf{p}_i} \quad , \quad i\mathcal{L}_1^{(f)} = \mathbf{F}_i^{(f)} \frac{\partial}{\partial \mathbf{p}_i} \quad , \quad i\mathcal{L}_2 = \mathbf{v}_i \frac{\partial}{\partial \mathbf{q}_i} \tag{73}$$

where the superscript (s) and (f) mean slow and fast contributions to the forces. The idea behind this decomposition is simply to take into account contributions from slowly varying components only every p'th timestep with a large time interval. Therefore, the force computation may be considerably speeded up in the the $p-1$ intermediate force computation steps. In general, the scheme may be extended to account for more time scales. Examples for this may be found in Refs.[114–116]. One obvious problem, however, is to separate the timescales in a proper way. The scheme of Eq.72 is *exact* if the time

scales decouple completely. This, however, is very rarely found and most often timescales are coupled due to nonlinear effects. Nevertheless, for the case where Δt is not very much larger than δt ($p \approx 10$), the separation may be often justified and lead to stable results. Another criteria for the separation is to distinguish between long range and short range contributions to the force. Since the magnitude and the fluctuation frequency is very much larger for the short range contributions this separation makes sense for speeding up computations including long range interactions[117].

The method has, however, its limitations[118,119]. As described, a particle gets every n'th timestep a *kick* due to the slow components. It was reported in literature that this may excite a system's resonance which will lead to strong artifacts or even instabilities[120,121]. Recently different schemes were proposed to overcome these resonances by keeping the property of symplecticness[122–128].

3.4 Stability

Performing simulations of stable many-body systems for long times should produce configurations which are in thermal equilibrium. This means that system properties, e.g. pressure, internal energy, temperature etc. are fluctuating around constant values. To measure these equilibrium properties it should not be relevant where to put the time origin from where configurations are considered to calculate average quantities. This requires that the integrator should propagate phase space variables in such a way that small fluctuations do not lead to a diverging behavior of a system property. This is a kind of minimal requirement in order to simulate any physical system without a domination of numerical artifacts. It is clear, however, that any integration scheme will have its own stability range depending on the step size δt. This is a kind of sampling criterion, i.e. if the step size is too large, in order to resolve details of the energy landscape, an integration scheme may end in instability.

For linear systems it is straight forward to analyze the stability range of a given numerical scheme. Consider e.g. the harmonic oscillator, for which the equations of motion may be written as $\dot{q}(t) = p(t)$ and $\dot{p}(t) = -\omega^2 q(t)$, where ω is the vibrational frequency and it is assumed that it oscillates around the origin. The exact solution of this problem may be written as

$$\begin{pmatrix} \omega\, q(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} \cos\omega t & \sin\omega t \\ -\sin\omega t & \cos\omega t \end{pmatrix} \begin{pmatrix} \omega\, q(0) \\ p(0) \end{pmatrix} \tag{74}$$

For a numerical integrator the stepwise solution may be written as

$$\begin{pmatrix} \omega\, q_{n+1} \\ p_{n+1} \end{pmatrix} = \mathbf{M}(\delta t) \begin{pmatrix} \omega\, q_n \\ p_n \end{pmatrix} \tag{75}$$

where $\mathbf{M}(\delta t)$ is a propagator matrix. It is obvious that any stable numerical scheme requires eigenvalues $|\lambda(\mathbf{M})| \leq 1$. For $|\lambda| > 1$ the scheme will be unstable and divergent, for $|\lambda| < 1$ it will be stable but will exhibit friction, i.e. will loose energy. Therefore, in view of the conservativity of the scheme, it will be required that $|\lambda(\mathbf{M})| = 1$.

As an example the propagator matrices for the Implicit Euler (IE) and Position Verlet (PV) algorithms are calculated as

$$\mathbf{M}_{IE}(\delta t) = \frac{1}{1+\omega^2\delta t^2} \begin{pmatrix} 1 & \omega\delta t \\ -\omega\delta t & 1 \end{pmatrix} \tag{76}$$

$$\mathbf{M}_{PV}(\delta t) = \begin{pmatrix} 1 - \frac{1}{2}\omega^2\delta t^2 & \omega\delta t\left(1 - \frac{1}{4}\omega^2\delta t^2\right) \\ -\omega\delta t & 1 - \frac{1}{2}\omega^2\delta t^2 \end{pmatrix} \tag{77}$$

It is then straight forward to calculate the eigenvalues as roots of the characteristic polynomials. The eigenvalues are then calculated as

$$\lambda_{EE} = 1 \pm i\omega\delta t \tag{78}$$

$$\lambda_{IE} = \frac{1}{1+\omega^2\delta t^2}(1 \pm i\omega\delta t) \tag{79}$$

$$\lambda_{PV} = \lambda_{VV} = \lambda_{VIE} = \lambda_{PIE} = 1 - \frac{1}{2}\omega^2\delta t^2\left(1 \pm \sqrt{1 - \frac{4}{\omega^2\delta t^2}}\right) \tag{80}$$

This shows that the absolute values for the Explicit Euler (EE) and the Implicit Euler methods never equals one for $\delta t \neq 0$, i.e. both methods do not produce stable trajectories. This is different for the Position Verlet, the Velocity Verlet (VV), the Position Implicit Euler (PIE) and the Velocity Implicit Euler (VIE), which all have the same eigenvalues. It is found that the range of stability for all of them is in the range $\omega^2\delta t^2 < 2$. For larger values of δt the absolute values of the eigenvalues bifurcates, getting larger and smaller values than one. In Figure 7 the absolute values are shown for all methods and in in Figure 8 the imaginary versus real parts of λ are shown. For EE it is clear that the imaginary part diverges linearly with increase of δt. The eigenvalues of the stable methods are located on a circle until $\omega^2\delta t^2 = 2$. From there one branch diverges to $-\infty$, while the other decreases to zero.

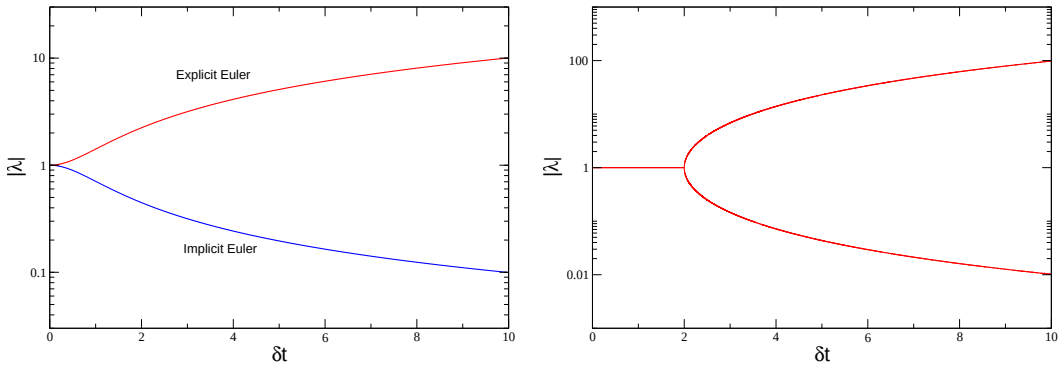


Figure 7: Absolute value of the eigenvalues λ as function of the time step δt. Left: Explicit and implicit Euler method. Right: Velocity and Position Verlet as well as Velocity Implicit and Position implicit Euler method. All methods have the eigenvalues.

As a numerical example the phase space trajectories of the harmonic oscillator for $\omega = 1$ are shown for the different methods in Figure 9. For the stable methods, results

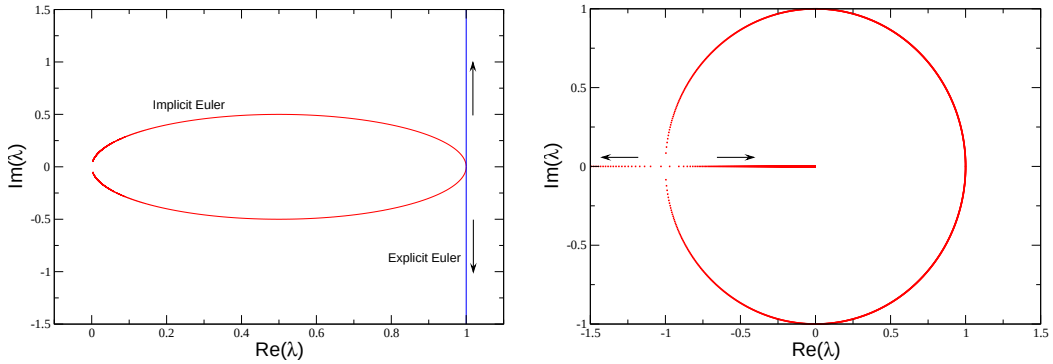


Figure 8: Imaginary versus real part of eigenvalues λ of the propagator matrices. Left: Implicit and Explicit Euler. Right: Velocity and Position Verlet as well as Velocity Implicit and Position implicit Euler method.

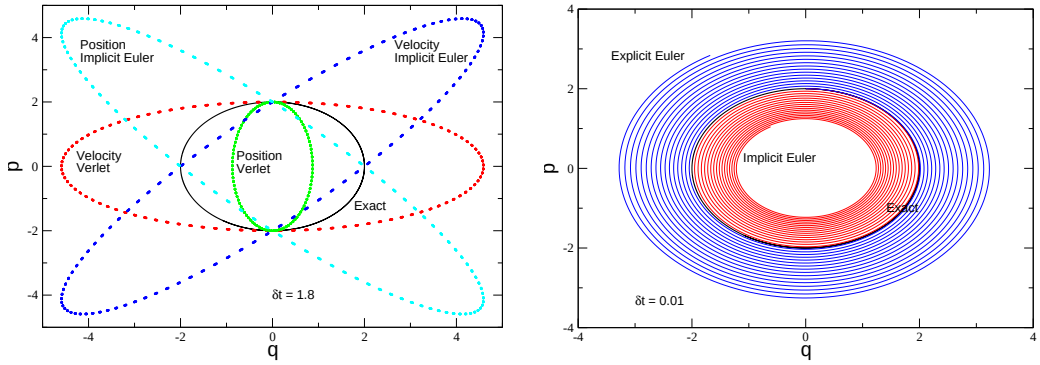


Figure 9: Phase space trajectories for the one-dimensional harmonic oscillator, integrated with the Velocity Implicit Euler, Position Implicit Euler, Velocity Verlet, Position Verlet and integration step size of $\delta t = 1.8$ (left) and the Implicit Euler and Explicit Euler and step size $\delta t = 0.01$ (right).

for a time step close to instability is shown. All different methods produce closed, stable orbits, but it is seen on the other hand that they strongly deviate from the exact solution, which is shown for reference. This demonstrates that stability is a necessary, but only a weak criterion for correct results. Numerically correct results are only obtained for much smaller time steps in the range of $\delta t \approx 0.01$. Also shown are the results for EE and IE. Here a very much smaller time step, $\delta t = 0.01$ is chosen. It is seen that the phase space trajectory of EE spirals out while the one of IE spirals in with time, showing the instable or evanescent character of the methods.

Another issue related to stability is the effect of a trajectory perturbation. If initial conditions are slightly perturbed, will a good integrator keep this trajectory close to the reference trajectory? The answer is No and it is even found that the result is not that strong dependent on the integrator. Even for integrators of high order, trajectories will not stay close to each other. The time evolution of the disturbance may be studied similar to the system trajectory. Consider the time evolution for $\Gamma + \delta\Gamma$, where $\Gamma = (\mathbf{p}, \mathbf{q})$ and

$\delta\Gamma = (\delta\mathbf{p}, \delta\mathbf{q})$ is a small disturbance. Then

$$\frac{d\Gamma}{dt} = \nabla_\Gamma \mathcal{H}(\Gamma) \tag{81}$$

Similarly one can write for small $\delta\Gamma$

$$\frac{d}{dt}(\Gamma + \delta\Gamma) = \nabla_\Gamma \mathcal{H}(\Gamma + \delta\Gamma) \tag{82}$$

$$= \nabla_\Gamma \mathcal{H}(\Gamma) + \nabla_\Gamma(\nabla_\Gamma \mathcal{H}(\Gamma))\delta\Gamma \tag{83}$$

where the second line is a truncated Taylor series. Comparing terms one simply gets as equation of motion for a perturbation

$$\frac{d\delta\Gamma}{dt} = \nabla_\Gamma^2 \mathcal{H}(\Gamma)\delta\Gamma \tag{84}$$

It is found that the disturbance develops exponentially, with a characteristic, system dependent exponent, which is the Ljapunov exponent[129,130].

Now consider the following situation where identical starting configurations are taken for two simulations. They will be carried out by different yet exact algorithms, therefore leading formally to the same result. Nevertheless it may happen that different orders of floating-point operations are used in both algorithms. Due to round off errors, floating-point arithmetic is not necessarily associative, i.e. in general

$$a \hat{\circ} (b \hat{\circ} c) \neq (a \hat{\circ} b) \hat{\circ} c \tag{85}$$

where $\hat{\circ}$ is a floating-point machine operation $(+, -, /, *)$. Therefore, both simulations will be different by round off errors. According to the above discussion, this may be considered as the slightest disturbance of a system trajectory, $\delta\Gamma_{\min}$, and the question is, what effect such a round off error will have. A different method to study difference in system trajectories is the calculation of the difference

$$\gamma_x(t) = \frac{1}{3N} \sum_{i=1}^{N} \sum_{\alpha=x,y,z} (x(t) - \tilde{x}(t))^2 \tag{86}$$

where N is the number of particles, $x(t)$ a certain property, e.g. the coordinates or momenta, and $\tilde{x}$ the same property of a disturbed trajectory. In Figure 10 results are shown for a system of Lennard-Jones particles, where the disturbance was induced by reversing the order of summation in the force routine, thereby provoking round off errors in the first time step. Shown are results for the coordinates, the velocities and the forces and it is seen that all quantities diverge exponentially from machine accuracy up to a certain behavior at long times, which is shown in the inset. To understand the long time behavior, $\gamma_x(t)$ can be written as average property

$$\gamma_x(t) = \langle (x(t) - x(0) - \tilde{x}(t) + x(0))^2 \rangle \tag{87}$$

$$= \langle |x(t) - x(0)|^2 \rangle + \langle |\tilde{x}(t) - x(0)|^2 \rangle \tag{88}$$

$$-2\langle x(t)\tilde{x}(t) \rangle + 2\langle x(0)\tilde{x}(t) \rangle + 2\langle x(t)x(0) \rangle - 2\langle x(0)^2 \rangle$$

In the second equation the first two terms are mean square displacements of x in the two systems (note that $\tilde{x}(0) = x(0)$ since the same starting configurations are used), the next

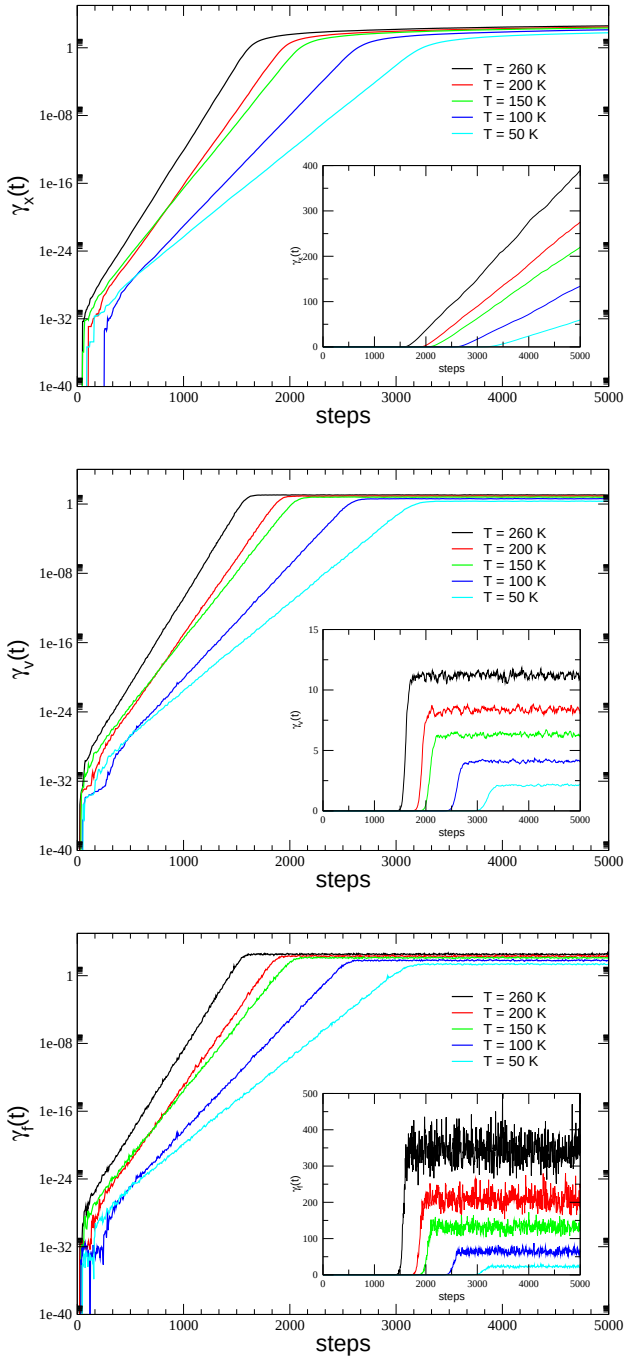


Figure 10: Divergent behavior of trajectories due to round off errors, induced by different summation order in the force routine. From top to bottom: coordinates, velocities, forces. The insets show on a linear scale the long time behavior of the trajectory differences, i.e. when the two systems get uncorrelated.

term is a cross correlation between the systems. This will vanish if the systems become independent of each other. The next two systems consist of auto-correlation functions of x in each system. For long times they will also decrease to zero. Finally, the last term gives a constant offset which does not depend on time. Therefore the long time behavior will be governed for coordinates, momenta and forces by

$$\lim_{t\to\infty} \gamma_q(t) = 2\langle|\mathbf{q}(t) - \mathbf{q}(0)|^2\rangle = 12Dt \tag{89}$$

$$\lim_{t\to\infty} \gamma_p(t) = 2\langle\mathbf{p}(t)^2\rangle = mk_BT \tag{90}$$

$$\lim_{t\to\infty} \gamma_f(t) = 2\langle\mathbf{F}(t)^2\rangle = 2(\nabla\mathcal{W})^2 \tag{91}$$

where D is the diffusion coefficient, T the temperature and $\mathcal{W}$ the potential of mean force.

That the divergent behavior of neighbored trajectories is a system dependent property is shown in Figure 10 where results for Lennard-Jones systems at different temperatures are shown.

In conclusion, the individual trajectories of a physical complex system will end up at different places in phase space when introducing round off errors or small perturbations. Round off errors cannot be avoided with simple floating-point arithmetic (only discrete calculations are able to avoid round off errors; but then the physical problem is transformed into a different space). Since one cannot say anything about a *true* summation order, the location in phase space cannot have an absolute meaning. Therefore, the solution to come out of this dilemma is to interpret the phase space location as a *possible* and *allowed* realization of the system, which makes it necessary, however, to average over a lot of possible realizations.

3.5 Accuracy

For an integrator of order $p \geq 1$, the local error may be written as an upper bound[8]

$$\|\Phi_{\delta t,\mathcal{H}} - \phi_{\delta t}\| \leq M\delta t^{p+1} \tag{92}$$

where $M > 0$ is a constant, $\Phi_{\delta t,\mathcal{H}}$ is the exact and $\phi_{\delta t}$ the numerical flow of the system. The global error, i.e. the accumulated error for larger times, is thereby bound for stable methods by[8]

$$\|\Gamma(t_n) - \Gamma_n\| \leq K\left(e^{Lt_n} - 1\right)\delta t^p \qquad , \qquad t_n = n\delta t \tag{93}$$

where $K > 0$ is a constant, $L > 0$ the Lipschitz constant, $\Gamma(t_n) = (\mathbf{p}(t_n), \mathbf{q}(t_n))$ the exact and $\Gamma_n = (\mathbf{p}_n, \mathbf{q}_n)$ the numerically computed trajectory at time t_n. This estimate gives of course not too much information for $Lt_n 1$ unless δt is chosen very small. Nevertheless, qualitatively this estimate shows a similar exponential divergent behavior of numerical and exact solution for a numerical scheme, as was observed in Section 3.4.

A different approach to the error behavior of a numerical scheme is backward error analysis, first mentioned in Ref.[131] in the context of differential equations. The idea is to consider the numerical solution of a given scheme as the exact solution of a modified equation. The comparison of the original and the modified equation then gives qualitative insight into the long time behavior of a given scheme.

It is assumed that the numerical scheme can be expressed as a series of the form

$$\phi_{\delta t}(\Gamma_n) = \Gamma_n + \delta t f(\Gamma) + \delta t^2 g_2(\Gamma) + \delta t^3 g_3(\Gamma) \pm \ldots \tag{94}$$

where the g_i are known coefficients and for consistency of the differential equation it must hold

$$f(\Gamma) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \nabla_p \\ \nabla_q \end{pmatrix} \mathcal{H}(\mathbf{p}, \mathbf{q}) \tag{95}$$

On the other hand it is assumed that there exists a modified differential equation of the form

$$\frac{d}{dt}\tilde{\Gamma} = f(\tilde{\Gamma}) + \delta t f_2(\tilde{\Gamma}) + \delta t^2 f_3(\tilde{\Gamma}) + \dots \tag{96}$$

where $\tilde{\Gamma}$ will be equivalent to the numerically obtained solution. In order to construct the modified equation, the solution of Eq. (96) is Taylor expanded, i.e.

$$\begin{aligned} \tilde{\Gamma}(t+\delta t) = \tilde{\Gamma}(t) &+ \delta t \left(f(\tilde{\Gamma}) + \delta t f_2(\tilde{\Gamma}) + \delta t^2 f_3(\tilde{\Gamma}) + \dots \right) \\ &+ \frac{\delta t^2}{2!} \left(f'(\tilde{\Gamma}) + \delta t f_2'(\tilde{\Gamma}) + \dots \right) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \left(f(\tilde{\Gamma}) + \delta t f_2(\tilde{\Gamma}) + \dots \right) \\ &+ \frac{\delta t^3}{3!} \Bigg\{ \left(f''(\tilde{\Gamma}) + \delta t f_2''(\tilde{\Gamma}) + \dots \right) \left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \left(f(\tilde{\Gamma}) + \delta t f_2(\tilde{\Gamma}) + \dots \right) \right)^2 \\ &\quad + \left(f'(\tilde{\Gamma}) + \delta t f_2'(\tilde{\Gamma}) + \dots \right) \left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \left(f'(\tilde{\Gamma}) + \delta t f_2'(\tilde{\Gamma}) + \dots \right) \right) \\ &\quad \times \left(f(\tilde{\Gamma}) + \delta t f_2(\tilde{\Gamma}) + \dots \right) \Bigg\} \\ &+ \dots \end{aligned} \tag{97}$$

The procedure to construct the unknown functions f_i proceeds in analogy to perturbation theory, i.e. coefficients with same powers of δt are collected which leads to a recursive scheme to solve for all unknowns.

To give an example the Lennard-Jones oscillator is considered, i.e. a particle performing stable motions in negative part of a Lennard-Jones potential. As was observed already for the harmonic oscillator, the Explicit Euler method will gain energy during the time, i.e. the particle will increase kinetic energy which finally will lead to an escape of the Lennard-Jones potential well. Solving for the modified equation of the Explicit Euler, one gets as a first correction

$$\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}} + \frac{\delta t}{2} \frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{98}$$

$$\dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} + \frac{\delta t}{2} \mathbf{p} \frac{\partial^2 \mathcal{H}}{\partial \mathbf{p}^2} \tag{99}$$

Figure 11 shows results for the integration of equations of motion with the Explicit Euler scheme. Different time steps for integration were applied which show a faster escape from a stable orbit with increasing time step. Also plotted in the same figure is the solution of the modified equations with a high order symplectic scheme, which can be considered as *exact* on these time scales. It is found that the trajectories more or less coincide and cannot be distinguished by eye. A more quantitative analysis (Figure 11) shows that for relatively long times the solution is rather well approximated by the modified equation, although with increasing time the differences between solutions become more pronounced. This means

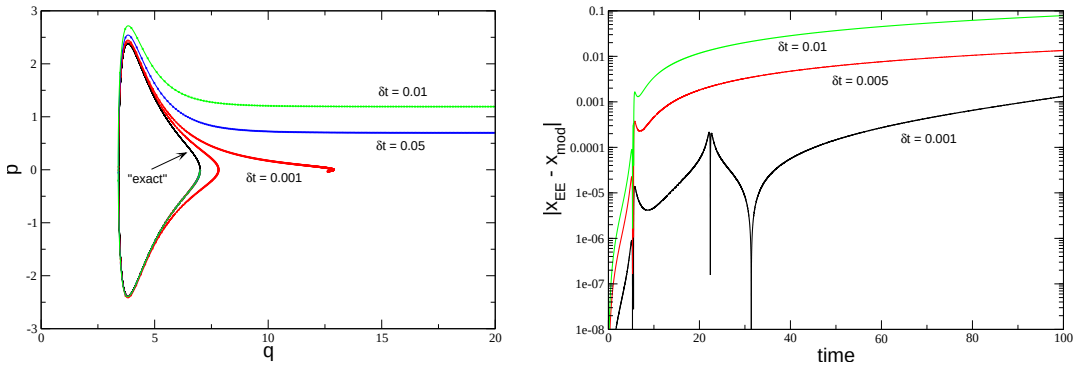


Figure 11: Phase space trajectories of the Lennard-Jones oscillator calculated with the Explicit Euler scheme and different time steps of integration. The *exact* solution (numerical solution of a high order composition scheme with small time step) is shown as a reference - it forms closed orbits. Superimposed to the solutions are results, obtained with a Velocity Verlet scheme, applied to the modified equations, Eqs. (98,99). The right figure shows the differences in coordinates between the calculation with Explicit Euler scheme applied to Lennard-Jones oscillator and Velocity Verlet applied to the modified equation, $|\mathbf{q}_{EE}(t) - \mathbf{q}_{mod}(t)|$.

that for longer times it would be necessary to include more terms of higher order in δt into the modified equation. It should be mentioned that, in general, the series expansion of the modified equation diverges.

References

1. K. Binder and D. Landau. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, Cambridge, 2000.
2. A.R. Leach. *Molecular Modelling - Principles and Applications*. Pearson Education Ltd., Essex, England, 2001.
3. T. Schlick. *Molecular Modeling and Simulation*. Springer, New York, 2002.
4. K. Binder and D.W. Heermann. *Monte Carlo Simulation in Statistical Physics*. Springer, Berlin, 1997.
5. D. Frenkel and B. Smit. *Understanding molecular simulation. From algorithms to applications*. Academic Press, San Diego, 1996.
6. J. M. Haile. *Molecular Dynamics Simulation*. Wiley, New York, 1997.
7. H. Goldstein, Ch. Poole, and J. Safko. *Classical Mechanics*. Addison Wesley, San Francisco, CA, 2002.
8. B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge University Press, Cambridge, 2004.
9. J. Grotendorst, D. Marx, and A. Muramatsu, editors. *Quantum simulations of many-body systems: from theory to algorithms*, volume 10, Jülich, 2001. John von Neumann Institute for Computing.
10. S. Blügel J. Grotendorst and D. Marx, editors. *Computational Nanoscience - Do it yourself*, volume 31, Jülich, 2006. John von Neumann Institute for Computing.
11. J.G. Gay and B.J. Berne. *J. Chem. Phys.*, 74:3316, 1981.

12. S. J. Marrink, A. H. de Vries, and A. E. Mark. Coarse grained model for semiquantitative lipid simulations. *J. Phys. Chem. B*, 108:750, 2004.
13. B. J. Alder and T. E. Wainwright. Phase transition for a hard sphere system. *J. Chem. Phys.*, 27:1208–1209, 1957.
14. B. J. Alder and T. E. Wainwright. Studies in molecular dynamics. I. General method. *J. Chem. Phys.*, 31:459, 1959.
15. J. Roth, F. Gähler, and H.-R. Trebin. A molecular dynamics run with 5.180.116.000 particles. *Int. J. Mod. Phys. C*, 11:317–322, 2000.
16. K. Kadau, T. C. Germann, and P. S. Lomdahl. Large-scale molecular-dynamics simulation of 19 billion particles. *Int. J. Mod. Phys. C*, 15:193, 2004.
17. K. Kadau, T. C. Germann, and P. S. Lomdahl. World record: Large-scale molecular-dynamics simulation of 19 billion particles. Technical Report LA-UR-05-3853, Los Alamos NAtional Laboratory, 2005.
18. K. Kadau, T. C. Germann, and P. S. Lomdahl. Molecular-Dynamics Comes of Age: 320 Billion Atom Simulation on BlueGene/L. *Int. J. Mod. Phys. C*, 17:1755, 2006.
19. T. C. Germann and K. Kadau. Trillion-atom molecular dynamics becomes a reality. *Int. J. Mod. Phys. C*, 19:1315–1319, 2008.
20. P.S. Lomdahl, P. Tamayo, N. Gronbach-Jensen, and D.M. Beazley. In G.S. Ansell, editor, *Proc. Supercomputing 93*, page 520, Los Alamitos CA, 1993. IEEE Computer Society Press.
21. D.M. Beazley and P.S. Lomdahl. *Comput. Phys.*, 11:230, 1997.
22. Y. Duan, L. Wang, and P. A. Kollman. The early stage of folding of villin headpiece subdomain observed in 200-nanosecond fully solvated molecular dynamics simultion. *Proc. Natl. Acad. Sci. USA*, 95:9897, 1998.
23. Y. Duan and P. A. Kollman. Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution. *Science*, 282:740, 1998.
24. C. Mura and J.A. McCammon. Molecular dynamics of a κB DNA element: base flipping via cross-strand intercalative stacking in a microsecond-scale simulation. *Nucl. Acids Res.*, 36:4941–4955, 2008.
25. http://www.ccp5.ac.uk/.
26. http://amber.scripps.edu/.
27. http://www.charmm.org.
28. http://www.ks.uiuc.edu/Research/namd/.
29. http://www.emsl.pnl.gov/docs/nwchem/nwchem.html.
30. http://www.gromacs.org.
31. http://www.cs.sandia.gov/ sjplimp/lammps.html.
32. Gregory A. Voth. *Coarse-Graining of Condensed Phase and Biomolecular Systems*. CRC Press, 2008.
33. A. Arkhipov A.Y. Shih, P.L. Freddolino, and K. Schulten. Coarse grained protein-lipid model with application to lipoprotein particles. *J. Phys. Chem. B*, 110:3674–3684, 2006.
34. P.M. Kasson, A. Zomorodian, S. Park, N. Singhal, L.J. Guibas, and V.S. Pande. Persistent voids: a new structural metric for membrane fusion. *Bioinformatics*, 23:1753–1759, 2007.
35. A. J. Stone. Intermolecular potentials. *Science*, 321:787–789, 2008.
36. W. L. Cui, F. B. Li, and N. L. Allinger. *J. Amer. Chem. Soc.*, 115:2943, 1993.

37. N. Nevins, J. H. Lii, and N. L. Allinger. *J. Comp. Chem.*, 17:695, 1996.
38. S. L. Mayo, B. D. Olafson, and W. A. Goddard. *J. Phys. Chem.*, 94:8897, 1990.
39. M. J. Bearpark, M. A. Robb, F. Bernardi, and M. Olivucci. *Chem. Phys. Lett.*, 217:513, 1994.
40. T. Cleveland and C. R. Landis. *J. Amer. Chem. Soc.*, 118:6020, 1996.
41. A. K. Rappé, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff. *J. Amer. Chem. Soc.*, 114:10024, 1992.
42. Z. W. Peng, C. S. Ewig, M.-J. Hwang, M. Waldman, and A. T. Hagler. Derivation of class ii force fields. 4. van der Waals parameters of Alkali metal cations and Halide anions. *J. Phys. Chem.*, 101:7243–7252, 1997.
43. W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Amer. Chem. Soc.*, 117:5179–5197, 1995.
44. A. D. Mackerell, J. Wiorkiewiczkuczera, and M. Karplus. *J. Amer. Chem. Soc.*, 117:11946, 1995.
45. W. L. Jorgensen, D. S. Maxwell, and J. Tiradorives. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Amer. Chem. Soc.*, 118:11225–11236, 1996.
46. T. A. Halgren. Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *J. Comp. Chem.*, 17:490–519, 1996.
47. J. Kong. Combining rules for intermolecular potential paramters. II. Rules for the Lennard-Jones (12-6) potential and the Morse potential. *J. Chem. Phys.*, 59:2464–2467, 1973.
48. M. Waldman and A. T. Hagler. New combining rules for rare gas van der Waals parameters. *J. Comp. Chem.*, 14:1077, 1993.
49. J. Delhommelle and P. Millié. Inadequacy of the Lorentz-Bertelot combining rules for accurate predictions of equilibrium properties by molecular simulation. *Molec. Phys.*, 99:619–625, 2001.
50. L. Verlet. Computer experiments on classical fluids. I. Thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98, 1967.
51. G. Sutmann and V. Stegailov. Optimization of neighbor list techniques in liquid matter simulations. *J. Mol. Liq.*, 125:197–203, 2006.
52. R. W. Hockney. The potential calculation and some applications. *Meth. Comput. Phys.*, 9:136–211, 1970.
53. R. W. Hockney, S. P. Goel, and J. W. Eastwood. Quite high-resolution computer models of a plasma. *J. Comp. Phys..*, 14:148, 1974.
54. G. Sutmann and V. Stegailov (to be published).
55. P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, 64:253, 1921.
56. S. W. de Leeuw, J. M. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants. *Proc. R. Soc. London*, A373:27, 1980.
57. S. W. de Leeuw, J. M. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. II. Equivalence of boundary conditions. *Proc. R. Soc. London*, A373:57, 1980.

58. T. Darden, D. York, and L. Pedersen. A NlogN method for Ewald sums in large systems. *J. Chem. Phys.*, 98:10089, 1993.
59. U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen. A smooth particle mesh ewald method. *J. Chem. Phys.*, 103:8577, 1995.
60. R. W. Hockney and J. W. Eastwood. *Computer simulation using particles*. McGraw-Hill, New York, 1981.
61. M. Deserno and C. Holm. How to mesh up Ewald sums. I. a theoretical and numerical comparison of various particle mesh routines. *J. Chem. Phys.*, 109:7678, 1998.
62. M. Deserno and C. Holm. How to mesh up Ewald sums. II. an accurate error estimate for the P3M algorithm. *J. Chem. Phys.*, 109:7694, 1998.
63. L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Phys.*, 73:325, 1987.
64. H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *J. Comp. Phys..*, 155:468–498, 1999.
65. C. A. White and M. Head-Gordon. Derivation and efficient implementation of the fast multipole method. *J. Chem. Phys.*, 101:6593–6605, 1994.
66. C. A. White and M. Head-Gordon. Rotating around the quartic angular momentum barrier in fast multipole method calculations. *J. Chem. Phys.*, 105:5061–5067, 1996.
67. C. A. White and M. Head-Gordon. Fractional tiers in fast multipole method calculations. *Chem. Phys. Lett.*, 257:647–650, 1996.
68. H. Dachsel. An improved implementation of the fast multipole method. In *Proceedings of the 4th MATHMOD Vienna*, Vienna, 2003.
69. J. E. Barnes and P. Hut. A hierarchical O($N \log N$) force calculation algorithm. *Nature*, 324:446, 1986.
70. S. Pfalzner and P. Gibbon. *Many Body Tree Methods in Physics*. Cambridge University Press, New York, 1996.
71. G. Sutmann and B. Steffen. A particle-particle particle-multigrid method for long-range interactions in molecular simulations. *Comp. Phys. Comm.*, 169:343–346, 2005.
72. G. Sutmann and S. Wädow. A Fast Wavelet Based Evaluation of Coulomb Potentials in Molecular Systems. In U.H.E. Hansmann, editor, *From Computational Biophysics to Systems Biology 2006*, volume 34, pages 185–189, Jülich, 2006. John von Neumann Institute for Computing.
73. N. W. Ashcroft and N. D. Mermin. *Solid State Physics*. Saunders College Publishing, Fort Worth, 1976.
74. R. A. Robinson and R. H. Stokes. *Electrolyte Solutions*. Butterworth, London, 1965.
75. J. W. Perram, H. G. Petersen, and S. W. de Leeuw. An algorithm for the simulation of condensed matter which grows as the 3/2 power of the number of particles. *Molec. Phys.*, 65:875–893, 1988.
76. D. Fincham. Optimisation of the Ewald sum for large systems. *Molec. Sim.*, 13:1–9, 1994.
77. W. Smith. Point multipoles in the Ewald sum. *CCP5 Newsletter*, 46:18–30, 1998.
78. T. M. Nymand and P. Linse. Ewald summation and reaction field methods for potentials with atomic charges, dipoles and polarizabilities. *J. Chem. Phys.*, 112:6152–6160, 2000.
79. G. Salin and J. P. Caillol. Ewald sums for yukawa potentials. *J. Chem. Phys.*,

113:10459–10463, 2000.
80. L. Greengard. *The rapid evaluation of potential fields in particle systems*. MIT press, Cambridge, 1988.
81. L. Greengard. The numerical solution of the N-body problem. *Computers in Physics*, 4:142–152, 1990.
82. L. Greengard. Fast algorithms for classical physics. *Science*, 265:909–914, 1994.
83. J. D. Jackson. *Classical Electrodynamics*. Wiley, New York, 1983.
84. M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover Publ. Inc., New York, 1972.
85. R. K. Beatson and L. Greengard. A short course on fast multipole methods. In M. Ainsworth, J. Levesley, W.A. Light, and M. Marletta, editors, *Wavelets, Multilevel Methods and Elliptic PDEs*, pages 1–37. Oxford University Press, 1997.
86. C. G. Lambert, T. A. Darden, and J. A. Board. A multipole-based algorithm for efficient calculation of forces and potentials in macroscopic periodic assemblies of particles. *J. Comp. Phys.*., 126:274–285, 1996.
87. M. Challacombe, C. A. White, and M. Head-Gordon. Periodic boundary conditions and the fast multipole method. *J. Chem. Phys.*, 107:10131–10140, 1997.
88. S. J. Marrink, E. Lindahl, O. Edholm, and A. E. Mark. Simulation of the spontaneous aggregation of phospholipids into bilayers. *J. Am. Chem. Soc.*, 123:8638, 2001.
89. Michael L. Klein and Wataru Shinoda. Large-scale molecular dynamics simulations of self-assembling systems. *Science*, 321:798 – 800, 2008.
90. A. H. de Vries, A. E. Mark, and S. J. Marrink. Molecular dynamics simulation of the spontaneous formation of a small dppc vesicle in water in atomistic detail. *J. Am. Chem. Soc.*, 126:4488, 2004.
91. G. Srinivas, S. O. Nielsen, P. B. Moore, and M. L. Klein. Molecular dynamics simulations of surfactant self-organization at a solid-liquid interface. *J. Am. Chem. Soc.*, 128:848, 2006.
92. S. J. Marrink X. Periole, Th. Huber and Th. P. Sakmar. Protein-coupled receptors self-assemble in dynamics simulations of model bilayers. *J. Am. Chem. Soc.*, 129:10126, 2007.
93. W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.*, 79:926–935, 1983.
94. W. L. Jorgensen and J. D. Madura. Temperature and size dependence for Monte Carlo simulations of TIP4P water. *Mol. Phys.*, 56:1381–1392, 1985.
95. H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma. The missing term in effective pair potentials. *J. Phys. Chem.*, 91:6269, 1987.
96. M. W. Mahoney and W. L. Jorgensen. A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. *J. Chem. Phys.*, 112:8910–8922, 2000.
97. S. J. Marrink. The martini force field: Coarse grained model for biomolecular simulations. *J. Phys. Chem. B*, 111:7812, 2007.
98. R.L. Henderson. A uniqueness theorem for fluid pair correlation functions. *Phys. Lett.*, 49 A:197–198, 1974.
99. A.P.Lyubartsev and A.Laaksonen. Reconstruction of pair interaction potentials from radial distribution functions. *Comp. Phys. Comm.*, 121-122:57–59, 1999.

100. A.P.Lyubartsev and A.Laaksonen. Determination of pair potentials from ab-initio simulations: Application to liqid water. *Chem. Phys. Lett.*, 325:15–21, 2000.
101. V.Lobaskin, A.P.Lyubartsev, and P.Linse. Effective macroion-macroion potentials in asymmetric electrolytes. *Phys. Rev. E*, 63:020401, 2001.
102. A.P.Lyubartsev and A.Laaksonen. Calculation of effective interaction potentials from radial distribution functions: A reverse Monte Carlo approach. *Comp. Phys. Comm.*, 121-122:57–59, 1999.
103. T.C. Terwilliger. Improving macromolecular atomic models at moderate resolution by automated iterative model building, statistical density modification and refinement. *Acta Cryst.*, D59:1174–1182, 2003.
104. H. F. Trotter. On the product of semi-groups of operators. *Proc. Am. Math. Soc.*, 10:545–551, 1959.
105. O. Buneman. Time-reversible difference procedures. *J. Comp. Phys..*, 1:517–535, 1967.
106. E. Hairer and P. Leone. Order barriers for symplectic multi-value methods. In D. Griffiths, D. Higham, and G. Watson, editors, *Pitman Research Notes in Mathematics*, volume 380, pages 133–149, 1998.
107. D. Okunbor and R. D. Skeel. Explicit canonical methods for Hamiltonian systems. *Math. Comput.*, 59:439–455, 1992.
108. E. Hairer. Backward error analysis of numerical integrators and symplectic methods. *Ann. Numer. Math.*, 1:107–132, 1994.
109. E. Hairer and C. Lubich. The lifespan of backward error analysis for numerical integrators. *Numer. Math.*, 76:441–462, 1997.
110. S. Reich. Backward error analysis for numerical integrators. *SIAM J. Numer. Anal.*, 36:1549–1570, 1999.
111. D. M. Stoffer. *Some geometrical and numerical methods for perturbed integrable systems*. PhD thesis, Swiss Federal Institute of Technology, Zürich, 1988.
112. J. M. Sanz-Serna M. Calvo. *Numerical Hamiltonian Problems*. Chapman and Hall, London, 1994.
113. R. D. Skeel. Integration schemes for molecular dynamics and related applications. In M. Ainsworth, J. Levesley, and M. Marletta, editors, *The Graduate Student's Guide to Numerical Analysis*, pages 119–176, New York, 1999. Springer.
114. M. E. Tuckerman and W. Langel. Multiple time scale simulation of a flexible model of co_2. *J. Chem. Ohys.*, 100:6368, 1994.
115. P. Procacci, T. Darden, and M. Marchi. A very fast Molecular Dynamics method to simulate biomolecular systems with realistic electrostatic interactions. *J. Phys. Chem.*, 100:10464–10468, 1996.
116. P. Procacci, M. Marchi, and G. L. Martyna. Electrostatic calculations and multiple time scales in molecular dynamics simulation of flexible molecular systems. *J. Chem. Phys.*, 108:8799–8803, 1998.
117. P. Procacci and M. Marchi. Taming the Ewald sum in molecular dynamics simulations of solvated proteins via a multiple time step algorithm. *J. Chem. Phys.*, 104:3003–3012, 1996.
118. J. J. Biesiadecki and R. D. Skeel. Dangers of multiple time step methods. *J. Comp. Phys..*, 109:318–328, 1993.
119. J. L. Scully and J. Hermans. Multiple time steps: limits on the speedup of molecular

dynamics simulations of aqueous systems. *Molec. Sim.*, 11:67–77, 1993.
120. B. J. Leimkuhler and R. D. Skeel. Symplectic numerical integrators in constrained Hamiltonian systems. *J. Comp. Phys..*, 112:117–125, 1994.
121. T. Schlick. Some failures and success of long timestep approaches to biomolecular simulations. In P. Deuflhard, J. Hermans, B. J. Leimkuhler, A. Mark, S. Reich, and R. D. Skeel, editors, *Lecture notes in computational science and engineerung. Algorithms for macromolecular modelling*, volume 4, pages 221–250, New York, 1998. Springer.
122. E. Barth and T. Schlick. Overcoming stability limitations in biomolecular danamics. I. Combining force splitting via extrapolation with Langevin dynamics. *J. Chem. Phys.*, 109:1617–1632, 1998.
123. E. Barth and T. Schlick. Extrapolation versus impulse in multiple-timestepping schemes. II. Linear analysis and applications to Newtonian and Langevin dynamics. *J. Chem. Phys.*, 109:1633–1642, 1998.
124. B. Garcia-Archilla, J. M. Sanz-Serna, and R. D. Skeel. Long-time-step methods for oscillatory differential equations. *SIAM J. Sci. Comp.*, 20:930–963, 1998.
125. B. Garcia-Archilla, J. M. Sanz-Serna, and R. D. Skeel. The mollified impulse method for oscillatory differential equations. In D. F. Griffiths and G. A. Watson, editors, *Numerical analysis 1997*, pages 111–123, London, 1998. Pitman.
126. B. Garcia-Archilla, J. M. Sanz-Serna, and R. D. Skeel. The mollified impulse method for oscillatory differential equations. *SIAM J. Sci. Comp.*, 20:930–963, 1998.
127. J. A. Izaguirre. *Longer time steps for molecular dynamics*. PhD thesis, University of Illinois at Urbana-Champaign, 1999.
128. J. A. Izaguirre, S. Reich, and R. D. Skeel. Longer time steps for molecular dynamics. *J. Chem. Phys.*, 110:9853, 1999.
129. B. V. Chirikov. A universal instability of many-dimensional oscillator systems. *Phys. Rep.*, 52:264–379, 1979.
130. F. Calvo. Largest Lyapunov exponent in molecular systems: Linear molecules and application to nitrogen clusters. *Phys. Rev. E*, 58:5643–5649, 1998.
131. R. F. Warming and B. J. Hyett. The modified equation approach to the stability and accuracy analysis of finite difference methods. *J. Comp. Phys..*, 14:159–179, 1974.

Fourier Transform-Based Methods for Long-Range Interactions: Ewald, P^{3}M and More

Axel Arnold

Institute for Computational Physics
Universität Stuttgart
Pfaffenwaldring 27, 70569 Stuttgart, Germany
E-mail: arnolda@icp.uni-stuttgart.de

Fourier transform-based methods for the calculation of electrostatic interactions were developed by Ewald as a tool for the calculation of crystal lattice energies in the 1920's, long before the appearance of computers. Nevertheless, the Ewald summation is still widely used for computer simulations of small systems. For larger systems, there are several extensions using fast fourier transforms, the so-called mesh-based Ewald methods, such as P^{3}M, PME or SPME. This methods are used in most classical molecular dynamics simulations, especially of biological and other soft matter. We will review the classical Ewald sum and the P^{3}M approach, as well as extensions for systems with partially periodic boundary conditions or dipolar interactions. We provide error formulas which allow to tune the algorithm for optimal computational speed at given accuracy.

1 Introduction

Computer simulations are by now an established tool to determine material properties *ab initio* or in general to investigate processes on the nano scale. On this scale, it is often crucial to include the long-ranged electrostatic interaction, especially when simulating biological matter. However, the system sizes that can be handled in simulations are small compared to the real, physical dimensions, which drastically enhances the influence of boundary effects. To avoid this, one typically uses of periodic boundaries.

For this kind of boundary conditions, the famous Ewald sum[1–4] does a remarkable job in splitting the very slowly converging sum over the Coulomb potential into two exponentially converging sums. Moreover, this methods reduces the computational complexity from $\mathcal{O}(N^2)$ if the interaction of all charges with all charges were to be calculated, to a more favorable scaling of $\mathcal{O}(N^{3/2})$. This requires the use of cutoffs which are optimized with respect to the splitting parameter; this is easily done since error formulas for the Ewald truncation errors exist[5].

By replacing the charges with a regular mesh, one can use FFT methods to speed up the Ewald method to a computation time of $\mathcal{O}(N \log N)$. Since there are various possible schemes to interpolate the charges onto the mesh and to calculate the forces from the mesh, several different schemes of mesh-based Ewald methods exist, such as P^{3}M, PME or SPME. We will discuss here the P^{3}M method, since it is known to be the computationally optimal variant[6,7].

Thin polyelectrolyte films or interactions of charged species with membranes cannot be studied by fully periodic boundary conditions. For these systems, partially periodic boundary conditions with only two periodically replicated dimensions are required, while the third one has a finite extend h ($2D + h$ geometry). In this geometry, the Ewald formula

is only slowly convergent and has an unfavorable $\mathcal{O}(N^2)$ scaling and no *a priori* error estimates exist[8]. Because of the superior scaling of the methods for fully periodic systems, there have been early attempts to use the fully 3D Coulomb sum also for slab problems by leaving a gap in the non-periodic dimension[9–11]. We will present the ELC approach[12,13] which allows to subtract numerically the contributions of the image layers again and to estimate the error that one makes by introducing them.

As a last example of a Fourier-based method, we want to present the extension of the Ewald method to systems which can be modeled by interacting point dipoles. Substances of that kind are ferrofluids, which are basically dispersed magnetic particles[14], magneto-rheological (MR), electro-rheological (ER) fluids or solvents which can be modeled approximately by dipolar interactions like water. The computational $\mathcal{O}(N^{3/2})$-scaling of the Ewald method still applies, and *a priori* error formulas exist[15].

The material in this article has been mainly collected from the sources 6,7,12,13,15,16. A review article on the general topic of long-range interactions in soft matter can be found in 17. As good textbooks for background material we recommend the second edition of Frenkel and Smit[18] and the book by Allen and Tildesley[19].

2 The Standard 3D Ewald Method

We consider a system of N particles with charges q_i at positions $\mathbf{r}_i$ in an overall neutral and, for simplicity, cubic simulation box of length L and volume $V = L^3$. If periodic boundary conditions are applied, the total electrostatic energy of the box is given by

$$U = \frac{1}{2} \sum_{\mathbf{m}\in\mathbb{Z}^3}^{\infty} \sum_{i,j=1}^{N}{}' \frac{q_i q_j}{|\mathbf{r}_{ij} + \mathbf{m}L|}, \tag{1}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $\mathbf{m}$ counts the periodic images, and the prime denotes that the summand for $i = j$ has to be omitted for $\mathbf{m} = \vec{0}$. Due to the long-range nature of the Coulomb interaction, this sum is only conditionally convergent. Therefore its value is not well-defined unless one specifies the precise way in which the cluster of simulation boxes should fill the $\mathbb{R}^3$, i.e., its *shape*[3,4,20,21]. Usually a spherical limit is applied, i. e.

$$U = \frac{1}{2} \sum_{S=0}^{\infty} \sum_{\mathbf{m}^2=S} \sum_{i,j=1}^{N}{}' \frac{q_i q_j}{|\mathbf{r}_{ij} + \mathbf{m}L|}. \tag{2}$$

The energies and forces differ from this spherical limit by some function of the total dipole moment of the system, if another summation order is chosen.

However, the conditional convergence of the Coulomb sum is not the only complication in the treatment of electrostatic interactions. In fact, the Coulomb potential bears *two* intrinsic difficulties. It is slowly decaying at large distances, and strongly varying at small distances. It is the combination of these two properties which leads to severe problems. If only one of them was present, everything would be comparatively easy, since a short-range potential could be treated by a simple cutoff, as it is done, e. g., for interactions of the Lennard-Jones type, and a long-range potential, which is periodic and slowly varying *everywhere*, can accurately be represented by the first few terms of its Fourier series.

Obviously, each of the two complications forbids the simple solution of the other, and the slowly decaying long-range part of the Coulomb potential renders a straightforward summation of Eqn. (1) impracticable. The trick is thus to split the problem into two parts by the trivial identity

$$\frac{1}{r} = \frac{f(r)}{r} + \frac{1-f(r)}{r}. \tag{3}$$

The underlying idea is to distribute the two complications between the two terms in Eqn. (3) by a suitable choice of the splitting function f. In particular:

- The first part $\frac{f(r)}{r}$ should be negligible, or even zero, beyond some cutoff $r_{\max}$, so that the summation up to the cutoff is a good approximation to (or the exact result of) this contribution to the total electrostatic potential.
- The second part $\frac{1-f(r)}{r}$ should be a slowly varying function for *all* r, so that its Fourier transform can be represented by only a few $\mathbf{k}$-vectors with $|\mathbf{k}| \leq k_{\max}$. This permits an efficient calculation of this contribution to the total electrostatic potential in reciprocal space.

Since the field equations are linear, the sum of these two contributions gives the solution for the potential of the original problem.

The two requirements on the splitting function f mentioned above leave a large freedom of choice[2,22,23]. The traditional selection is the complementary error function $\mathrm{erfc}(r) := \frac{2}{\sqrt{\pi}} \int_r^\infty \mathrm{d}t\, \mathrm{e}^{-t^2}$. This results in the well known Ewald formula for the electrostatic energy of the primary box:

$$U = U^{(r)} + U^{(k)} + U^{(s)} + U^{(d)}, \tag{4}$$

where $U^{(r)}$ is the contribution from real space, $U^{(k)}$ the contribution from reciprocal space, $U^{(s)}$ the self energy and $U^{(d)}$ the dipole term. They can be written as[19,18]

$$U^{(r)} = \frac{1}{2} \sum_{\mathbf{m}\in\mathbb{Z}^3} {\sum_{i,j}}' q_i q_j \frac{\mathrm{erfc}(\alpha|\mathbf{r}_{ij} + \mathbf{m}L|)}{|\mathbf{r}_{ij} + \mathbf{m}L|} \tag{5}$$

$$U^{(k)} = \frac{1}{2} \frac{1}{V} \sum_{\mathbf{k}\neq\mathbf{0}} \frac{4\pi}{k^2} e^{-k^2/4\alpha^2} |\tilde{\rho}(\mathbf{k})|^2 \tag{6}$$

$$U^{(s)} = -\frac{\alpha}{\sqrt{\pi}} \sum_i q_i^2, \tag{7}$$

where the Fourier transformed charge density or the structure factor $\tilde{\rho}(\mathbf{k})$ is defined as

$$\tilde{\rho}(\mathbf{k}) = \int_V \mathrm{d}^3 r\ \rho(\mathbf{r})\, e^{-\mathrm{i}\,\mathbf{k}\cdot\mathbf{r}} = \sum_{j=1}^N q_j\ e^{-\mathrm{i}\,\mathbf{k}\cdot\mathbf{r}_j} \qquad \text{where} \qquad \mathbf{k} \in \frac{2\pi}{L}\mathbb{Z}^3. \tag{8}$$

The dipole term

$$U^{(d)} = \frac{2\pi}{(1+2\epsilon')V} \left(\sum_i q_i \mathbf{r}_i \right)^2 \tag{9}$$

is special. First of all note that the term is independent of α, which is due to the fact that this term is not specific to the Ewald approach, but rather a consequence of the conditional

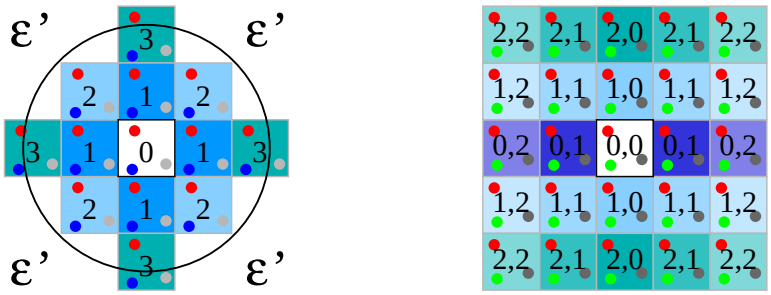


Figure 1: Schema of the spherical (left) and planewise (right) summation orders. The numbers give the order in which the contributions are added up; in the right graph, this is done in lexicographic order, that is first all contributions $0, 0 \ldots \infty$, then contribution $1, 0 \ldots \infty$ and so on. In the planewise summation order, the summation in the x, y–plane is in fact spherical, which cannot be seen in the presented cut.

convergence. The form (9) is specific for the spherical summation order and assumes that the medium outside this sphere is a homogeneous dielectric with dielectric constant ϵ', whereas inside we assume $\epsilon = 1$ (see Fig. 1 left). Due to the surrounding medium, the particles of the growing ball will feel a polarization force. This leads to an additional contribution that will not vanish even in the limit of an infinite ball, although then the complete space is filled by copies of the simulation box. The term does not vanish when embedding the system in vacuum, that is, when perform the purely mathematical, spherical summation, but rather under metallic boundary conditions, i. e. $\epsilon' = \infty$. This solution of the sum is frequently called *intrinsic*, since it corresponds to a truly periodic electrostatic potential, whereas the dipole term is obviously a non-periodic function of the particle coordinates. Its non-periodicity also means that one has to use itinerant, that is continuous, particle coordinates in a simulation to avoid a discontinuous energy contribution[20]. For systems that contain free ions, this has unwanted effects, so that one should always use metallic boundary conditions. A detailed discussion of this term can be found in Refs. 3, 4, 21.

In subsection 4 we will consider a different summation order from the spherical limit, namely slab-wise summation. We add up the particles along z slab–wise, i. e. ordered by increasing z–distance, but radially in x and y (see Fig. 1 right). Smith has shown that the dipole term now takes the form[24]

$$U^{(d)} = \frac{2\pi}{V} \left(\sum_i q_i z_i \right)^2 . \tag{10}$$

Note that in this case, the dielectric medium at infinity does not play a role; the deeper reason is that the the summation in each x, y-plane is no longer shape-dependent.

If the system under investigation is not electrostatically neutral, the infinite sum in Eqn. (1) diverges. It can be made convergent by adding a homogeneously distributed background charge which restores neutrality – a typical situation for one-component plasma simulations. This results in an additional electroneutrality-term $U^{(n)}$ to be included in Eqn. (4), which reads (see, e. g.,[25])

$$U^{(n)} = -\frac{\pi}{2\alpha^2 V} \left(\sum_i q_i \right)^2 . \tag{11}$$

Since the neutralizing background is homogeneous, the correction term in Eqn. (11) is independent of the particle positions. However, the correction term is α-dependent. This means that it is *crucial* to include it — otherwise, the result will depend on the non-physical parameter α. Note that the neutralizing background also has a dipole moment and contributes to the dipole term. Like the particles, its absolute position during the summation matters. If we assume that the background is distributed homogeneously in the central simulation box $[0, L]^3$, then the dipole term for spherical summation should read

$$U^{(d)} = \frac{2\pi}{(1+2\epsilon')V} \left[\sum_i q_i \left(\mathbf{r}_i - \frac{L}{2}\mathbf{e} \right) \right]^2, \tag{12}$$

where $(L/2)\mathbf{e}$ is the center of the simulation box. If the dipole moment of the background is omitted, the dipole term depends on the absolute position of the particles in space, which is certainly unphysical.

The advantage of Eqn. (4) over Eqn. (1)) is that the exponentially converging sums over $\mathbf{m}$ and $\mathbf{k}$ in (5,6) allow the introduction of comparatively small cutoffs $r_{\max}$ and $k_{\max}$ without much loss in accuracy. Typically one chooses α large enough as to employ the minimum image convention in Eqn. (5). The inverse length α, which is often referred to as the Ewald (or splitting) parameter, tunes the relative weights of the real space and the reciprocal space contributions. However, the final result of the exact equation (4), not terminating the sums at some finite cut-off value, is independent of α.

The force $\mathbf{F}_i$ on particle i is obtained by differentiating the electrostatic potential energy U with respect to $\mathbf{r}_i$, i.e.,

$$\mathbf{F}_i = -\frac{\partial}{\partial \mathbf{r}_i} U. \tag{13}$$

Using Eqns. (4–8) one obtains the following Ewald formula for the forces:

$$\mathbf{F}_i = \mathbf{F}_i^{(r)} + \mathbf{F}_i^{(k)} + \mathbf{F}_i^{(d)}, \tag{14}$$

with the real space, Fourier space and dipole contributions given by:

$$\begin{aligned} \mathbf{F}_i^{(r)} = q_i \sum_j q_j \sum_{\mathbf{m}\in\mathbb{Z}^3}{}' \Bigg(& \frac{2\alpha}{\sqrt{\pi}} \exp(-\alpha^2 |\mathbf{r}_{ij} + \mathbf{m}L|^2) \\ & + \frac{\operatorname{erfc}(\alpha|\mathbf{r}_{ij} + \mathbf{m}L|)}{|\mathbf{r}_{ij} + \mathbf{m}L|} \Bigg) \frac{\mathbf{r}_{ij} + \mathbf{m}L}{|\mathbf{r}_{ij} + \mathbf{m}L|^2} \end{aligned} \tag{15}$$

$$\mathbf{F}_i^{(k)} = q_i \sum_j q_j \frac{1}{V} \sum_{\mathbf{k}\neq\mathbf{0}} \frac{4\pi\mathbf{k}}{k^2} \exp\left(-\frac{k^2}{4\alpha^2} \right) \sin(\mathbf{k}\cdot\mathbf{r}_{ij}) \tag{16}$$

$$\mathbf{F}_i^{(d)} = -\frac{4\pi q_i}{(1+2\epsilon')V} \sum_j q_j \left(\mathbf{r}_j - \frac{L}{2}\mathbf{e} \right). \tag{17}$$

Since the self energy in Eqn. (7) and the neutralizing contribution in Eqn. (11) are independent of particle positions, they do not contribute to the force.

2.1 Why and how to control errors

An investigation of the errors connected with any method to calculate the Coulomb sum (or, in fact, any interaction) is important for three reasons:

1. One has to ensure that the errors are small enough so that they don't influence the outcome of the simulation. Most importantly, results have to be independent of the applied summation technique and its tuning parameters.

2. The tuning parameters should be chosen in such a way as to run the algorithm at its optimal operation point to save computing time.

3. Comparing the efficiency of different methods is only fair if it is done at the same level of accuracy.

Errors can tell us if we might see artifacts in simulations due to too small cut-offs, or if our observations have some other, maybe even physical origin. They can tell us how the algorithm scales at its optimal point and they can help us save a lot of expensive computer time. Therefore, error estimates are an required tool to apply any numerical method in computer simulations.

For given finite real- and reciprocal space cutoffs there exists an optimal α such that the accuracy of the approximated Ewald sum is the highest possible. Note that there is no unique or optimal measure of accuracy. In molecular dynamics simulations, the main interest lies in force errors, which we will consider here, while in Monte Carlo simulations, one is concerned with the errors in the energy. One can be interested in either absolute or relative errors; we will consider the first. The reason is again practical — most applications, particulary in soft matter research, include a considerable level of thermal noise on the forces. As long as the absolute error in the electrostatic force is significantly below this noise level, errors should not influence the system, even if the relative errors are large.

In the following, we discuss the force error estimates by Kolafa and Perram for the Ewald sum[26]. Giving a general expression for the expected error is difficult; there are always pathological cases, in which the errors of the used methods are unusually high. However, this are only a few special configurations, which we will rarely encounter in a thermal simulation. Therefore, we assume the most common case, namely that the charges are homogeneously and randomly distributed within the periodic cell V. Our goal is to calculate the root mean square (RMS) error

$$\Delta F = \sqrt{\langle (\mathbf{F}^{\text{exact}} - \mathbf{F}^{\text{Ewald}})^2 \rangle} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \Delta F_i^2}, \tag{18}$$

where $\Delta \mathbf{F}_i = \mathbf{F}_i^{\text{exact}} - \mathbf{F}_i^{\text{Ewald}}$ denotes the error in the force on particle number i.

It is reasonable to assume that the error in the force on particle i can be written as

$$\Delta \mathbf{F}_i = q_i \sum_{j \neq i} q_j \boldsymbol{\chi}_{ij}, \tag{19}$$

that is, we assume that the error on F_i is a sum of errors stemming from the $N-1$ interactions with the other charges. χ_{ij} is a pairwise error, which is algorithm dependent; we assume that

$$\langle \boldsymbol{\chi}_{ij} \cdot \boldsymbol{\chi}_{ik} \rangle = \delta_{jk} \langle \boldsymbol{\chi}_{ij}^2 \rangle = \delta_{jk} \, \chi^2, \tag{20}$$

i. e. that the contributions from different particles are uncorrelated and that the magnitude is independent of the particle properties, except for the charge prefactor. The first assumption

is justified only for random systems, as we have assumed. Inserting into Eqn. (19) gives

$$\left\langle \Delta F_i^2 \right\rangle = q_i^2 \sum_{j \neq i} \sum_{k \neq i} q_j q_k \langle \boldsymbol{\chi}_{ij} \cdot \boldsymbol{\chi}_{ik} \rangle = q_i^2 \, \chi^2 \sum_{j=1}^{N} q_j^2, \tag{21}$$

which shows that the RMS force error has the form

$$\Delta F \approx \frac{\sum q_i^2}{\sqrt{N}} \chi, \tag{22}$$

where the factor χ is the only constant that is algorithm-dependent[6].

Coming back to the Ewald sum, we further simplify the problem by assuming that the real and Fourier space errors, stemming from truncating Eqns. (15) and (16), respectively, are uncorrelated; this is sensible, since the parts are calculated by very different types of algorithms. In this case, we can calculate their contributions separately:

$$\Delta F^2 = \Delta F_{\text{real}}^2 + \Delta F_{\text{Fourier}}^2 \approx \frac{\sum q_i^2}{\sqrt{N}} \left(\chi_{\text{real}} + \chi_{\text{Fourier}} \right). \tag{23}$$

By replacing the sums over n and k in Eqns. (15) and (16) by integrals, the constants χ_{real} and χ_{Fourier} can be estimated. The resulting errors are[26]

$$\Delta F_{\text{real}} \approx \sum q_i^2 \frac{2}{\sqrt{N r_{\max} V}} \exp\left(-\alpha^2 r_{\max}^2\right) \qquad \text{and} \tag{24}$$

$$\Delta F_{\text{Fourier}} \approx \sum q_i^2 \frac{2\alpha}{\sqrt{N \pi k_{\max} V}} \exp\left(-\frac{k_{\max}^2}{4\alpha^2}\right). \tag{25}$$

This error estimates allow to estimate the error at given $k_{\max}$, $r_{\max}$ and α, so that one can numerically determine the optimal value α with minimal overall error at given fixed $k_{\max}$ and $r_{\max}$. Moreover, we can estimate the computation time scaling of the Ewald sum: from the equations, it is easy to see that when changing α, the real space cutoff $r_{\max}$ should be inversely proportional to α to maintain a constant truncation error, while $k_{\max}$ should be proportional to it. Assuming homogeneously distributed particles at a fixed density ρ, the computation time for the evaluation of the real space sum using a cell list-like approach is $\mathcal{O}(N \rho \, r_{\max}^3)$. The number of k-vectors we need to sum over in Eqn. (6) grows proportional to $V k_{\max}^3$, so the computation time of the Fourier space sum is $\mathcal{O}(k_{\max}^3 N^2/\rho)$. Inserting the expected relations between α and $r_{\max}$ and $k_{\max}$, the overall computation time is [27]

$$\mathcal{T} = \mathcal{O}(N \alpha^{-3}) + \mathcal{O}(N^2 \alpha^3), \tag{26}$$

which is minimal for $\alpha = N^{-1/6}$. Therefore, the optimal scaling of the Ewald sum is $\mathcal{O}(N^{3/2})$. This, however, may require that $r_{\max} > L/2$, prohibiting the simple minimum image convention in real space and rendering this procedure less tempting.

The error estimates can also be used to tune the Ewald sum to consume as little computer time as possible for a given error goal $\Delta \mathbf{F} \approx \tau$. A good approximation to the optimal α with the minimal error can be found by simply requiring the real space and k-space errors to be equal. Therefore, given a Fourier-cutoff $k_{\max}$, one can determine α numerically from Eqn. (25), such that $\Delta F_{\text{Fourier}} \approx \tau/\sqrt{2}$. Then, Eqn. (24) allows to determine the cutoff $r_{\max}$, such that also the real space error is $\approx \tau/\sqrt{2}$, and the total error is $\Delta \mathbf{F} \approx \tau$. By this, one can try different values for $k_{\max}$, determine optimal values for α and $r_{\max}$, and measure the required computation time for this combination. Since one can assume

that the computation time has one minimum with respect to $k_{\max}$, one can easily find the optimal value for the given hardware.

3 Mesh-Accelerated Ewald Methods (P^{3}M)

The Fourier transformations involved in Eqn. (6) are the most time consuming part of the Ewald sum. Mesh-accelerated Ewald methods are based on the idea to modify the problem in such a way that it permits application of the Fast Fourier Transformation (FFT). This reduces the complexity of the reciprocal part of the Ewald sum to $\mathcal{O}(N \log N)$ and allows a constant real space cutoff, so that the real space computation time scales like $\mathcal{O}(N)$.

Performing the Fourier transformations in the reciprocal space part of the Ewald sum by FFT routines is by no means straightforward, and consists of four main steps:

1. The point charges with continuous coordinates have to be replaced by a grid based charge density, since the FFT is a discrete and finite transformation.

2. The potential has to be calculated in the discrete Fourier space by solving Poisson's equation; that is, by multiplication of the Fourier transformed charge density with the Green's function. It is neither obvious nor true that the best grid approximation to the continuum solution of the Poisson equation is achieved by using the continuum Green's function $4\pi/k^2$.

3. The electric field has to be calculated by differentiation from the electric potential. There are at least three possibilities for implementing this differentiation, which differ in accuracy and speed.

4. Finally, the forces on the particle have to calculated from the electric field that is known only on the discrete grid points. This can – under certain circumstances – lead to unwanted violations of Newton's third law. They can be anything between harmless and disastrous.

There exist three major mesh-based Ewald summation methods – similar in spirit but different in detail, namely in how the four steps above are performed. The oldest is the original particle-particle–particle-mesh (P^{3}M) method of Hockney and Eastwood[28], and then there are two variants, namely the Particle Mesh Ewald (PME) method of Darden *et al.*[29] and an extension of the latter by Essmann *et al.*[30], which is usually referred to as Smooth Particle Mesh Ewald (SPME). Deserno *et al.* have shown how the three methods differ in detail, and it was demonstrated that the oldest method, namely the original P^{3}M algorithm is actually the most accurate one[6]. Since in addition error estimates exist[7], this mesh method should be the preferred method of choice, and will be introduced here.

3.1 P^{3}M in a Nutshell

The P^{3}M method maps the system onto a mesh, such that the necessary Fourier transformations can be accomplished by Fast Fourier routines. At the same time the simple Coulomb Green function $4\pi/k^2$ is adjusted to make the result of the mesh calculation most closely resemble the continuum solution.

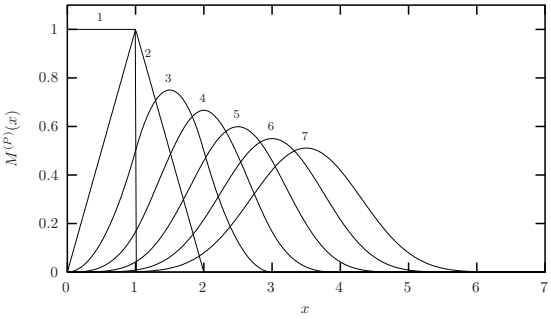


Figure 2: Sketch of the first 7 cardinal-B-splines $M^{(P)}(x)$, parameterized by P. Note that the charge assignment functions $W^{(P)}(x)$ for the P^{3}M algorithm are just the "centered" B-splines.

The first step, i.e., generating the mesh based charge density $\rho_{\mathbb{M}}$ (defined at the mesh points $\mathbf{r}_p \in \mathbb{M}$), is carried out with the help of a charge assignment function W:

$$\rho_{\mathbb{M}}(\mathbf{r}_p) = \frac{1}{h^3} \sum_{i=1}^{N} q_i W(\mathbf{r}_p - \mathbf{r}_i). \tag{27}$$

Here h is the mesh spacing, and the number of mesh points $N_{\mathbb{M}} = L/h$ along each direction should preferably be a power of two, since in this case the FFT is most efficient. The charge assignment function is classified according to its order P, i.e. between how many grid points – per coordinate direction – each charge is distributed. For W a cardinal B-spline[31] is chosen, which is a piecewise polynomial function of weight one. The order P gives the number of sections in the function. The first 7 cardinal-B-splines are sketched in Fig. 2. Their Fourier transforms are

$$\tilde{W}(\mathbf{k}) = h^3 \left(\frac{\sin(\frac{1}{2}k_x h)}{\frac{1}{2}k_x h} \frac{\sin(\frac{1}{2}k_y h)}{\frac{1}{2}k_y h} \frac{\sin(\frac{1}{2}k_z h)}{\frac{1}{2}k_z h} \right)^P \tag{28}$$

The second and third step, i. e. solving Poisson's equation and deriving the mesh-based electric field $\mathbf{E}(\mathbf{r}_p)$ from it, happen simultaneously. There exist several alternatives for implementing the differentiation on a lattice[6]; here we will restrict ourselves to the case of i$\mathbf{k}$-*differentiation*, that is, multiplying the Fourier transformed potential with i$\mathbf{k}$. In this case $\mathbf{E}(\mathbf{r}_p)$ can be written as

$$\mathbf{E}(\mathbf{r}_p) \;=\; \overleftarrow{\text{FFT}} \left[-\text{i}\mathbf{k} \;\times\; \hat{G}_{\text{opt}} \;\times\; \overrightarrow{\text{FFT}} \left[\rho_{\mathbb{M}}\right] \right] (\mathbf{r}_p). \tag{29}$$

In words, $\mathbf{E}(\mathbf{r}_p)$ is the *backward* finite Fourier transform of the product of $-i\mathbf{k}$, the *forward* finite Fourier transform of the mesh based charge density $\rho_{\mathbb{M}}$ and the so-called optimal influence function $\hat{G}_{\text{opt}}$, given by

$$\hat{G}_{\text{opt}}(\mathbf{k}) = \frac{\text{i}\mathbf{k} \cdot \sum_{\mathbf{m}\in\mathbb{Z}^3} \tilde{U}^2(\mathbf{k} + \frac{2\pi}{h}\mathbf{m}) \tilde{\boldsymbol{R}}(\mathbf{k} + \frac{2\pi}{h}\mathbf{m})}{|\mathbf{k}|^2 \left[\sum_{\mathbf{m}\in\mathbb{Z}^3} \tilde{U}^2(\mathbf{k} + \frac{2\pi}{h}\mathbf{m}) \right]^2}, \tag{30}$$

where the true, analytic reference force from (6) is

$$\tilde{\boldsymbol{R}}(\mathbf{k}) := -\mathrm{i}\mathbf{k}\frac{4\pi}{k^2}e^{-k^2/4\alpha^2} \tag{31}$$

and the dimensionless Fourier transform of the B-spline

$$\tilde{U}(\mathbf{k}) := \tilde{W}(\mathbf{k})/h^3. \tag{32}$$

The last step of P^{3}M, the back-interpolation of the forces onto the particles, is performed again using the B-splines. The force on particle i is determined as

$$\mathbf{F}_i = q_i \sum_{\mathbf{r}_p \in \mathbb{M}} \mathbf{E}(\mathbf{r}_p) W(\mathbf{r}_i - \mathbf{r}_p). \tag{33}$$

The sum extends over the complete mesh $\mathbb{M}$; however, since the B-splines have compact support, the sum in fact only extends over a small vicinity of $\mathbf{r}_i$.

Although the presented formulas (27–33) look somewhat complicated, they are rather easy to implement step by step. If the real space cutoff $r_{\max}$ is chosen small enough, so that the real space contribution (15) can be calculated in order $\mathcal{O}(N)$, the complete algorithm is of order $\mathcal{O}(N \log N)$, for details see e. g. the article by Petersen[27]. Note that the mesh-based calculation of the Fourier part does influence other parts of the Ewald summation. This means, that one still has the same dipole term $U^{(d)}$ as before, and can study non-neutral systems only by including the additional contribution from the neutralizing background $U^{(n)}$.

3.2 The error measure of Hockney and Eastwood

While the real space error estimate of Kolafa and Perram[26] of course also applies to the P^{3}M real space sum, the four steps involved in any particle mesh calculation introduce completely different errors than the simple k-space truncation of the standard Ewald sum. In fact, being a discrete Fourier transform, the P^{3}M k-space sum is *not* truncated at all. However, there are new sources of errors, originating, e. g., from discretization, interpolation or aliasing[a] problems. Since these contributions are not independent of each other (reducing one might enhance another), the only reasonable demand is the minimization of the *total* error at given computational effort.

The most interesting ingredient of the P^{3}M method is the optimal influence function from Eqn. (30). It is constructed such that the result of the mesh calculation is as close as possible to the solution of the original continuum problem. More precisely, the P^{3}M method is derived from the requirement that the resulting Fourier space contribution to the force minimizes the the following error measure Q:

$$Q := \frac{1}{h^3}\int_{h^3} \mathrm{d}^3 r_1 \int_V \mathrm{d}^3 r \left[\mathbf{F}(\mathbf{r};\mathbf{r}_1) - \boldsymbol{R}(\mathbf{r})\right]^2 \tag{34}$$

$\mathbf{F}(\mathbf{r};\mathbf{r}_1)$ is the Fourier space contribution of the force between two unit charges at positions $\mathbf{r}_1$ and $\mathbf{r}_1 + \mathbf{r}$ as calculated by the P^{3}M method (note that due to broken rotational and translational symmetry this does in fact depend on the coordinates of *both* particles),

[a] A finite grid cannot represent arbitrarily large k-vectors. Instead, they are folded back into the first "Brillouin zone" and distort there the true spectrum. This effect is usually referred to as "aliasing".

and $\boldsymbol{R}(\mathbf{r})$ is the corresponding exact reference force (whose Fourier transform is just Eqn. (31)). The inner integral over $\mathbf{r}$ scans all particle separations, whereas the outer integral over $\mathbf{r}_1$ averages over all possible locations of the first particle within a mesh cell. Obviously, up to a factor L^{-3} this expression is just the mean square error in the force for two unit charges, in other words, the quantity χ^2 from Eqn. (21). Inserting into Eqn. (22), the RMS force error of an N particle system is given by

$$\Delta F \approx \sum q_i^2 \sqrt{\frac{Q}{NV}}. \tag{35}$$

It is important to realize that Hockney and Eastwood not only provide a closed expression for the optimal influence function $\hat{G}_{\mathrm{opt}}$, but also a closed expression for the corresponding "optimal error" $Q_{\mathrm{opt}} = Q[\hat{G}_{\mathrm{opt}}]$:

$$\begin{aligned} Q_{\mathrm{opt}} = \frac{1}{L^3} \sum_{\mathbf{k}\in\hat{\mathbb{M}}} \Bigg\{ & \sum_{\mathbf{m}\in\mathbb{Z}^3} \left|\tilde{\boldsymbol{R}}(\mathbf{k}+\frac{2\pi}{h}\mathbf{m})\right|^2 \\ & - \frac{\left|\mathrm{i}\mathbf{k}\cdot\sum_{\mathbf{m}\in\mathbb{Z}^3}\tilde{U}^2(\mathbf{k}+\frac{2\pi}{h}\mathbf{m})\tilde{\boldsymbol{R}}^*(\mathbf{k}+\frac{2\pi}{h}\mathbf{m})\right|^2}{|\mathbf{k}|^2\left[\sum_{\mathbf{m}\in\mathbb{Z}^3}\tilde{U}^2(\mathbf{k}+\frac{2\pi}{h}\mathbf{m})\right]^2} \Bigg\}, \end{aligned} \tag{36}$$

where the asterisk denotes the complex conjugate. Admittedly, Eqn. (36) looks rather complicated. Still, in combination with Eqn. (35) it gives the RMS force error of the Fourier contribution of the P^{3}M method. After all, the computation of Q_{opt} and that of $\hat{G}_{\mathrm{opt}}$ are quite similar. It should be emphasized that the formula (36) for the optimal Q-value and the optimal influence function (30) are of a very general nature, and can be applied to different charge assignment functions, reference forces or other differentiation schemes[6].

With the real space error estimate by Kolafa and Perram and the k-space error estimate by Hockney and Eastwood at hand, it is easy to determine the optimal value of the splitting parameter α *a priori* just from the system parameters N, $\sum q_i^2$ and L and the tuning parameters of the algorithm $r_{\max}$, $N_{\mathbb{M}}$, P. Just like for the standard Ewald method, a good approximation to the optimal α can be obtained by requiring the real and k-space RMS force errors to be equal, and a similar tuning routine can be applied, although now the two parameters $N_{\mathbb{M}}$ and P both need to be tried out.

3.3 Improving the performance

Although the P^{3}M in its oldest variant dates back to the early seventies, recent developments allow to improve the accuracy of the mesh based methods, and or to improve the execution speed at fixed accuracy (by reducing the used FFT mesh, for example). Most notably is the technique called interlacing, a technique developed by Hockney and Eastwood[28] and then virtually forgotten until very recently when it was applied to the SPME algorithm[32]. Interlacing has also been applied to the P^{3}M algorithm with the modern differentiation schemes, yielding an accuracy much higher than that of any other particle-mesh algorithm with the same parameters[33].

Interlacing[28,32] consists of replacing the long-range P^{3}M force (33) by the average of two terms:

$$\mathbf{F}_i^{(int)} = \frac{1}{2}\left(\mathbf{F}_i + \mathbf{F}_i^{\mathbf{p}}\right). \tag{37}$$

Here, $\mathbf{F}_i$ is the original P^{3}M force, and the second term $\mathbf{F}_i^{\mathbf{p}}$ is the P^{3}M force calculated for the particles shifted by the vector $\mathbf{p} = (h/2)\mathbf{e}$, i. e. by half of the diagonal of the grid cell. If some particles find themselves outside of the primary box after the shift, their coordinates are wrapped around periodically.

Interlacing involves doubling the amount of computation for the reciprocal part of a particle-mesh algorithm. However, it gives a large increase in the accuracy of the forces that can be converted into a speedup if one changes the parameters of the algorithm[33]. For example, interlacing allows one to double the grid spacing while keeping the same accuracy, and this means reducing the number of flops for the computation of the long-range Ewald forces by the factor of 8/2=4.

It is very important that adding interlacing to an existing particle-mesh implementation only involves a relatively minor modification. As a quick check of the validity of the algorithm, one can just make two copies of the particle data, compute two forces and average. At a later stage one can improve performance by merging the two computations. The two real-to-complex FFTs necessary for the two calculations of the Fourier transformed charge density can be performed simultaneously in a single complex-to-complex FFT, which is typically faster than two real-to-complex FFTs. Alternatively, the two real-to-complex FFTs of the interlaced algorithm can be done in parallel. Note, however, that the optimal influence function of the conventional P^{3}M is no longer optimal in case of the interlaced P^{3}M and needs to be adjusted. The optimized influence function for the interlaced P^{3}M can be found in Ref. 33.

Interlacing was originally developed in Ref. 28 for the P^{3}M with finite differences. However, when applying interlacing to a finite difference scheme, one does not get an increase in accuracy comparable with those for other differentiation schemes. The reason for this is probably that a finite difference approximation of not very high order is a significant source of error by itself[6,28] that dominates once the other error sources are suppressed by interlacing. When applied with the correct influence function to P^{3}M, interlacing can give an increase in accuracy of the forces by up to two orders of magnitude.

Most notable is, that the interlaced P^{3}M algorithms have more than an order of magnitude higher accuracy than the interlaced SPME for the same execution speed. A typical simulation package computes the Coulombic forces via a conventional analytically differentiated SPME. Converting it to the interlaced P^{3}M with analytic differentiation involves making a relatively minor change of the code, and will give a large increase in accuracy. Typically, one can double the mesh spacing, that is, the same accuracy can be reached in less than a quarter of the computation time.

3.4 Parallelization

Being an order $\mathcal{O}(N \log N)$-method, P^{3}M and other mesh-based Ewald methods are well suited to study even large systems with many thousands of particles. This quickly raises the question of parallelization, i. e. employing N_P processors to compute the electrostatic energy simultaneously. Just as with the charge interpolation and differentiation, there are

several ways of parallelization; we briefly present here one that we found to scale rather well even on large computers with hundreds of processors.

The real space part of the Ewald sum is a short-ranged potential, for which several good parallelization strategies exist. In fact, every parallel Molecular Dynamics code has such a strategy built in, or rather, is built around such a strategy. And to be able to scale up to hundreds of processors, almost all codes use a domain decomposition and cell lists, often combined with Verlet lists. These methods reach the ideal N/N_P scaling with modern fast networks, at least in weak scaling, i. e. constant number of particles per processor. Moreover, communication only occurs between neighboring processors in a 3D toroidal structure, which is efficiently handled by most hardware.

The domain decomposition strategy also allows to conveniently split up the work load of charge assignment and force interpolation, by using the same domain decomposition also for the k-space mesh. What remains, is computing the 3D Fourier transform. The Fourier transforms are typically performed by highly efficient libraries, e. g. the excellent FFTW[34]. However, this library at present does not scale very well when it comes to parallel 3D Fourier transforms — and the parallelization of the 3D Fourier transform is in fact the major bottle neck. To understand this, one has to see that a 3D Fourier transform of a mesh of ${N_{\mathbb{M}}}^3$ total points consists of three times performing ${N_{\mathbb{M}}}^2$ 1D Fourier transforms of length $N_{\mathbb{M}}$. To perform the latter, all data should be available on one processor. This means, that to perform the 1D Fourier transforms along the different axes, one has to completely exchange the data between all the processors, requiring a lot of communication. However, by assigning the processors by a two dimensional domain decomposition to the plane perpendicular to which the FFTs are performed, the communication overhead can be minimized. When changing the direction of the FFT, each processor still has to send all its data, however, only to $\sqrt{P}$ others, if the number processors is P. Moreover, the communication happens in $\sqrt{P}$ independent planes, that is, processor groups.

To our knowledge the most efficient way to implement the 3D Fourier transform is as follows:

1. redistribute the 3D domain-decomposed mesh such that the processors form a 2D mesh in the x, y-plane, and each processor obtains all data of its domain, in particulary always has one or more *full* columns along the z-axis. Calculate the Fourier transforms along the z-axis.

2. redistribute such that the processors form a 2D mesh in the x, z-plane and have full columns along the y-axis. Calculate the Fourier transforms along the y-axis.

3. redistribute such that the processors form a 2D mesh in the y, z-plane and have full columns along the y-axis. Calculate the Fourier transforms along the x-axis.

After applying the optimal influence function and the difference operator,

4 distribute the data back into the original, 3D domain-decomposed mesh.

4 Electrostatic Layer Correction (ELC)

With the classical Ewald method for small systems with only a few particles, and P^{3}M or other mesh-accelerated Ewald algorithms for large systems, we can efficiently simulate

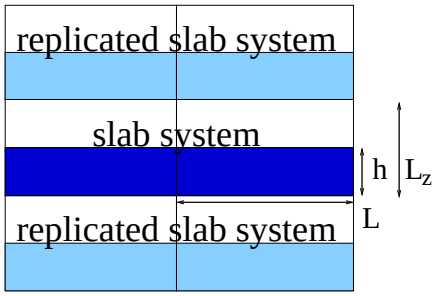


Figure 3: Schematic representation of a fully periodically replicated slab system. h is the range in which particles are located, the space between h and L_z is empty.

all system sizes with fully periodic boundary conditions. However, not all systems can be studied with fully periodic boundary conditions, for example surfaces or thin films. For these systems, we need a slab geometry with only two periodic dimensions, and one non-periodic of finite extension. In this geometry, the Ewald formula is only of order $\mathcal{O}(N^2)$, and no simple mesh-based acceleration possible. Therefore, there have been soon attempts to use a 3D Ewald sum for these slab problems. The main idea is to fill only parts of the simulation box with charges and to leave some space empty, in an attempt to decouple the interactions in the third dimension[9–11] (compare also Fig. 3).

However, in a naive implementation, one finds systematic errors[11]. This is due to the order of summation for the three dimensional Coulomb sum, which is spherical by convention. This order implies that with increasing shell cutoff S, the number of image shells grows faster than the number of shells of the primary layer, namely $\mathcal{O}(S^3)$ versus $\mathcal{O}(S^2)$ (see Fig. 1). In other words, we include the unwanted terms faster than the actually wanted terms. Yeh and Berkowitz[11] already suggested that this problem can be solved by changing the order of summation to planewise summation.

The situation can be improved further by removing numerically the contribution of the artificial images. This contribution is called the electrostatic layer correction (ELC) term and can be calculated in linear computation time[12,13]. In addition, rigorous error estimates for this term exist, from which one can also estimate the error produced by the method proposed by Yeh *et al*. Because of its linear scaling, the ELC term can be combined with P^{3}M or any other mesh-based Ewald method to an overall $\mathcal{O}(N \log N)$-method for slab systems. Typically, a ELC+P^{3}M calculation is at least a factor of two faster than using the approach of Yeh and Berkowitz.

We consider a system where the non-periodic dimension is z without loss of generality. We require that all particles stay in a slab of dimension $0 \leq z_j \leq h$ for some height h, and otherwise assume as before a periodic cell of size $L \times L$ in the x, y-plane. We artificially replicate the system, that is, we introduce copies of the primary layer such that all charges also appear at position $\mathbf{r}_j + \mathbf{m}$, where $\mathbf{m} \in 0 \times 0 \times L_z\mathbb{Z}$; this includes also the periodic images in the x, y-plane. The artificial periodicity L_z should be larger than h, so that we leave a gap of size $L_z - h$.

Since we assume a charge neutral system, the additional image layers (those layers above or below the original slab system) are charge neutral, too. Now let us consider the

m^{th} image layer, which has an offset of mL_z to the original layer. If L_z is large enough, each particle in the m^{th} layer and its replicas in the x, y-plane can be viewed as constituting a homogeneous charged sheet of charge density $\sigma_j = \frac{q_j}{L^2}$. The potential of such a charged sheet at distance z is $2\pi\sigma_j|z|$; the interaction of a pair of image layers located at $z_j \pm mL_z$, with a charge q_i in the primary layer is therefore approximately

$$2\pi q_i \sum_{j=1}^{N} \sigma_j(|z_j - z_i + mL_z| + |z_j - z_i - mL_z|) = 4\pi q_i \, nL_z \sum_{j=1}^{N} \sigma_j = 0. \tag{38}$$

The only errors occurring are those coming from the approximation of assuming homogeneously charged, infinite sheets instead of discrete charges, which should reduce with increasing distance $|m|L_z$ from the central layer. Naturally, this argument only holds when using planewise summation. Yeh and Berkowitz stated that a gap size of at least h is normally sufficient to obtain an moderately accurate result. However, no theoretical estimates exist for the error introduced by the image layers, which could justify this statement; we will see that it is in fact not true. Therefore one might be forced to use even larger gaps to assure that no artifacts are produced by the image layers. One simple deducible artifact is that the pairwise error will be position dependant. Particles in the middle of the slab will see no effect of the image layers due to symmetry, and particles near the surface will encounter for the same reason the largest errors, which is definitely an unwanted feature for studying surface effects.

The ELC term is the energy contribution of the artificially introduced image layers, that is

$$U_{lc} = -\frac{1}{2} \sum_{m>0} \sum_{\mathbf{m}\in\mathbb{Z}^2\times\{\pm mL_z/L\}} \sum_{i,j=1}^{N} \frac{q_i q_j}{|\mathbf{r}_{ij} - \mathbf{m}L|}, \tag{39}$$

Since we assume that the image layers all are separated from the main layer by at least the gap size $L_z - h$, we can use a convergence factor approach to calculate the interaction of the primary layer with the image layer m. More precisely, we use a screened Coulomb potential $\frac{e^{-\beta r}}{r}$ in the limit of infinite screening length β^{-1}. The screened potential can be conveniently Fourier transformed along each of the two periodic coordinates, and one obtains[35,36,12]

$$U_{lc}(m) = -\frac{1}{2} \sum_{i,j=1}^{N} q_i q_j \frac{2\pi}{L^2} \sum_{\substack{\mathbf{k}\in\frac{2\pi}{L}\mathbb{Z}^2 \\ \mathbf{k}^2>0}} \frac{e^{-|\mathbf{k}||z_{ij}-mL_z|}}{|\mathbf{k}|} e^{\mathrm{i}(k_x x_{ij} + k_y y_{ij})} \; - \; \frac{2\pi}{L^2}|z_{ij} - mL_z|. \tag{40}$$

The sum over m to obtain U_{lc} can be performed analytically by a geometric series. The terms $\frac{2\pi}{L^2}|z_{ij} - mL_z|$ are exactly the homogeneous sheet potential, which we have seen to

cancel out for charge neutral systems (see Eqn. (38)). We obtain the ELC term

$$\begin{aligned} U_{lc} =& \frac{\pi}{L^2} \sum_{\substack{\mathbf{k}\in\frac{2\pi}{L}\mathbb{Z}^2 \\ \mathbf{k}^2>0}} \sum_{i,j=1}^{N} q_i q_j \frac{e^{|\mathbf{k}|z_{ij}} + e^{-|\mathbf{k}|z_{ij}}}{f_{pq}(e^{f_{pq}L_z}-1)} e^{\mathrm{i}(k_x x_{ij} + k_y y_{ij})} \\ =& \frac{\pi}{L^2} \sum_{\substack{\mathbf{k}\in\frac{2\pi}{L}\mathbb{Z}^2 \\ \mathbf{k}^2>0}} \sum_{i=1}^{N} q_i \frac{e^{|\mathbf{k}|z_i} e^{\mathrm{i}(k_x x_i + k_y y_i)}}{f_{pq}(e^{|\mathbf{k}|L_z}-1)} \sum_{j=1}^{N} q_j e^{-|\mathbf{k}|z_j} e^{-\mathrm{i}(k_x x_j - k_y y_j)} \\ &+ \frac{\pi}{L^2} \sum_{\substack{\mathbf{k}\in\frac{2\pi}{L}\mathbb{Z}^2 \\ \mathbf{k}^2>0}} \sum_{i=1}^{N} q_i \frac{e^{-|\mathbf{k}|z_i} e^{\mathrm{i}(k_x x_i + k_y y_i)}}{|\mathbf{k}|(e^{|\mathbf{k}|L_z}-1)} \sum_{j=1}^{N} q_j e^{|\mathbf{k}|z_j} e^{-\mathrm{i}(k_x x_j - k_y y_j)}. \end{aligned} \tag{41}$$

The forces can be obtained by simple differentiation since the sums are absolutely convergent. Although Eqn.(41) has a much better convergence than the original form in Eqn.(39), its main advantage is a linear computation time with respect to the number of particles N, since the two sums over N can be evaluated independently. Just as for the classical Ewald method, parallelization is trivial, since we only need to calculate simple sums over all compute nodes, for which efficient reduce operations exist in the MPI standard.

To complete the calculation, one of course still needs the forces and energies of the fully periodic system, for example calculated by the P^{3}M method. However, it is *vital* to use the correct summation order, that is, dipole term. Since we performed the summation over the image layers explicitely layer by layer, also the 3D summation has to be performed in the same order. This means, that one has to use the dipole term according to Eqn. (10), just as in the method proposed by Yeh and Berkowitz.

Similar to P^{3}M it is possible to consider non-neutral systems[37]. To this aim, one first neutralizes the system using a homogeneous background for the calculation using the 3D method, e.g. P^{3}M . This homogeneous background gives rise to an unwanted force that drives the particles towards the boundaries of the system. However, one can calculate this force analytically, and subtract it again. Note that the background also contributes to the dipole term for slabwise summation. It should now read:

$$U^{(d)} = \frac{2\pi}{V} \left[\sum_{i=1}^{N} q_i \left(z_i - \frac{L_z}{2} \right) \right]^2, \tag{42}$$

where the shift accounts for the neutralizing background, which in the slab system is a homogeneous charge in the range from 0 to L_z. For an Ewald-type method, the neutralizing background term $U^{(n)}$ according to Eqn. (11) is required in addition. The interaction energy of the background with the slab system and itself is finally

$$U^{(b)} = \frac{2\pi}{V} \left[\sum_{i=1}^{N} q_i \sum_{j=1}^{N} q_j z_j \left(L_z - z_j \right) - \frac{1}{3} \left(\sum_{i=1}^{N} q_i \right)^2 \right], \tag{43}$$

which we need to subtract together with the ELC correction.

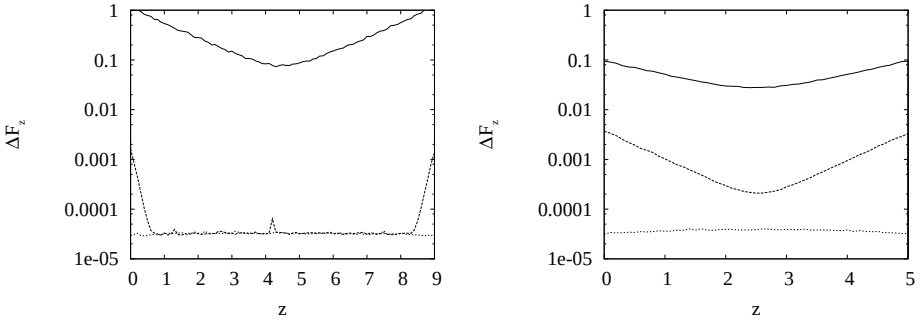


Figure 4: Absolute RMS force error ΔF_z as a function of the particle's z–coordinate for a homogeneous random system of 1000 particles in a box of size $10 \times 10 \times h$ with a system height $h = 9$ (left graph) and 5 (right graph). The curves are from top to bottom: results for cutoff $R = 0$, i. e. the Yeh and Berkowitz approach, for $R = 1$ and $R = 1.8$ in the left graph with $h = 9$. In the right graph with $h = 5$, we also use $R = 0$, but then $R = 0.1$ and $R = 0.3$. These cutoffs correspond to similar boundary errors for both heights, showing that one needs larger cutoffs with decreasing gap size. We used P^{3}M as the underlying 3D method, tuned for a RMS force error of 10^{-4}. The overall accuracy is limited by this P^{3}M error, the accuracy of ELC in the center of the system is much better.

4.1 Error estimates

Like before, we want to estimate the error that we introduce by the necessary truncation of the Fourier sum. While for the standard Ewald sum, this was only used to tune the algorithm, the error estimates for ELC can also be used to estimate the error that one introduces by not taking into account the ELC term at all, that is, using the slab-wise method of Yeh and Berkowitz[11]. Our error estimates can be used to determine "a priori" the necessary gap size to reach a preset precision without calculating the ELC term. Therefore we also have to deal with small cutoffs, especially the case when no terms of E_{lc} are added.

We assume that the Fourier sum is truncated at a cutoff of $k_{\max}$, that is, the summation is performed only over all $(p, q) \in (2\pi L\mathbb{Z})^2$, where $0 < p^2 + q^2 \leq k_{\max}^2$. An upper bound for the absolute value of the summands is

$$\frac{2\pi}{L^2} \left| \frac{\cosh(|\mathbf{k}|z_{ij})}{|\mathbf{k}|(e^{|\mathbf{k}|L_z} - 1)} e^{\mathrm{i}k_x x_{ij}} e^{\mathrm{i}k_y y_{ij}} \right| \leq \frac{2\pi}{L^2} e^{-2\pi|\mathbf{k}|L_z} \frac{\cosh(|\mathbf{k}|z_{ij})}{|\mathbf{k}|(1 - e^{-|\mathbf{k}|L_z})}. \tag{44}$$

The sum over all these upperbounds can then be performed by an careful approximation of the sum by an integral, which will give an *upper bound* for the algorithm dependent factor χ of Eqn. (22). From that, we obtain a limit for the RMS force error:

$$\begin{aligned} \Delta F \leq \frac{\sum q_i^2}{2\sqrt{N}(e^{k_{\max}L_z} - 1)} \Bigg[& \left(k_{\max} + \frac{4}{L} + \frac{1}{L_z - h} \right) \frac{e^{k_{\max}h}}{(L_z - h)} + \\ & \left(k_{\max} + \frac{4}{L} + \frac{1}{L_z + h} \right) \frac{e^{-k_{\max}h}}{(L_z + h)} \Bigg] \end{aligned} \tag{45}$$

$$\approx \frac{\sum q_i^2 k_{\max}}{2\sqrt{N}(L_z - h)} e^{-k_{\max}(L_z - h)}, \tag{46}$$

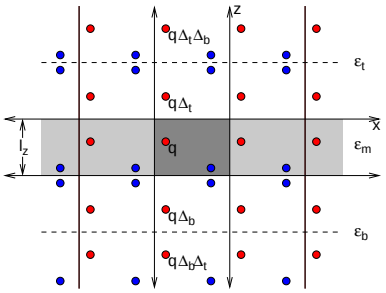


Figure 5: A schematic view of image charges. The dielectric interfaces are characterized by the $\varepsilon_t - \varepsilon_m$ and $\varepsilon_m - \varepsilon_b$ boundaries, which give rise to an infinite number of image charges along the z-direction due to polarization and multiple reflection under two parallel dielectrics. In the x, y-direction the image charges due to periodic boundaries are shown. The dotted lines are only provided to visualize the positioning of the image charges.

where the latter simplification is only valid for large $k_{\max}$. The total error of the combination of ELC and any 3D method can be obtained again by summing up the error contributions according to Eqn. (23).

Our error estimates show that the error drops exponential both with $k_{\max}$ and L_z. This means that it is easy to achieve increase the accuracy with our layer correction formula, either by increasing the k-space cutoff or the gap size. An upper bound for the error produced by the method of Yeh and Berkowitz can be obtained by inserting $k_{\max} = 2\pi\frac{1}{L}$. In this case, the error formula shows that the error decays with the ratio $k_{\max}(L_z - h) \sim (L_z - h)/L$ and *not* L_z/h, as was claimed by Yeh and Berkowitz[b].

Note that Eqn. (44) shows that the error in the potential or the force for a single particle will be largest if it is located near the gap, since there $|z_{ij}|$ will be maximal. This effect will increase with increasing R, therefore, it is important to tune ELC (and with that, also the method of Yeh and Berkowitz) to errors small enough to be negligible even at the system boundaries. Just measuring the overall RMS force error might be misleading, especially, since for thin films or surfaces, mostly the particles at the surfaces are of interest. This is demonstrated in Fig 4.

If one assumes an mesh-based Ewald method to be used for the 3D periodic system, one can estimate the optimal L_z as follows. The computation time of the mesh method is proportional to $V = L^2 L_z$. For ELC, $k_{\max}$ is proportional to $1/(L_z - h)$, therefore the computation time spent with ELC is proportional to $L^2/(L_z - h)^2$. The total computation time is therefore minimized by a constant gap size independent of the box dimensions.

4.2 Varying dielectric constants

In the three dimensional Ewald sum, the dielectric constant at infinity, ϵ', plays an important role through the dipole term (9). This is of physical origin, as can be seen from the fact that the dipole term is not algorithm dependent. For systems with only two periodic dimensions, the dielectric constant at infinity does not play a role; however, it may be necessary to include dielectric contrasts in the non-periodic coordinate. For example, a thin

[b] In fact, already the extreme case of $h = 0$ immediately shows that this cannot be true, as a conventional method for three dimensional periodicity will not deliver exact results for a purely planar system.

water film in vacuum has a dielectric constant of $\epsilon \approx 80$, while the surrounding vacuum has $\epsilon = 1$. The dielectric contrasts at the film boundaries give rise to an infinite series of image charges, which can be taken into account by the ELC method[37].

We assume that the charges are all contained in a medium of height L_z and a dielectric constant ε_m, which is sandwiched from above by a medium of dielectric constant ε_t and from below by a medium of dielectric constant ε_b (compare Fig. 5 left). We define the dielectric contrasts

$$\Delta_b = \frac{\varepsilon_m - \varepsilon_b}{\varepsilon_m + \varepsilon_b}, \quad \text{and} \quad \Delta_t = \frac{\varepsilon_m - \varepsilon_t}{\varepsilon_m + \varepsilon_t}, \tag{47}$$

and the double reflection factor $\Delta = \Delta_b \Delta_t$. Then the potential of each charge including its polarization images is represented by the following four series of charges:

- charges $q_i \Delta^n$ at positions $z_i - 2L_z n$, $n \geq 0$ (this series also includes the real charge)
- charges $q_i \Delta_b \Delta^n$ at positions $-z_i - 2L_z n$, $n \geq 0$
- charges $q_i \Delta^n$ at positions $z_i + 2L_z n$, $n \geq 1$
- charges $q_i \Delta_t \Delta^n$ at positions $-z_i + 2L_z(n+1)$, $n \geq 0$.

What we are interested in is the electrostatic energy, that is, the energy of the real charges due to the potential generated by (a) the real particles, (b) the image charges due to the dielectric boundaries and (c) the periodic images of both.

Even though the task looks daunting, the solution turns out to be simple. Basically, the sum of the potential generated by each of the charge series is again a geometric series in the image layer count, which once more can be calculated by Eqn. (40) and summed up analytically. One just has to be careful with the images that are close to the primary layer; here, a direct summation, that is, the application of for example ELC+P^{3}M is required. For further details, see Ref. 37.

5 Dipolar Ewald Summation in 3D

In the previous section we treated methods to deal with the Coulomb interaction, or more generally, interactions which vary with $1/r$. In this section, we turn towards another electrostatic interaction, namely the dipolar interaction. For (fully) periodic geometries again the Ewald method can be used, with the same computation time scale as for the Coulomb interaction, that is at best $\mathcal{O}(N^{3/2})$, if the cutoff is optimally varied with the splitting parameter[5]. For MC simulations, knowing the energy formulas is sufficient, whereas for MD simulations, we need to know forces and torques. In this section we show how the Ewald sum can be used also for dipolar interactions, and give a reliable error estimate for the forces and the torques[15].

Consider a system of N particles with a point-dipole $\boldsymbol{\mu}_i$ at their center position $\mathbf{r}_i$ in a cubic simulation box of length L. If periodic boundary conditions are applied, the total electrostatic energy of the box is given by

$$U = \frac{1}{2} \sum_{\mathbf{m} \in \mathbb{Z}^3} {\sum_{i,j=1}^{N}}' \left\{ \frac{\boldsymbol{\mu}_i \cdot \boldsymbol{\mu}_j}{|\mathbf{r}_{ij} + \mathbf{m}L|^3} - \frac{3[\boldsymbol{\mu}_i \cdot (\mathbf{r}_{ij} + \mathbf{m}L)][\boldsymbol{\mu}_j \cdot (\mathbf{r}_{ij} + \mathbf{m}L)]}{|\mathbf{r}_{ij} + \mathbf{m}L|^5} \right\}, \tag{48}$$

where the prime indicates that the $i = j$ term must be omitted for $\mathbf{m} = 0$. The slowly decaying long range part of the dipolar potential makes the straightforward summation of Eqn. (48) too time consuming. The Ewald trick splits the problem again into two rapidly convergent parts, one in real space and one in reciprocal space. The details of the method are discussed in Refs.[1,5,19,3], here we only give the final expressions. The energy is

$$U = U^{(r)} + U^{(k)} + U^{(s)} + U^{(d)}, \tag{49}$$

where the real-space $U^{(r)}$, the k-space (reciprocal space) $U^{(k)}$, the self $U^{(s)}$ and the dipole (surface) $U^{(d)}$ contributions are respectively given by:

$$\begin{aligned} U^{(r)} = \frac{1}{2} \sum_{\mathbf{m}\in\mathbb{Z}^3} \sum_{i,j=1}^{N}{}' \Big\{ & (\boldsymbol{\mu}_i \cdot \boldsymbol{\mu}_j) B(|\mathbf{r}_{ij} + \mathbf{m}L|) \\ & - [\boldsymbol{\mu}_i \cdot (\mathbf{r}_{ij} + \mathbf{m}L)][\boldsymbol{\mu}_j \cdot (\mathbf{r}_{ij} + \mathbf{m}L)] C(|\mathbf{r}_{ij} + \mathbf{m}L|) \Big\}, \end{aligned} \tag{50}$$

$$U^{(k)} = \frac{1}{2L^3} \sum_{\mathbf{k}\neq 0} \frac{4\pi}{k^2} e^{-(\pi k/\alpha L)^2} \sum_{i,j=1}^{N} (\boldsymbol{\mu}_i \cdot \mathbf{k})(\boldsymbol{\mu}_j \cdot \mathbf{k}) e^{2\pi i \mathbf{k}\cdot\mathbf{r}_{ij}/L}, \tag{51}$$

$$U^{(s)} = -\frac{2\alpha^3}{3\sqrt{\pi}} \sum_{i=1}^{N} \mu_i^2, \tag{52}$$

$$U^{(d)} = \frac{2\pi}{(2\epsilon' + 1)L^3} \sum_{i,j=1}^{N} \boldsymbol{\mu}_i \cdot \boldsymbol{\mu}_j. \tag{53}$$

The $\mathbf{k}$-sum is again over reciprocal vectors $\mathbf{k} \in \frac{2\pi}{L}\mathbb{Z}^3$, the sums over i and j are for the particles in the central box and

$$B(r) = [\operatorname{erfc}(\alpha r) + (2\alpha r/\sqrt{\pi}) \exp(-\alpha^2 r^2)]/r^3, \tag{54}$$

$$C(r) = [3 \operatorname{erfc}(\alpha r) + (2\alpha r/\sqrt{\pi})(3 + 2\alpha^2 r^2) \exp(-\alpha^2 r^2)]/r^5. \tag{55}$$

The inverse length α is the splitting parameter of the Ewald summation which should be chosen so as to optimize the performance. The form Eqn. (53) for the surface term again assumes spherical summation and that the medium outside the growing sphere is an uniform dielectric with dielectric constant ϵ'.

In practical calculations, the infinite sums in Eqns. (50) and (51) are truncated by only taking into account distances which are smaller than some real space cutoff $r_{\max}$ and wave vectors with a modulus smaller than some reciprocal space cutoff $k_{\max}$. If $r_{\max} \leq L/2$, the sum in real space [Eqn. (50)] reduces to the normal minimum image convention. The double sum over particles in $U^{(k)}$ can be replaced by a product of two single sums which is more suitable for numerical calculations.

The force $\mathbf{F}_i$ acting on particle i is obtained by differentiating the non-constant contributions to the potential energy U with respect to $\mathbf{r}_i$, i.e.,

$$\mathbf{F}_i = -\frac{\partial}{\partial \mathbf{r}_i} U = \mathbf{F}_i^{(r)} + \mathbf{F}_i^{(k)}, \tag{56}$$

with the real-space and k-space contributions given by:

$$\mathbf{F}_i^{(r)} = \sum_{\mathbf{m}\in\mathbb{Z}^3} \sum_{j=1}^{N}{}' \Bigg\{ \Big[(\boldsymbol{\mu}_i\cdot\boldsymbol{\mu}_j)\mathbf{r}_{ij}^{\mathbf{m}} + \boldsymbol{\mu}_i(\boldsymbol{\mu}_j\cdot\mathbf{r}_{ij}^{\mathbf{m}}) + (\boldsymbol{\mu}_i\cdot\mathbf{r}_{ij}^{\mathbf{m}})\boldsymbol{\mu}_j\Big] C(r_{ij}^{\mathbf{m}}) - (\boldsymbol{\mu}_i\cdot\mathbf{r}_{ij}^{\mathbf{m}})(\boldsymbol{\mu}_j\cdot\mathbf{r}_{ij}^{\mathbf{m}}) D(r_{ij}^{\mathbf{m}})\,\mathbf{r}_{ij}^{\mathbf{m}} \Bigg\}, \tag{57}$$

$$\mathbf{F}_i^{(k)} = \frac{2\pi}{L^4}\sum_{j=1}^{N}\sum_{\mathbf{k}\neq 0} \frac{4\pi\mathbf{k}}{k^2}\Big[(\boldsymbol{\mu}_i\cdot\mathbf{k})(\boldsymbol{\mu}_j\cdot\mathbf{k})\exp[-(\pi k/\alpha L)^2] \cdot\sin(2\pi\mathbf{k}\cdot\mathbf{r}_{ij}/L)\Big], \tag{58}$$

where $r_{ij}^{\mathbf{m}} = \mathbf{r}_{ij} + \mathbf{m}L$ and

$$D(r) = [15\,\mathrm{erfc}(\alpha r) + (2\alpha r/\sqrt{\pi})(15 + 10\,\alpha^2 r^2 + 4\,\alpha^4 r^4)\exp(-\alpha^2 r^2)]/r^7. \tag{59}$$

Since the self and surface energy terms [Eqns. (52), (53)] are independent of the particle positions, they have no contributions to the force, unlike the Ewald sum for Coulomb systems where the surface term contributes. The torque $\boldsymbol{\tau}_i$ acting on particle i is related to the electrostatic field $\mathbf{E}_i$ at the location of this particle via

$$\boldsymbol{\tau}_i = \boldsymbol{\mu}_i \times \mathbf{E}_i = \boldsymbol{\tau}_i^{(r)} + \boldsymbol{\tau}_i^{(k)} + \boldsymbol{\tau}_i^{(d)}, \tag{60}$$

with

$$\mathbf{E}_i = -\frac{\partial}{\partial\boldsymbol{\mu}_i}U, \tag{61}$$

and thus

$$\boldsymbol{\tau}_i^{(r)} = -\sum_{\mathbf{m}\in\mathbb{Z}^3}\sum_{j=1}^{N}{}' \Big\{(\boldsymbol{\mu}_i\times\boldsymbol{\mu}_j)B(r_{ij}^{\mathbf{m}}) - (\boldsymbol{\mu}_i\times\mathbf{r}_{ij}^{\mathbf{m}})(\boldsymbol{\mu}_j\cdot\mathbf{r}_{ij}^{\mathbf{m}})C(r_{ij}^{\mathbf{m}})\Big\}, \tag{62}$$

$$\boldsymbol{\tau}_i^{(k)} = -\frac{1}{L^3}\sum_{j=1}^{N}\sum_{\mathbf{k}\in\mathbb{Z}^3,\mathbf{k}\neq 0}\frac{4\pi}{k^2}(\boldsymbol{\mu}_i\times\mathbf{k})(\boldsymbol{\mu}_j\cdot\mathbf{k})e^{-(\pi k/\alpha L)^2}e^{2\pi i\mathbf{k}\cdot\mathbf{r}_{ij}/L}, \tag{63}$$

$$\boldsymbol{\tau}_i^{(d)} = -\frac{4\pi}{(2\epsilon'+1)L^3}\sum_{j=1}^{N}\boldsymbol{\mu}_i\times\boldsymbol{\mu}_j. \tag{64}$$

5.1 Error Formulas

We now give estimates for the RMS error caused by cutting off the Ewald summation in real-space and k-space for the forces and the torques. There are no errors involved in the self and surface contributions [Eqns. (52), (53) and (64)], because no cutoff operations are applied to them. As can be shown similarly to the way as it was presented in Sec. 2.1, the RMS error for the force can be cast in the following form

$$\Delta F \approx \chi\frac{\mathcal{M}^2}{\sqrt{N}} \quad \text{with} \quad \mathcal{M}^2 := \sum_{j=1}^{N}\mu_j^2. \tag{65}$$

After some lengthy calculation one obtains for the algorithmic contribution from the real space[15]

$$\chi^{(r)2} \approx L^{-3} r_{\max}^{-9} \alpha^{-4} (\frac{13}{6} C_c^2 + \frac{2}{15} D_c^2 - \frac{13}{15} C_c D_c) \exp(-2\alpha^2 r_{\max}^2), \tag{66}$$

where the terms C_c and D_c are given by

$$C_c = 4\,\alpha^4 r_{\max}^4 + 6\,\alpha^2 r_{\max}^2 + 3, \tag{67}$$

$$D_c = 8\,\alpha^6 r_{\max}^6 + 20\,\alpha^4 r_{\max}^4 + 30\,\alpha^2 r_{\max}^2 + 15. \tag{68}$$

The resulting RMS expectation of the real-space cutoff error in the forces is thus

$$\Delta F^{(r)} \approx \mathcal{M}^2 (L^3 \alpha^4 r_{\max}^9 N)^{-1/2} (\frac{13}{6} C_c^2 + \frac{2}{15} D_c^2 - \frac{13}{15} C_c D_c)^{1/2} \exp(-\alpha^2 r_{\max}^2) \tag{69}$$

and in the torques

$$\Delta \tau^{(r)} \approx \mathcal{M}^2 (L^3 \alpha^4 r_{\max}^7 N)^{-1/2} [\frac{1}{2} B_c^2 + \frac{1}{5} C_c^2]^{1/2} \exp(-\alpha^2 r_{\max}^2). \tag{70}$$

Eqns. (69) and (70) both contain the exponential $\exp(-\alpha^2 r_{\max}^2)$. For sufficiently low errors, $\alpha r_{\max}$ has to be larger than one, for example $\alpha r_{\max} \approx \pi$ for an error of $\exp(-\pi^2) \approx 5 \times 10^{-5}$. If only the highest powers of $\alpha r_{\max}$ are retained, the estimates of drastically simplify to

$$\Delta F^{(r)} \approx 8 \mathcal{M}^2 \alpha^4 (2 r_{\max}^3 / 15 N L^3)^{1/2} \exp(-\alpha^2 r_{\max}^2), \tag{71}$$

$$\Delta \tau^{(r)} \approx 4 \mathcal{M}^2 \alpha^2 (r_{\max} / 5 N L^3)^{1/2} \exp(-\alpha^2 r_{\max}^2), \tag{72}$$

The advantage of these simplified formulas is that they reflect the dependence of the RMS errors on α and $r_{\max}$ more directly and thus could be used more easily in determining the optimal values of these parameters.

In deriving the estimates of the reciprocal-space (k-space) cutoff errors, we assume that the radial distribution function of the particles is approximately unity at all distances. Then one finds the RMS expectation of the k-space cutoff error in the forces as[15]

$$\Delta F^{(k)} \approx 8\pi \mathcal{M}^2 L^{-3} \alpha (2\pi k_{\max}^3 / 15 N)^{1/2} \exp[-(\pi k_{\max} / \alpha L)^2], \tag{73}$$

and in the torques

$$\Delta \tau^{(k)} \approx 4 \mathcal{M}^2 L^{-2} \alpha (\pi k_{\max} / 5 N)^{1/2} \exp[-(\pi k_{\max} / \alpha L)^2]. \tag{74}$$

The total error can be obtained again by taking into account the real and k-space errors according to Eqn. (23).

5.2 Optimization of Parameters

In this section, we discuss the use of the analytical formulas derived in Sec. III to determine the optimal values of α, $r_{\max}$ and $k_{\max}$ by which the required accuracy could be satisfied and the computation time is minimized. The detailed discussions on this subject can also be found in Refs.[5,27]. First, we note that the same argumentation as for the Coulombic Ewald sum holds, namely that the computation time is $\mathcal{T} = \mathcal{O}(N\alpha^{-3}) + \mathcal{O}(N^2\alpha^3)$, which is minimized by $\alpha \sim N^{-1/6}$. Therefore, the optimal computation time of the dipolar Ewald sum is as well $\mathcal{O}(N^{3/2})$.

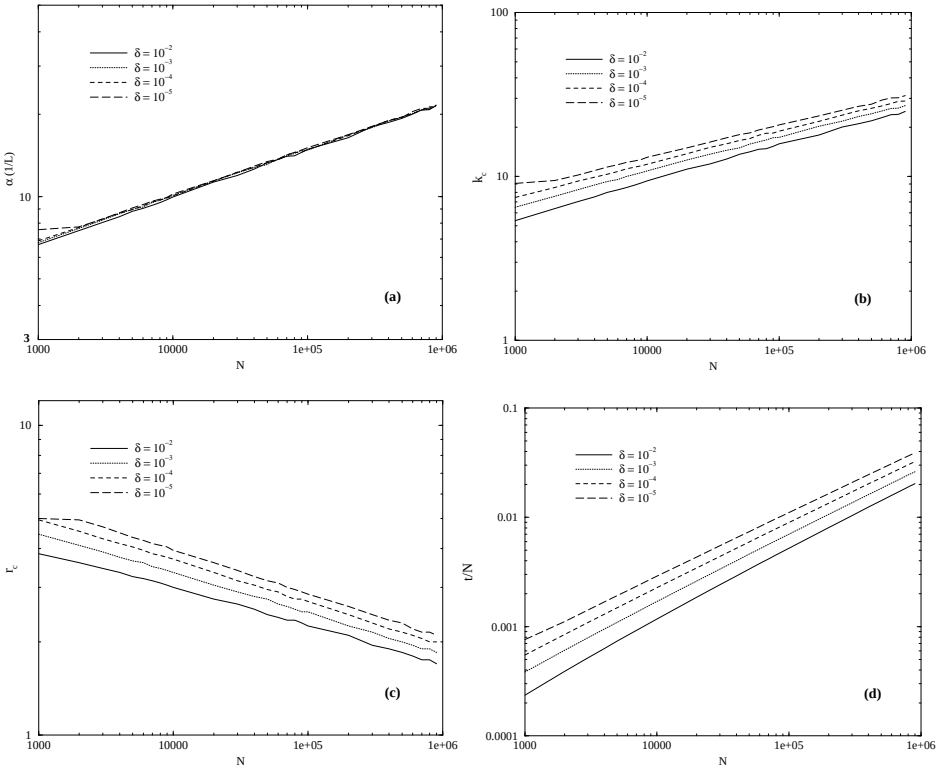


Figure 6: Optimal values of the parameters α (a), $k_{\max}$ (b) and $r_{\max}$ (c) as well as the corresponding minimized computation time $\mathcal{T}/N$ (d) as a function of the number of particles.

The numerical investigation of the functional dependence of the optimal parameters on N and δ are performed as follows. For each given N and δ, we at first choose different values for $r_{\max}$ within the inequality $r_{\max} \leq L/2$. For each $r_{\max}$ the parameters α and $k_{\max}$ are calculated by solving Eqns. (71)) and (73). Now, one can perform a trial computation to measure the required computer time for each $r_{\max}$, and by this figure out the optimal value of $r_{\max}$ which gives the minimum computation time. In our calculations the size of the simulation cell is fixed to a dimensionless length of $L = 10$. The range of accuracy requirement and number of particles are chosen to be $\delta = 10^{-2}$ to 10^{-5} measuring in $\mathcal{P}^2/\mathcal{L}^4$ and $N = 10^3$ to $N = 10^6$ which should cover most of the applications. The particles are supposed to have an uniform dipole moment of $\mathcal{P}$. The results for the optimal values of the parameters and the corresponding computation time per particle are shown in Fig. 6 (a-d), respectively. It can be clearly seen that the functional dependence of the parameters and the overall computation time on N are just as discussed above. Fig. 6(c) shows that when a high accuracy is required for a system with a small number of particles, the predicted real-space cutoff is larger than half of the box length and $r_{\max} = L/2$ must be used, which prohibits the use of the minimum image convention. The optimal α values hardly depend on the accuracies. Note that unlike in the discussion above, we assume a

constant volume and thus the optimal $\alpha \sim N^{1/6}$ instead of $N^{-1/6}$, as can be easily seen; the overall optimal computation time at constant volume is still $\mathcal{O}(N^{3/2})$.

Finally we should remark that besides the standard Ewald method, dipolar variants of basically all Coulomb solvers are possible; a dipolar SPME has been described in Ref. 38, a dipolar P^3M with optimal influence function and known error estimates in Ref. 39, and the ELC term was derived for dipolar systems in Ref. 40.

6 Concluding Remarks

We have given a review of Fourier transform-based methods to compute the electrostatic interaction. We have shown that this oldest family of Coulomb solvers is very versatile and can be applied to systems with any combination of periodic boundary conditions and also to dipolar interactions. We tried to give reasons why one should worry about errors, and gave error estimates for all presented algorithms. We also described how these error estimates can be used to tune the algorithms to perform optimally for speed and accuracy.

References

1. P. P. Ewald, *Die Berechnung optischer und elektrostatischer Gitterpotentiale*, Ann. Phys., **369**, no. 3, 253–287, 1921.
2. D. M. Heyes, *Electrostatic potentials and fields in infinite point charge lattices*, J. Chem. Phys., **74**, no. 3, 1924–1929, 1981.
3. 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. Lond., A, **373**, no. 1752, 27–56, 1980.
4. 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. Lond., A, **373**, no. 1752, 57–66, 1980.
5. John W. Perram, Henrik G. Petersen, and Simon 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., **65**, no. 4, 875–893, 1988.
6. M. Deserno and C. Holm, *How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines*, J. Chem. Phys., **109**, 7678, 1998.
7. M. Deserno and C. Holm, *How to mesh up Ewald sums. II. An accurate error estimate for the Particle-Particle-Particle-Mesh algorithm*, J. Chem. Phys., **109**, 7694, 1998.
8. Albert H. Widmann and David B. Adolf, *A comparison of Ewald summation techniques for planar surfaces*, Comput. Phys. Commun., **107**, 167–186, 1997.
9. J. C. Shelley and G. N. Patey, *Boundary condition effects in simulations of water confined between planar walls*, Mol. Phys., **88**, 385, 1996.
10. E. Spohr, *Effect of electrostatic boundary conditions and system size on the interfacial properties of water and equeous solutions*, J. Chem. Phys., **107**, 6342, 1997.
11. In-Chul Yeh and Max L. Berkowitz, *Ewald summation for systems with slab geometry*, J. Chem. Phys., **111**, no. 7, 3155–3162, 1999.
12. A. Arnold, J. de Joannis, and C. Holm, *Electrostatics in Periodic Slab Geometries I*, J. Chem. Phys., **117**, 2496–2502, 2002.

13. A. Arnold, J. de Joannis, and C. Holm, *Electrostatics in Periodic Slab Geometries II*, J. Chem. Phys., **117**, 2503–2512, 2002.
14. R. E. Rosensweig, *Ferrohydrodynamics*, Cambridge Univ. Press, Cambridge, 1985.
15. Z. W. Wang and C. Holm, *Estimate of the Cutoff Errors in the Ewald Summation for Dipolar Systems*, J. Chem. Phys., **115**, 6277–6798, 2001.
16. C. Holm, "Efficient methods for long range interactions in periodic geometries plus one application", in: Computational Soft Matter: From Synthetic Polymers to Proteins, N. Attig, K. Binder, H. Grubmüller, and K. Kremer, (Eds.), vol. 23 of *NIC series*. Research Centre Jülich, 2004.
17. A. Arnold and C. Holm, "Efficient methods to compute long range interactions for soft matter systems", in: Advanced Computer Simulation Approaches for Soft Matter Sciences II, C. Holm and K. Kremer, (Eds.), vol. II of *Advances in Polymer Sciences*, pp. 59–109. Springer, Berlin, 2005.
18. D. Frenkel, *Colloidal systems: Playing Tricks with Designer "Atoms"*, Science, **296**, 65, 2002.
19. Mike P. Allen and Dominik J. Tildesley, *Computer Simulation of Liquids*, Oxford Science Publications. Clarendon Press, Oxford, 1 edition, 1987.
20. Jean-Michel Caillol, *Comments on the numerical simulation of electrolytes in periodic boundary conditions*, J. Chem. Phys., **101**, no. 7, 6080–6090, 1994.
21. Stefan Boresch and Othmar Steinhauser, *Presumed Versus Real Artifacts of the Ewald Summation Technique: The Importance of Dielectric Boundary Conditions*, Ber. Bunsenges. Phys. Chem., **101**, no. 7, 1019–29, 1997.
22. H. J. C. Berendsen, in: Computer Simulation of Biomolecular Systems, Wilfred F. van Gunsteren, P. K. Weiner, and A. J. Wilkinson, (Eds.), vol. 2, pp. 161–81, ESCOM, The Netherlands, 1993.
23. Philippe H. Huenenberger, *Optimal charger-shaping functions for the particle-particle-particle-mesh (P3M) method for computing electrostatic interactions in molecular simulations*, J. Chem. Phys., **113**, no. 23, 10464–10476, 2000.
24. E. R. Smith, *Electrostatic potentials for thin layers*, Mol. Phys., **65**, 1089–1104, 1988.
25. Gerhard Hummer, Lawrence R. Pratt, and Angle E. García, *Molecular Theories and Simulation of Ions and Polar Molecules in Water*, J. Phys. Chem. A, **102**, no. 41, 7885–97, 1998.
26. Jiri Kolafa and John W. Perram, *Cutoff errors in the Ewald summation formulae for point charge systems*, Mol. Simul., **9**, no. 5, 351–368, 1992.
27. Henrik G. Petersen, *Accuracy and efficiency of the particle mesh Ewald method*, J. Chem. Phys., **103**, no. 9, 3668–3679, 1995.
28. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, IOP, London, 1988.
29. T. Darden, D. York, and L. Pedersen, *Particle Mesh Ewald: An $N\log(N)$ method for Ewald sums in large systems*, J. Chem. Phys., **98**, 10089–10092, 1993.
30. U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. Pedersen, *A smooth Particle Mesh Ewald method*, J. Chem. Phys., **103**, 8577, 1995.
31. I. J. Schoenberg, *Cardinal Spline Interpolation*, Society for Industrial and Applied Mathematics, Philadelphia, 1973.
32. David S. Cerutti, Robert E. Duke, Thomas A. Darden, and Terry P. Lybrand, *Staggered Mesh Ewald: An Extension of the Smooth Particle-Mesh Ewald Method Adding Great Versatility*, J. Chem. Theory Comput., **5**, no. 9, 2322–2338, 2009.

33. A. Neelov and C. Holm, *Interlaced P3M algorithm with analytical and ik-differentiation*, J. Chem. Phys., **132**, no. 23, 234103, 2010.
34. Matteo Frigo and Steven G. Johnson, *The Design and Implementation of FFTW3*, Proceedings of the IEEE, **93**, no. 2, 216–231, 2005, Special issue on "Program Generation, Optimization, and Platform Adaptation".
35. A. Arnold and C. Holm, *MMM2D: A fast and accurate summation method for electrostatic interactions in 2D slab geometries*, Comput. Phys. Commun., **148**, no. 3, 327–348, 2002.
36. A. Arnold and C. Holm, *A novel method for calculating electrostatic interactions in 2D periodic slab geometries*, Chem. Phys. Lett., **354**, 324–330, 2002.
37. Sandeep Tyagi, Axel Arnold, and Christian Holm, *Electrostatic layer correction with image charges: A linear scaling method to treat slab 2D + h systems with dielectric interfaces*, J. Chem. Phys., **129**, no. 20, 204102, 2008.
38. Abdulnour Toukmaji, Celeste Sagui, John Board, and Tom Darden, *Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions*, J. Chem. Phys., **113**, no. 24, 10913–10927, 2000.
39. Juan J. Cerdà, V. Ballenegger, O. Lenz, and Ch. Holm, *P3M algorithm for dipolar interactions.*, J. Chem. Phys., **129**, 234104, 2008.
40. A. Bródka, *Ewald summation method with electrostatic layer correction for interactions of point dipoles in slab geometry*, Chem. Phys. Lett., **400**, no. 1-3, 62–67, 2004.

Parallel Tree Codes

Paul Gibbon, Robert Speck, Lukas Arnold, Mathias Winkel, and Helge Hübner

Institute for Advanced Simulation
Jülich Supercomputing Centre
Forschungszentrum Jülich, 52425 Jülich, Germany
E-mail: p.gibbon@fz-juelich.de

A highly scalable parallel tree code (PEPC) for rapid computation of long-range $(1/r)$ Coulomb forces is presented. It can be used as a library for applications involving electrostatics or Newtonian gravity in 3D. The code is based on the Hashed-Oct-Tree algorithm, in which particle coordinates are encoded on a space-filling curve prior to sorting and allocation to processor domains. An overview is given of the main components of this code: domain decomposition, tree construction and execution of traversals necessary to perform the force summation. Particular bottlenecks which can impair parallel performance are the sorting routines and the exchange of multipole information between processor tasks. These are analysed with a view to scaling the code on petaflop supercomputer systems.

1 Introduction

Even in the era of exascale computing, the naive approach of solving the N-body problem directly by an $\mathcal{O}(N^2)$-algorithm is still impractical for the vast majority of physical systems. Despite the high accuracy and scalability of these algorithms they are highly ineffective for problems where statistically significant results can only be obtained by simulating the presence of more than a few thousand particles.

In the mid-1980s two techniques – the hierarchical Tree Code[1] and the Fast Multipole Method (FMM)[2], with respective algorithmic scalings of $O(N \log N)$ and $O(N)$ – have revolutionized long-range N-body simulation for scientists across a wide range of disciplines[3]. These methods reduce the number of direct particle-particle interactions through the systematic use of multipole expansions up to a given degree. For many dynamical systems, there is no need to compute potentials and forces to higher accuracy than the error incurred by integrating the equations of motion. In such cases tree codes and the FMM make it possible to perform significantly fast simulations with many millions of particles (see also[4]). A further advantage is that these methods are mesh-free: they do not depend on structured grids for the field solver and are therefore intrinsically adaptive.

Over the past few years at JSC we have developed the parallel tree code PEPC – Pretty Efficient Parallel Coulomb-solver, demonstrating the performance and scalability on the former Jülich IBM p690 and BlueGene/L machines[5–7]. This highly portable code was initially designed for mesh-free modelling of complex plasma systems[8], but has since been adapted for gravitational problems and vortex fluid methods, the latter utilising potentials differing from $1 - r$. Based on the original Warren-Salmon 'hashed oct-tree' scheme[9] with a fixed multipole expansion up to $p = 2$ (quadrupole), PEPC provides a flexible, high accuracy and fully parallelized tool for simulations of long-range N-body systems.

We begin this overview with a general outline of the parallel tree code PEPC, and then proceed to describe the main steps involved to compute the potentials and/or forces on a set of particles: the distribution and decomposition of the particles among processor tasks, the tree construction and multipole definitions, the determination of interaction lists and force summation, including a recent extension to periodic systems. In Section 3 we present performance analyses of the code on the Jülich supercomputers Jugene and Juropa.

2 Parallel Tree Codes: the Basics

Algorithmically speaking, an electrostatic tree code is no different to the various Newtonian gravity N-body solvers used in astrophysics, which are nearly all based on some form of the Barnes-Hut hierarchical tree algorithm[1]. Briefly summarized: the electrostatic force-sum on each particle is computed by systematically replacing more distant charges by multipole expansions of charge groups, thus reducing the standard $\mathcal{O}(N^2)$ direct sum to an $\mathcal{O}(N \log N)$ complexity at the price of a small, controllable error[3]. For many applications – for example in plasma or astrophysics – there is little to be gained in computing potentials and forces to higher accuracy than the error incurred by time-integration, which can be anywhere between 10^{-4} for a high-order Runge-Kutta scheme, to around 1% for the simple 2nd-order Leap-Frog method.

for all iteration steps **do**

domain decomposition:
convert local particle coordinates to keys
perform global key-sort

build tree:
construct local nodes
define global branches
fill in top-level nodes
compute multipole properties

define particle chunks
for each chunk **do**
perform *tree traversal*
perform *force summation*
end for

end for

Figure 1: Outline of parallel tree code: for dynamical systems, the tree and interaction lists are built afresh each timestep.

The price to be paid for the speed-up in the force sum is a considerable increase in programming complexity. First, the hierarchical data structure necessary to navigate around

the tree introduces an unavoidable bookkeeping overhead. Second, accessing multipole information lying on remote processors results in additional communication, which if not carefully implemented, can result in poor parallel efficiency. These issues notwithstanding, each step in the scheme illustrated in Fig.1 can be implemented in parallel with a scaling like N/P or $N \log N/P$.

2.1 Domain decomposition

At first sight, the hierarchical data structure required in a tree code to manage the multipole information would seem to preclude parallelism altogether. In fact, several schemes for parallel tree codes have been proposed and implemented, including virtual shared-memory versions[10], and distributed memory schemes using geometrical domain decomposition methods[11,12]. The scheme adopted here follows the one devised by Salmon and Warren[13,9], who practically reinvented the BH algorithm by scrapping memory pointers for bookkeeping in favour of a set of universal binary keys to represent particle and tree-node coordinates alike.

The basic idea is to convert the coordinate triple of each particle into a single, unique 64-bit integer key. The keys do not *replace* the coordinates, but as we shall see, provide a natural and rapid means of sorting the particles and building up the tree structure around them. Given its key and owner, locating any node in the tree is reduced to an $\mathcal{O}(1)$ operation.

In the present code, PEPC, the keys are constructed from the binary interleave operation:

$$k = p + \sum_{j=0}^{7} 8^j \left[4 \times \texttt{bit}(i_z, j, 1) + 2 \times \texttt{bit}(i_y, j, 1) + \texttt{bit}(i_x, j, 1)\right] \tag{1}$$

The function `bit(a,j,1)` selects 1 bit of the integer a starting from bit position j. The normalized integer coordinates are computed from:

$$\begin{aligned} i_x &= x/s \\ i_y &= y/s \\ i_z &= z/s \end{aligned} \tag{2}$$

where

$$s = L/2^{\texttt{nlevels}}$$

and L is the simulation box length; `nlevels` the maximum refinement level. The latter obviously depends on the machine precision, and for a 64-bit machine, we can have 21 bits per coordinate (or `nlevels=20`) plus a place-holder bit:

$$p = 2^{63}.$$

The place-holder bit is necessary to distinguish genuine particles on the lowest levels of the tree from higher level nodes of the internal tree structure. Mapping the coordinates to keys in this manner yields a space-filling curve, in this case following a pattern known as Morton- or Z-ordering This is by no means a unique choice: it is straightforward to modify the mapping to other curves such as the Hilbert (or Peano) ordering[14], yielding somewhat

improved data locality – Fig. 2. Both of these options are implemented in PEPC: choice of curve can in principle have an impact on communication effort at for very large processor numbers.

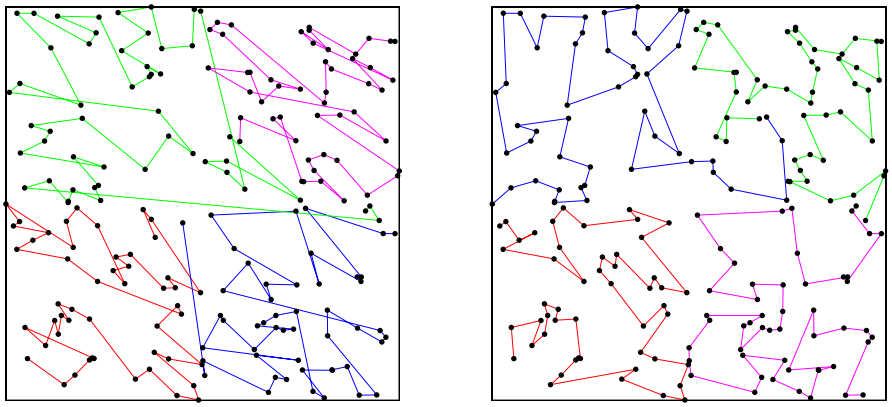


Figure 2: Domain decomposition of 200 simulation particles, shared equally among 4 tasks, using a space-filling curves with a) Z- or Morton ordering; b) Hilbert ordering

Domain decomposition can then be accomplished by cutting out equal portions of the sorted list and allocating these to the processors. A decomposition example for 200 particles divided among 4 processors is also seen in Fig. 2. Note that with this scheme, load balancing can be easily introduced by biasing the key-list segments according to the number of interactions computed for each particle in the force summation during the previous iteration.

After domain decomposition, the tasks have a set of local particles which are distinct except for the particles at task boundaries: here we copy the edge particles to the next task to ensure that we do not prematurely assign particles to leaf nodes when they may actually be part of shared twig nodes.

2.2 Parallel sorting on space-filling curves

An obvious prerequisite for well-connected task domains is that the particle keys are sorted, which in turn demands a fully parallel sorting algorithm. The one currently implemented in PEPC is an adaptation of the PSRS (parallel sorting by regular sampling) scheme[15], but with an important modification. Whereas the PSRS scheme assumes that the elements to be sorted are homogeneously spread over the computational domain limits, the particle key distribution generated by the coordinate interleave function in Eq. 1 is highly clustered, and in general cannot be sampled accurately without using a sampling frequency $\mathcal{O}(N)$ or greater, defeating the whole purpose of the parallel sort.

For this reason we have implemented a hierarchical version which performs *adaptive* sampling, resolving regions of higher key density with a recursive divide-and-conquer strategy. This enables the 'pivots' for the sort routine to be determined accurately, yielding

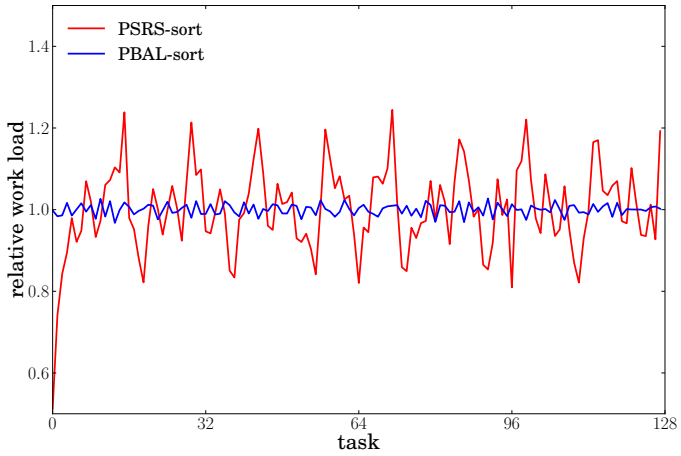


Figure 3: Relative work load per task for PSRS and PBAL-sort, applied to a medium size homogeneous example. PSRS: up to 40% imbalance, PBAL-sort: near-constant work load.

a balanced key distribution. In general, the particle count is weighted by the respective interaction list lengths from the previous iteration or timestep, thus allowing a form of downwind load-balancing which ensures roughly equal floating-point operations across tasks in the force summation step. Figure 3 shows the relative work load per task for PSRS compared to our adaptive sampling algorithm PBAL-sort, applied to a medium size homogeneous example. While PSRS already produces up to 40% imbalance across tasks, the hierarchical version can compensate this imbalance yielding a nearly constant load distribution.

However, even this improved adaptive sampling algorithm eventually reaches its limit on very higher core numbers, when the sampling resolution eventually becomes insufficient without using excessive statistics, and is also handicapped by $\mathcal{O}(P^2)$ collective operations during the merge step. For this reason it has been replaced by a more sophisticated parallel sort library developed at TU Chemnitz[16].

2.3 Construction of local trees

A big advantage of binary coordinate ordering over standard addressing techniques in tree codes is that the hierarchical structure is recovered automatically. As we will see shortly, keys of parent and neighbour cells are obtained by simple bit operations, so that the average access-time for any particle or node in the tree is $\mathcal{O}(1)$ instead of the usual $\mathcal{O}(\log N)$. The obvious drawback is that the number of possible keys, $2^{63} \simeq 10^{19}$ on a 64-bit machine, vastly exceeds the memory available, typically $\sim 10^5 - 10^6$ locations per processor. This mismatch is resolved by using a hashing function to map the key onto a physical address in memory, for example:

$$\texttt{address} = \texttt{k AND } (2^h - 1), \tag{3}$$

where h is the number of bits available for the address. This address then acts as a pointer to the particle or multipole properties. In case two or more keys give the same address (a 'collision'), a linked-list is constructed to resolve it. Clearly a high occurrence of collisions will ultimately degrade performance; however, as Warren & Salmon pointed out[9], the distribution of particles and nodes between many processors with their own address-spaces helps to reduce their number to a negligible level.

Once a set of particles has been allocated to a particular processor, and their associated properties (mass, charge, velocity etc.) have been fetched from their original location, one can immediately begin to construct the local trees. This can be done very efficiently because the particle keys implicitly contain the necessary information on all their ancestor nodes up to the root. The parent of a particle or twig-node is simply found by a 3-bit shift operation:

$$k_{\text{parent}} = \texttt{RIGHTSHIFT(k,3)} \tag{4}$$

Likewise, if a node's children are numbered from 0 to 7 (in a 3D oct-tree), their keys can be obtained by the inverse operation:

$$k_{\text{child}} = \texttt{LEFTSHIFT(k,3) OR child(0-7)}, \tag{5}$$

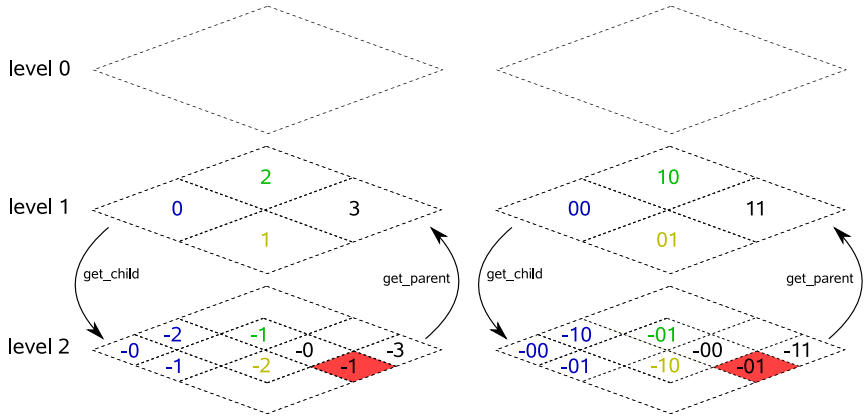


Figure 4: Obtaining parent and child key from node: quadric and binary notation

As an example, the full key for the highlighted cell in Fig. 4 is 131 in quadric, 11101 in binary notation (we have dropped the place-holder bit in the figure for clarity).

The local sorted list of particle keys would thus provide a natural starting point for determining their parent nodes if we knew how they were distributed. In a dynamic application we cannot assume anything about their distribution, however, so instead we start from the highest (coarsest) level and work down to the leaves. As in a sequential algorithm[3], all particles are initially attached to the root, in this case a cube encompassing the whole simulation region. Next, the region is subdivided into 8 sub-boxes, and the particles re-attached accordingly. A sub-box containing exactly one particle is defined as a leaf; a box with 2 or more constitutes a twig and empty boxes are discarded. This procedure is continued at the

next highest level until each particle sits in its own box. For highly clustered distributions, it may become necessary to relax this requirement, otherwise simulations with more than a few million particles can (and do!) result in identical key assignments.

Each new leaf or twig node created this way is added to the local hash-table via the same hash function (3) as the particles. Collisions are again dealt with via a simple linked list. In principle this function can be refined to improve the distribution of hash-table addresses in memory space: the sharing of keys across a number of processors keeps the collision count down to tolerable levels.

2.4 Global branch nodes

At their coarsest level, the local trees will contain 'incomplete' twig nodes; that is, nodes which cross domain boundaries. Information from neighbouring domains is therefore needed to complete them. To facilitate the exchange of information (and later multipole moments) between processors, a set of local 'branch' nodes is defined first, comprising the minimum number of *complete* twig and leaf nodes covering the whole local domain—Fig. 5. This set of branch nodes is then broadcast to all other processors, so that each one

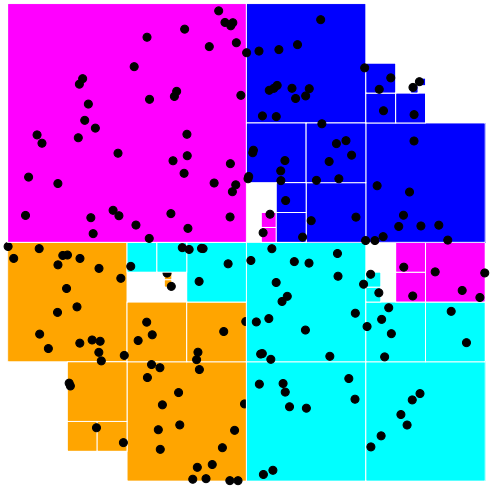


Figure 5: Branch nodes belonging to 4 processor domains.

subsequently knows where to request any missing non-local particle or multipole information. For example, a branch's child nodes can immediately be found from a byte code stored with the hash-table entry, the first 8 bits of which declare which children exist at the next refinement level. Applying the operation (5) yields each (still non-local) child key. A branch's hash-entry will also contain the total number of particles contained beneath it, so that the top level nodes above can now be filled in up to the root. At this point the

local trees comprise 3 types of node: i) twig or leaf nodes covering the local domain, ii) branch nodes and iii) top level twig nodes, each covering the whole simulation region—Fig.6. Leaf node entries contain a pointer to the actual particle coordinates, charge and mass, as well as a globally unique label for tracking purposes. Twig nodes, including the special branch nodes, contain pointers to the multipole moments of their associated charge distributions, together with some flags indicating the status of non-local child nodes (in particular, whether a local copy already exists).

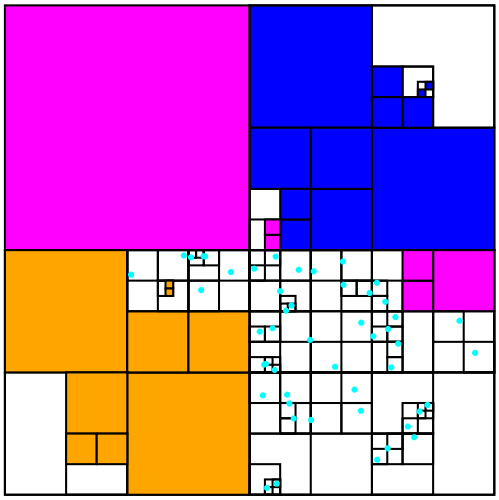


Figure 6: Local tree for processor 1 *prior* to tree-walk. The shaded boxes represent the branch nodes gathered from all other processors.

Before we take a closer look on these steps and their implementation we want to illustrate the tree structure with its different node categories. To this end Figure 7 shows a simplified hierarchical view of the tree data structure.

The top view in this figure shows nodes and particles as they would be organized in a serial version of the code. Each particle is identified as a leaf L while all other nodes are fill (global) nodes. A traversal starts from root level checking the multipole acceptance criterion and processes the whole structure down to the leaves, if necessary. The bottom view visualizes the parallel version of this structure: leaf nodes are distributed across the tasks but for the traversal the branch nodes B are necessary, acting as an entrance to non-local trees during tree traversal. To this end these nodes have to be available on every task in addition to their local trees (including the leaves) and the global structure from root to the branches. Furthermore, we have indicated the members of the interaction lists on task 1. Since only these nodes are necessary for the force summation, some parts of the branch and global structure stored in the tree of task 1 turn out to be redundant. We discuss the impact of this effect in section 3.

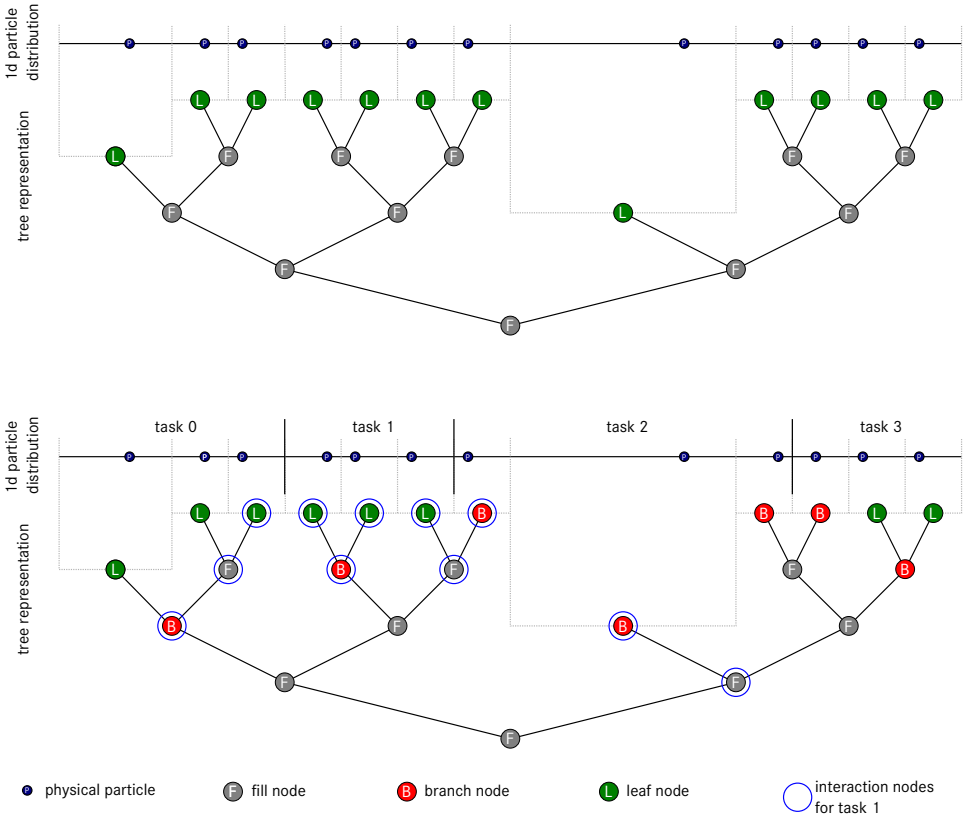


Figure 7: Hierarchical view of the tree structure. Top: nodes and particles as they would be organized in a serial version of the code. Bottom: leaf nodes are distributed across the tasks but for a consistent data structure the branch nodes B are necessary and act as an entrance to non-local trees during tree traversal. Members of the interaction lists on task 1 are also indicated.

2.5 Construction of multipole moments

Once the basic tree structure is in place, it is a straightforward matter to accumulate multipole moments and attach them to each node from the leaves up, making them accessible by the node's key. Once again, this procedure is considerably simplified by sorting the keys for the twig-nodes contained within the list of local branch nodes. Twig nodes with the highest keys will, by definition, have the highest refinement levels:

$$\texttt{level} = \frac{\log(\texttt{key})}{\log 8} \tag{6}$$

This means that multipole moments at deeper levels can be successively shifted up to their parent levels using simple displacement vectors:[3]

$$\sum_i q_i x_i \quad \rightarrow \quad \sum_i q_i x_i - x_s \sum_i q_i$$

$$\sum_i q_i x_i^2 \quad \rightarrow \quad \sum_i q_i x_i^2 - 2x_s \sum_i q_i x_i + x_s^2 \sum_i q_i$$

$$\sum_i q_i x_i y_i \quad \rightarrow \quad \sum_i q_i x_i y_i - x_s \sum_i q_i y_i - y_s \sum_i q_i x_i + x_s y_s \sum_i q_i,$$

where $\boldsymbol{r}_s$ is the shift vector from the child nodes to their parent.

This procedure is continued by working through the sorted list of twigs in reverse order up to the local branch nodes, which then contain the *complete* multipole information for the local domain. This information is then broadcast to all other processors, so that the remaining top-level nodes can be filled in using the shifting rules. At the end of this procedure, each processor has the complete multipole expansion for the entire simulation region contained in the root node.

2.6 Tree traversal

By far the most important and algorithmically challenging part of a parallel tree code is the tree-traversal, which in the present asynchronous implementation requests multipole information 'on the fly' from non-local processor domains. Rather than performing complete traversals for one particle at a time, as many 'simultaneous' traversals are made as possible, thus minimizing the duplication incurred when the same non-local multipole node is requested many times and maximising the communication bandwidth by accumulating many nodes before shipment. In practice, this means creating interaction lists for batches of around 1000 particles at a time before actually computing their forces. The routine `tree_walk`, which finds the interaction list for each batch has the structure depicted in Fig. 8 and Fig. 9.

while any particle not finished walk **do**
  find next node on particle's walk_list
  if MAC OK **then**
    put node on interaction_list
    `walk-key = next-node`
  else if MAC not OK for local node **then**
    subdivide: `walk-key = first-child`
  else if MAC not OK for *non*-local node **then**
    `walk-key = next-node`
    put particle on defer_list
    put node on request_list
  end if
  remove finished particles from walk list
end while

Figure 8: Local tree traversal for batch of particles

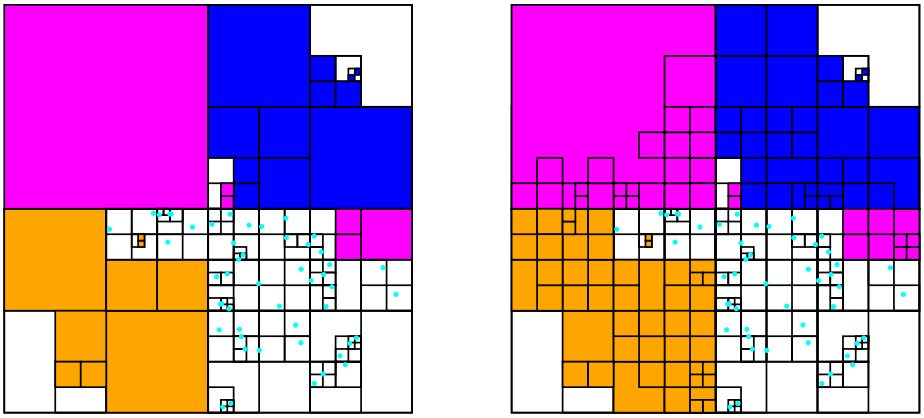
```
gather request lists for non-local nodes from all processors

for all remote processors do
   initiate receive buffer for incoming child data
   send off requests for remote child data
end for

for all remote processors do
   test for incoming request
   package and ship back child multipole data to processor that requested it
end for

for all requests do
   if data has arrived for requested node then
      create new hash-table entries for each child
   end if
end for

copy particle defer_lists to new walk_lists for next pass through tree
```

Figure 9: Parallel tree traversal with multipole exchange

In the first, local part of this procedure, traversals are made through the local tree using the familiar divide-and-conquer strategy common to sequential tree codes[17]. The multipole acceptance criterion (MAC) determines whether to accept or subdivide local nodes as usual, but also provides for a third possibility: the subdivision of a *non*-local node for which child data is not yet available. This is then placed on a special 'request list' to be processed in the 2nd half of the routine when all particles have completed their traversals as far as they can with the available local node data. Each processor then compiles a list of nodes it needs child data from, and sends them to the owners of the parent nodes. In the first pass, these will just be the branch nodes. On receipt of a request list, a processor packages and ships back the multipole data for the children. The use of non-blocking SENDS and RECEIVES for the multipole information allows some overlap of communication with the creation of new hash-table entries locally. At the end of all the traversals, each processor's local tree contains all the nodes required to compute the forces on its own particles. The nodes fetched during the traversals can eventually take up most of the space in the local hash-table, as Fig. 10 illustrates.

2.7 Force summation

Once an interaction list has been found for a particle, it is a straightforward task to compute its force and potential. Separation of the actual force sum from the tree traversal has the advantage that this floating-point-intensive routine can be hardware-optimised. Also, the physics and algorithm are kept naturally apart, so that additional forces, for example, short-range components or magnetic fields and/or corrections due to periodic boundary



a) b)

Figure 10: Tree for processor 1, the domain in bottom right quadrant: a) before and b) after traversals for all locally held particles.

conditions (see next section) can be added with relative ease. In the present implementation, forces are computed for each batch of interaction lists returned from the tree-walk routine. One subtlety which arises here is that even if overall load-balancing has been arranged during the domain decomposition, it is not necessarily guaranteed for each batch of particles (which may comprise only 1/100 of the total number on each processor). To redress this problem, the batch size N_b for each processor is determined individually, so that the integral

$$\sum_{p=1}^{N_b} N_{\text{int}}(p) \tag{7}$$

with $N_{\text{int}}(p)$ being the number of interactions of particle p in the previous timestep, is the same and each processor computes the same number of interaction pairs during each pass.

2.8 Extension to periodic systems

The parallel tree code is capable of simulating systems with large particle numbers. However, for bulk simulations, it is necessary to eliminate surface effects resulting from the simulation region's boundaries. This is done by periodic extension of the system to virtually infinite size using mirror images of the original simulation box, resulting in two modifications of the algorithm itself, as denoted in Fig. 11: i) particles leaving the original simulation box have to be reinserted on the opposite side, and ii) additional contributions to the formal force sum for each mirror box at position $\vec{n}$ have to be considered:

$$\vec{F}_i(\vec{r}_1, \ldots, \vec{r}_N; t) = \sum_{\vec{n}\in\mathbb{Z}^3} \sum_{\substack{j\neq i \\ \text{if } \vec{n}=0}} \frac{\mathrm{d}}{\mathrm{d}\vec{r}_i}\phi(r_{ij} + \vec{n}\cdot L) + \vec{F}_{\text{ext}}(\vec{r}_i; t)\ . \tag{8}$$

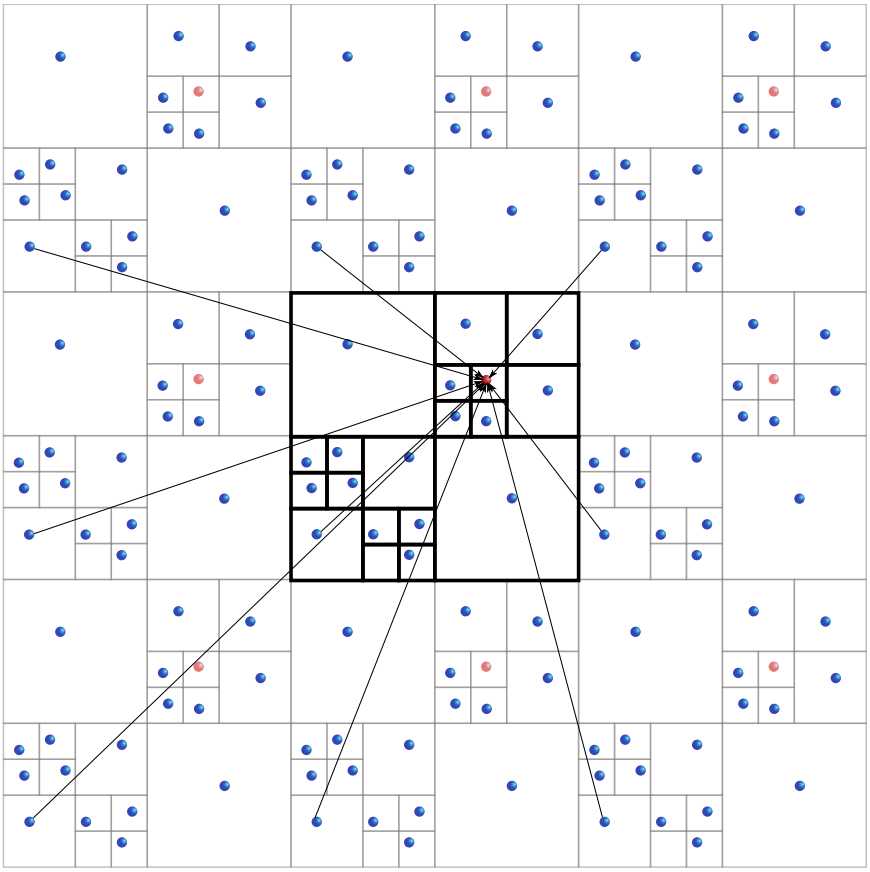


Figure 11: Modification of the algorithm for simulating periodically extended systems: the force acting on a particle (red) from another one (blue) results from all of its mirror images.

Due to the long-range character of the Coulomb interaction, the box sum cannot be truncated arbitrarily, since it is only conditionally convergent. Additionally, in Eq. 8, the effective number of interactions grows linearly with the number of mirror boxes and hence cubic with the effective simulation region size. Consequently, a direct summation is computationally much too expensive and fast summation schemes are necessary.

The standard Ewald method[18,19], i.e. splitting the box sum into a real and a Fourier space part is not used for periodic extension in PEPC, since its convergence and accuracy are rather difficult to control and it would have to be applied to each particle separately.

Instead, an elegant approach borrowed from the fast multipole method[2,20] is adopted here. It is based on the bipolar expansion of the inverse distance in terms of Legendre polynomials:

$$\frac{1}{|\vec{a}_2 - \vec{a}_1 + \vec{n}|} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{j=0}^{\infty} \sum_{k=-j}^{j} (-1)^j \, \mathcal{O}_l^m(\vec{a}_1) \, \mathcal{M}_{l+j}^{m+k}(\vec{n}) \, \mathcal{O}_j^k(\vec{a}_2) \qquad (9)$$

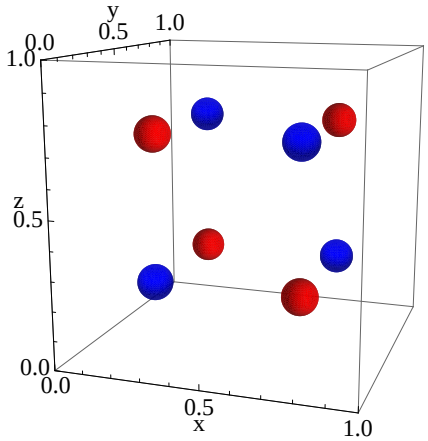
with the multipole coefficients

$$\mathcal{O}_l^m(\vec{a} = [r, \theta, \varphi]) = \frac{r^l}{(l+m)!} \mathrm{P}_{lm}(\cos\theta) \mathrm{e}^{-\mathrm{i}m\varphi} \tag{10}$$

and the Taylor(like) coefficients

$$\mathcal{M}_l^m(\vec{a} = [r, \theta, \varphi]) = \frac{(l-m)!}{r^{l+1}} \mathrm{P}_{lm}(\cos\theta) \mathrm{e}^{\mathrm{i}m\varphi} \ . \tag{11}$$

This expression allows for the precalculation of geometry-dependent lattice coefficients[21,22]. These can be utilized to formulate the lattice contribution to the force for each particle in an $\mathcal{O}(1)$ step, which results in an additional $\mathcal{O}(N)$ overhead for the periodic extension. Additionally, in contrast to the Ewald summation scheme, treating non-cubic simulation regions and systems with periodicity in less than three spatial directions is possible with only small and straightforward modifications.



- overall force per particle
  - analytic: $|\vec{F}| = 0$
  - PEPC: $|\vec{F}| \approx 2.79 \cdot 10^{-4}$
- total energy
  - analytic: $E \approx -13.980\,517$
  - PEPC: $E \approx -13.980\,618$

Figure 12: The 3D-Madelung system for computation precision verification is a cubic system with alternating charges in the cube's corners. Red points denote positively charged particles, blue particles are negatively charged. The PEPC results coincide very well with the analytically derived numbers.

The accuracy of this approach is illustrated with a simple Madelung lattice, an infinitely large cubic system with alternating charges on the lattice nodes, as depicted in Fig. 12. Due to symmetry, the overall force onto each particle has to vanish. On the other hand, the total energy in one cubic cell is

$$E_{\mathrm{Madelung}}^{\mathrm{3D}} = 8 \cdot \sum_{i,j,k \in \mathbb{Z}}{}' \frac{(-1)^{i+j+k}}{(i^2+j^2+k^2)^{\frac{1}{2}}} \approx 13.9805 \tag{12}$$

in normalized units. The results calculated by PEPC are given in Fig. 12 and comply satisfactorily.

3 Algorithm Scaling and Performance

The adaption of an algorithm to massively parallel systems (thousands of tasks) creates a multitude of challenges. In general, these differ from those which arise at scales of only a few tasks. E.g. in the case of strong scaling at large scales, all components which have a constant or even a P dependent complexity must be avoided. However, some algorithms might depend on non-scaling actions, like collective communication in PEPC, and will have to be optimized.

Initialise particle properties $\boldsymbol{r}_i, \boldsymbol{v}_i, q_i, m_i$	N/P
Key construction: $(x_i, y_i, z_i) \rightarrow k_i$	N/P
Sort keys: $k_1, k_2, ...k_N$	$N/P \log N$
Domain decomp.: $k_1, ..k_n$; $k_{n+1}, ..k_{2n}$; ...; $k_{N-n}...k_N$	N/P
Construct branch nodes	P $\log N/P$
Fill in top level local tree nodes	$\log P$
Build multipole moments	$\log N/P$
Construct interaction lists (tree traversal)	$N/P \log N$
Compute forces and potential	$N/P \log N$
Update particle velocities and positions	N/P

Table 1: Algorithmic scaling of major routines in PEPC. The symbols N and P represent the total number of particles and processors respectively, and $n = N/P$.

The overall algorithm is depicted together with the theoretical scaling of each major routine in Table 1. We see that in principle, all of the above routines can be performed in parallel, and thus require a computational effort $\mathcal{O}(N/P)$, give or take a slowly varying logarithmic factor. Single-timestep benchmarks with this new code broadly confirm the theoretical scalings indicated in Table 1. As expected, most of the time is spent in the tree-traversal and force-summation routines: the total overhead incurred by the tree construction (which includes the steps 2.3, 2.3 and 2.4 described previously) is around 3%, although this figure excludes tree-nodes copied locally during the traversal—Table 2.

Routine/ No. CPUs	8	16	64
Domain decomposition	0.2	0.24	0.33
Tree building	2.3	2.3	2.7
Tree traversal	32.9	36.1	40.8
Force summation	64.4	61.2	55.7

Table 2: Breakdown of relative computational effort (percentage of wall-clock time spent in each routine) in the parallel tree code for a test case with 100k particles and 8, 16 and 64 processors respectively on a PC cluster

The first part of this section demonstrates PEPC's scaling on a massive parallel system, IBM Blue Gene/P Jugene at the Jülich Supercomputing Centre. To our knowledge, the achieved scaling is exceptional among the currently available tree codes[23,24]. For strong scaling, there is unavoidably a maximal number of tasks up to which a speedup can be gained, due to essential non-scaling elements: sorting of particles and global information exchange. Even if unconditional strong scaling is currently not possible, it is important to ensure optimal efficiency of all application's parts to be able to reach the highest scales. Therefore, the following sections give an insight in the efficiency of the communication strategy and the parallel sorting routine, as well as the identification of one of the current bottle necks: the P-dependency of the number of branch nodes.

The PEPC code is written in a generic fashion without the usage of external libraries. This results in excellent portability. In the PRACE benchmarking framework. PEPC was run on four different computer architectures, namely: IBM Blue Gene/P (jugene), IBM Power6 (huygens), Cray XT5 (louhi), Intel Nehalem (juropa).

Figure 13 shows strong scaling results from 1 to 8192 mpi tasks on Jugene. We have used four different datasets since the number of particles per task for PEPC is bound by 10^5 on Jugene. To this end we used 1×10^5, 1.6×10^6, 2.56×10^7 and finally 1×10^8 particles to demonstrate PEPC's capabilities. As these runs show our implementation is able to use up to 8192 mpi tasks very efficiently with adequate datasets. The scaling behavior is very good for 1 to 4096 mpi tasks yielding more than 3 orders of magnitudes of speedup. Especially for a high number of particles per task the code scales nearly perfectly. However, lowering this ratio detaches the scaling from the ideal linear speedup. Furthermore, the appearance of this phenomenon is not only coupled to the ratio of particles per tasks but also on the number of tasks itself: the scaling behaviour is much better for small than for large datasets.

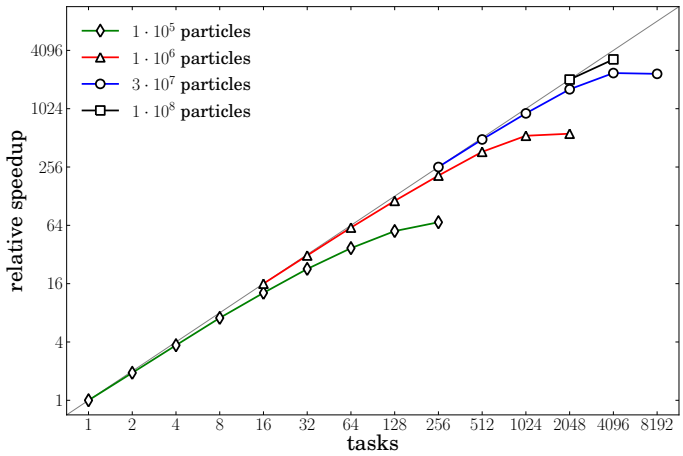


Figure 13: Hybrid scaling for various homogeneous datasets on IBM Jugene, strong scaling for each dataset.

To analyse this behavior, Figure 14 shows the detailed timings on the large dataset for each of the four major steps involved in the one iteration of the algorithm illustrated in Fig.1. It becomes clear that these steps behave very differently for larger number of tasks. While the time-consuming tree traversal and force calculation steps show perfect strong scaling the domain decomposition and tree construction are clearly responsible for its saturation at this level. Since steps 1 and 2 do not scale at all, they start to dominate the calculation time at 8192 tasks in this case.

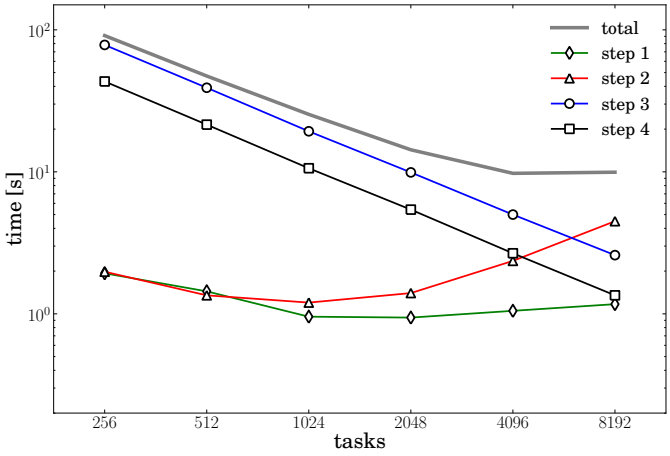


Figure 14: Detailed timings on IBM Jugene for 2.56×10^7 particles in a cube, split into the four main steps of PEPC: domain decomposition (step 1), tree construction (step 2), tree traversal (step 3) and force calculation (step 4)

Both steps 1 and 2 include collective operations: the domain decomposition during parallel sorting, the tree construction for the branch exchange process. Furthermore, the amount of data exchanged during these operations increases with the number of tasks. The branch structure becomes a particular problem at high task numbers. Each task defines at least one branch – but typically up to 30 – for its local particles, which have to be propagated to all other tasks – see Fig. 5. Moreover, the branch level inside the tree – and thereby the total number of branches – increases during a strong-scaling scenario, i.e. for a fixed problem size with variable number of tasks. In addition, the underlying structure of global fill nodes (see Fig. 7) increases as well, so that this step takes more time, too. The consequences and costs arising from the branch structure therefore are responsible for the lack of scaling of step 2.

While parallel sorting and reordering of the particles during step 1 is a necessary and inevitable part of tree codes and is already implemented very efficiently, it is interesting to ask how the branch- and fill-node structure affect the interaction list generation and force calculation. Detailed analyses[25] reveal that nearly all of the local and fetched nodes appear *at least once* on an interaction list, but with increasing number of tasks the proportion of used remote branches and even used fill nodes tends to zero. While the first result

demonstrates the efficiency of the implemented traversal strategy in PEPC – since nearly no node is fetched but unused, the second result indicates that branch and top level fill nodes themselves are generally redundant with respect to force calculation. On the one hand the tree construction step suffers from branch exchange and global node creation process consuming a significant amount of memory and communication-time, while on the other hand theses structures have virtually no impact on the force calculation. Further analysis shows that these results do not depend significantly on the particle distribution. Even for Plummer-like distributions[26] the role of the global tree structures is negligible with respect to the interaction lists. But in the same way the traversal efficiency remains very close to 100%, which is a very good result. Therefore, a starting-point for future optimization of efficiency, memory consumption and scaling would be to optimise the branch structure.

Moreover, an important further issue which must be addressed at such high concurrency levels before the code can be usefully applied to real physics problems is parallel I/O. The main output from PEPC takes the form of regular particle dumps containing their properties (coordinates, velocities, keys, labels and possibly local field information). These are used both for checkpointing (restarting the code mid-simulation) and for postprocessing purposes. Like any other particle code, a bottleneck is created if this data first has to be gathered or distributed among a few thousand tasks. Rather than relying on parallel versions of standard portable data formats (HDF5, netcdf, MPI-I/O), we have deployed a new scalable I/O library for writing code data – SIONlib – developed at JSC[27]. Using this format, particle data is written to and read from a single 'task-local' binary file at each checkpoint. Postprocessers equipped with SIONlib are also speeded up considerably.

4 Summary and Outlook

In this paper we have described our parallel tree code PEPC and demonstrated its capabilities on an IBM Blue Gene/P system, simulating many-million particle systems. With its highly optimized traversal routine this code scales exceptionally well up to 4096 tasks providing library users with a very flexible, portable and mesh-free algorithm for plasma physics, Newtonian gravity and vortex methods. However, as our benchmarks have shown, the collective operations during domain decomposition and the tree construction process induce a significant overhead. While parallel sorting is an inevitable part of a tree code a detailed analysis of the tree data structure proved the efficiency of the traversal, but also indicated a massive redundancy of branch and global nodes.

As a consequence, our current focus is on a interaction list prediction method to avoid the explicit storage of unused nodes during tree construction. By abstracting the branch concept we hope to optimize PEPC's memory footprint significantly, clearing the way for even larger datasets beyond 10^9 particles and scaling well beyond 10^4 tasks.

References

1. J. Barnes and P. Hut, *A hierarchical $O(N \log N)$ force-calculation algorithm*, Nature, **324**, 446–449, (1986).
2. L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys., **73**, 325–348, (1987).

3. S. Pfalzner and P. Gibbon, *Many Body Tree Methods in Physics*, Cambridge University Press, New York, (1996), ISBN 0-521-49564-4 (hardback); ISBN 0-521-01916-8 (paperback).
4. V. Springel, N. Yoshida, and S. D. M. White, *GADGET: A code for collisionless and gasdynamical cosmological simulations*, New Astronomy, **6**, 79–117, (2001).
5. P. Gibbon, W. Frings, S. Dominiczak, and B. Mohr, *Performance analysis and visualization of the N-body tree code PEPC on massively parallel computers*, in: Proc. Parallel Computing 2005, Malaga, G. Joubert and *et al.*, (Eds.), vol. 35, NIC Series, Forschungszentrum Jülich, Jülich, (2006).
6. PEPC can be obtained from: https://trac.version.fz-juelich.de/pepc
7. P. Gibbon, R. Speck, A. Karmakar, L. Arnold, W. Frings, B. Berberich, D. Reiter and M. Masek, *Progress in Mesh-Free Plasma Simulation With Parallel Tree Codes*, IEEE Transactions on Plasma Science, **38**, 2367–2376 (2010).
8. P. Gibbon, F. N. Beg, R. G. Evans, E. L. Clark, and M. Zepf, *Tree code simulations of proton acceleration from laser-irradiated wire targets*, Phys. Plasmas, **11**, 4032–4040, (2004).
9. M. S. Warren and J. K. Salmon, *A portable parallel particle program*, Comp. Phys. Commun., **87**, no. 266–290, (1995).
10. U. Becciani, V. Antonuccio-Delogu, and M. Gambera, *A modified parallel tree code for N-body simulation of the large-scale structure of the universe*, J. Comp. Phys., **163**, 118–132, (2000).
11. J. Dubinski, *A parallel tree code*, New Astronomy, **1**, 133–147, (1996).
12. H. Yahagi, M. Mori, and Y. Yoshii, *The forest method as a new parallel tree method with the sectional voronoi tessellation*, Ap. J. Supp., **124**, 1–9, (1999).
13. M. S. Warren and J. K. Salmon, *A parallel hashed oct-tree N-body algorithm*, in: Supercomputing '93, pp. 12–21, IEEE Comp. Soc., Los Alamitos, (1993).
14. H. Sagan, *Space-Filling Curves*, Springer, Heidelberg, (1994).
15. H. Shi and J. Schaeffer, *Parallel sorting by regular sampling*, J. Par. Dist. Comp., **14**, 361–372, (1992).
16. M. Hofmann, G. Rünger, P. Gibbon, and R. Speck, *Parallel sorting algorithms for optimizing particle simulations*, IEEE Cluster, (2010, in press).
17. S. Pfalzner and P. Gibbon, *A hierarchical tree code for dense plasma simulation*, Comp. Phys. Commun., **79**, 24–38, (1994).
18. P. P. Ewald, *Die Berechnung optischer und elektrostatischer Gitterpotentiale*, Ann. Phys., **64**, 253–287, (1921).
19. M. J. L. Sangster, M. Dixon, *Interionic potentials in alkali halides and their use in simulations of the molten salts*, Adv. Phys., **25**, 247–342, (1976).
20. I. Kabadshow, *The Fast Multipole Method – Alternative Gradient Algorithm and Parallelization*, Diploma Thesis, Forschungszentrum Jülich, jül-4215, ISSN 0944-2952 (2006).
21. M. Challacombe, C. White, M. Head-Gordon, *Periodic boundary conditions and the fast multipole method*, J. Chem. Phys., **107**, 10131–10140, (1997).
22. K. N. Kudin, G. E. Scuseria, *Revisting infinite lattice sums with the periodic fast multipole method*, J. Chem. Phys., **121**, 2886–2890, (2004).
23. B. Jeon, J. D. Kress, L. A. Collins, and N. Groenbech-Jensen, *Parallel Tree code for two-component ultracold plasma analysis*, Comp. Phys. Commun., **178**, 272–279, (2008).

24. A. F. Nelson, M. Wetzstein, and T. Naab, *Vine – A numerical code for simulating astrophysical systems using particles II: Implementation and performance characteristics*, Ap. J. Supp. Series, **184**, 326–360, (2009).
25. R. Speck, P. Gibbon, and M. Hoffmann, *Efficiency and scalability of the paralle Barnes-Hut tree code PEPC*, in: Parallel Computing (ParCo), (2009).
26. S. J. Aarseth, M. Henon, and R. Wielen, *Comparison of numerical-methods for study of star cluster dynamics*, Astronomy and Astrophysics, **37**, no. 1, 183–187, (1974).
27. W. Frings, F. Wolf, and V. Petkov, *Scalable massively parallel I/O to task-local files*, in: Proc. of the ACM/IEEE Conference on Supercomputing (SC09), pp. 1–11, ACM, Portland, OR, (2009).

The Error-Controlled Fast Multipole Method for Open and Periodic Boundary Conditions

Ivo Kabadshow and Holger Dachsel

Institute for Advanced Simulation
Jülich Supercomputing Centre
Forschungszentrum Jülich, 52425 Jülich, Germany
E-mail: {i.kabadshow, h.dachsel}@fz-juelich.de

The simulation of pairwise interactions in huge particle ensembles is a vital issue in scientific research. The calculation of long-range interactions poses limitations to the system size, since the number of these interactions scales quadratically with the number of particles. Fast summation techniques like the Fast Multipole Method (FMM) can help to reduce the complexity to $\mathcal{O}(N)$. In this article we describe the basic ideas behind the FMM for open boundary conditions, as well as the extension of the FMM to (mixed) periodic boundaries. Together with a tight error control, this scheme enables the simulation of particle systems for different applications without the need to know and tune the FMM specific parameters. The implemented error control scheme automatically optimizes the parameters to obtain a minimal calculation time for a given energy error bound ΔE.

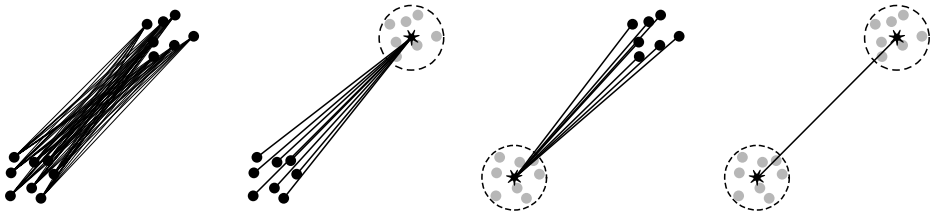
1 Introduction

The main idea behind fast summation methods, especially the FMM, is to give an approximate solution within a given precision goal ε of a certain quantity (energy, forces, potentials). However, this precision goal ε could be machine precision in which case the approximate solution does not differ from the "exact solution" when computed numerically.

The presented approximation will benefit from the following observation. Assume a system of several clustered particles. The contribution on the force or energy, respectively from nearby particles on a test particle within a cluster will be dominant compared to the contribution from remote particles outside the cluster. Admittedly, the remote contribution is not zero. Neglecting these particles would correspond to a cut-off scheme without rigorous error control but ideal complexity $\mathcal{O}(N)$.

Grouping Source Particles

However, a remote particle from a spatial group will have almost the same influence on the test particle near the origin as another particle from the same remote group, since the distance between the test particle at the origin and the remote particles is dominated by a large cluster–cluster distance. Therefore, all particles in a remote cluster could be combined together and may be represented by a new single pseudo particle with a new common center. The influence of several sources is combined into one source, which is depicted in figure 1b.



(a) 56 Direct Interactions

(b) 8 Interactions: Grouped Source Particles

(c) 7 Interactions: Grouped Target Particles

(d) 1 Interaction: Grouped Source & Target Particles

Figure 1: The figure depicts the main idea behind fast summation methods with respect to a particle system. The particles are randomly distributed in space, but show clustering. Chart (a) shows the direct interaction of all particles in one cluster with all particles in the other cluster. Inter-Cluster interactions are not shown. Chart (b) shows the interaction via a source pseudo particle. Chart (c) shows the interaction via a target pseudo particle. Chart (d) shows the interaction with both, source and target pseudo particles.

Grouping Target Particles

The grouping scheme can also be used in reverse. Considering a remote particle and a group of particles clustered together near the origin, the remote particle has almost the same influence on any source particle in the spatial group. Therefore, the interaction of this remote particle can be reduced to the interaction with a pseudo particle containing the clustered source particles (see figure 1c).

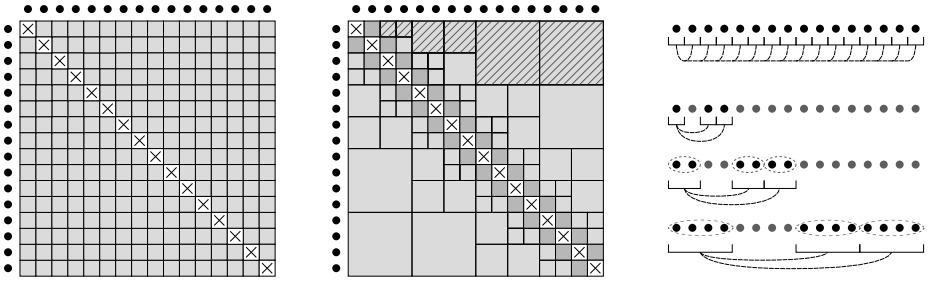
Grouping Both Source and Target Particles

Finally, the last two schemes can be combined together, allowing interactions of two pseudo particles, one at source and one at target particles. This scheme is depicted in figure 1d. Interaction takes place only via pseudo particles and not particles itself. This is advantageous since all particles in a simulation are sources and targets at the same time due to the mutual interaction. Hereby the number of interactions between the particles is reduced even more compared to asymmetric clustering of either source or target points.

Defining Groups of Different Size

To define near and remote particles, a space decomposition scheme is used. For reason of simplicity we assume a recursive decomposition in cubic boxes. The different sized boxes are stored in a tree-like (oct-tree) data structure providing efficient construction of interaction sets, since source and target sets can be increased in size the more distant they are.

To generate the source and target sets from the given particle distribution, two operators are necessary. We distinguish between operators working vertically and horizontally inside the tree. First, the vertical operators shift expansions of source or target sets up and down the tree. Second, the horizontal operator translates source sets into target sets on each tree level.



(a) Interaction Matrix (b) FMM Interaction Matrix (c) Direct and Clustered Interactions

Figure 2: Figure (a) depicts the interaction matrix of the direct interaction. Figure (b) depicts the interaction matrix via the FMM. Each square represents a certain particle-particle or multipole-multipole interaction.Figure (c) depicts the direct interaction (first line) of a 1D system with 16 elements (black circles) and the interaction via the FMM scheme.

However, it is possible that a limited number of nearby particles is not able to interact via pseudo particles. Therefore, these particles interact directly with each other. This restriction can be avoided by very deep trees, hence a very fine spatial decomposition where each particle has its own box. But this is not necessary, since the number of nearby particles is limited and thus does neither impair the overall complexity nor the computation time.

Interaction Sets

Figure 2a shows the interaction matrix of 16 particles. Every cell represents a single interaction pair. The crossed out cells represent interactions from particles with themselves which must not be calculated (singularity) and are dropped eventually. The first interaction matrix in figure 2a represents all $N(N-1)$ interactions. Due to symmetry, the lower left half contains the exact same elements of the upper half. Thus, the overall interactions reduce to $\frac{1}{2}N(N-1)$. The second interaction matrix in figure 2b represents the interaction of the same particles via an FMM computation. Again, the crossed out cells are omitted. The dark-grayed cells represent direct neighbors, which have to be calculated directly. The remaining cells represent interactions via pseudo particles. The streaked cells are shown separately with their interaction sets in figure 2c. One can easily see, that with increasing distance more and more source and target particles are grouped together which reduces the number of interactions dramatically. Again, the lower triangle is symmetric to the upper triangle similar to the direct summation. To establish an accurate algorithm, providing the mentioned features, we need to introduce additional mathematical tools. In the next section we will derive the necessary theorems.

2 Fast Multipole Method for Open Boundaries

2.1 Mathematical Preliminaries

In three dimensions, functions which satisfy the Laplace equation

$$\nabla^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = 0$$

are referred to as harmonic functions. The theory of such functions is called potential theory. A description of the theory can be found in[1]. A solution satisfying the Laplace equation is e.g. $\Phi = 1/d$ with $d = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}$.

If a point particle of unit strength is fixed at $A = (x_0, y_0, z_0)$ then the potential due to this charge at an arbitrary but distinct point $R = (x, y, z)$ is given by

$$\Phi(R) = \frac{1}{d}$$

with d representing the distance between point R and A. The electrostatic field is given by

$$\vec{E}(R) = -\nabla\Phi = -\left(\frac{x-x_0}{d^3}, \frac{y-y_0}{d^3}, \frac{z-z_0}{d^3}\right)^T .$$

Next we want to derive a series expansion for the potential at R in terms of the distance from the origin $\mathbf{r}$.

2.2 Expansion of the Inverse Distance

Given two points $A(a, \alpha, \beta)$ and $R(r, \theta, \phi)$ with a, α, β and r, θ, ϕ being the spherical coordinates, we define the distance d as

$$d := |\mathbf{r} - \mathbf{a}| = \sqrt{r^2 + a^2 - 2ra\cos\gamma}\,.$$

Thus,

$$\frac{1}{d} = \frac{1}{r\sqrt{1 - 2\frac{a}{r}\cos\gamma + \frac{a^2}{r^2}}} = \frac{1}{r\sqrt{1 - 2u\mu + \mu^2}}\,,$$

having set

$$\mu = \frac{a}{r} \quad \text{and} \quad u = \cos\gamma\,.$$

For $\mu \leq 1$, the inverse square root can be expanded in powers of μ, resulting in the following series

$$\frac{1}{\sqrt{1 - 2u\mu + \mu^2}} = \sum_{l=0}^{\infty} P_l(u)\mu^l\,,$$

where

$$P_0(u) = 1, \quad P_1(u) = u, \quad P_2(u) = \frac{3}{2}(u^2 - \frac{1}{3}), \quad \ldots\,.$$

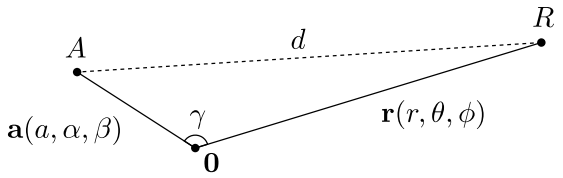


Figure 3: Expansion of the inverse distance $1/d$ into the radial parts a and r and the subtended angle γ between $\overline{OA}$ and $\overline{OR}$.

In general, the $P_l(u)$ are the Legendre polynomials[2] of degree l and are defined by

$$P_l(x) = \frac{1}{2^l l!} \frac{\mathrm{d}^l}{\mathrm{d}x^l} \left[(x^2 - 1)^l \right] .$$

Finally the expression for the inverse distance $1/d$ yields

$$\frac{1}{d} = \frac{1}{r} \sum_{l=0}^{\infty} \frac{a^l}{r^l} P_l(u) = \sum_{l=0}^{\infty} \frac{a^l}{r^{l+1}} P_l(\cos\gamma) . \tag{1}$$

The radial parts a and r of the two coordinates $\mathbf{a}$ and $\mathbf{r}$ are now factorized.

2.3 Spherical Harmonic Addition Theorem

Unfortunately, $P_l(\cos\gamma)$ does still depend on both coordinates A and R via $\cos\gamma$ and cannot be used to derive a fast summation scheme. A useful representation requires the introduction of spherical harmonics, allowing to factorize both source and target locations. By transforming the Laplace equation in spherical coordinates (see figure 4a), we get

$$\frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial \Phi}{\partial r} \right) + \frac{1}{r^2 \sin\theta} \frac{\partial}{\partial \theta} \left(\sin\theta \frac{\partial \Phi}{\partial \theta} \right) + \frac{1}{r^2 \sin^2\theta} \frac{\partial^2 \Phi}{\partial \phi^2} = 0 .$$

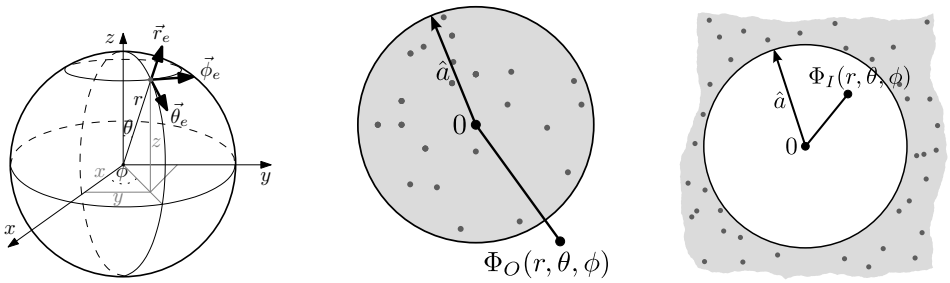
The solution of the equation can be found by assuming a separable solution of the form

$$\Phi(r, \theta, \phi) = R(r) T(\theta) P(\phi)$$

leading to an expression including spherical harmonics Y_{lm} and coefficients M_{lm} and L_{lm}

$$\Phi(r, \theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} L_{lm} Y_{lm}(\theta, \phi) r^l + \frac{M_{lm} Y_{lm}(\theta, \phi)}{r^{l+1}} .$$

Remark 2.1. It should be noted that for a potential $\Phi_O(r, \theta, \phi)$ with $r > \hat{a}$, the coefficients L_{lm} have to be zero to satisfy the decay at infinity for the potential Φ_O as shown in figure 4b. For the potential $\Phi_I(r, \theta, \phi)$ with $r \leq \hat{a}$ inside the sphere with radius $\hat{a}$ the elements M_{lm} must be zero. (see figure 4c).



(a) Spherical coordinate system

(b) Analytic domain of a multipole expansion of the potential Φ_O with $r > a$

(c) Analytic domain of a local expansion of the potential Φ_I with $r < a$

Figure 4: The solution for the potential $\Phi(r, \theta, \phi)$ consists of two separate solutions. The domain can be split into two distinct parts. First, the local part inside the sphere. Second, the external part outside the sphere.

2.4 Expansion of Particle-Particle Interactions

Following the notation of Head-Gordon[3] and the derived formulas from the last sections, we can expand the coordinates of a particle-particle interaction as follows

$$\begin{aligned}\frac{1}{|\mathbf{r}-\mathbf{a}|} &= \sum_{l=0}^{\infty} P_l(\cos\gamma)\frac{a^l}{r^{l+1}} \\ &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{(l-m)!}{(l+m)!}\frac{a^l}{r^{l+1}}P_{lm}(\cos\alpha)P_{lm}(\cos\theta)e^{-im(\beta-\phi)}\,.\end{aligned}$$

We define the scaled associated Legendre polynomials $\tilde{P}_{lm}$ and $\tilde{\tilde{P}}_{lm}$ to ensure numerical stability and to simplify the FMM operators with

$$\tilde{P}_{lm} = \frac{1}{(l+m)!}P_{lm} \quad \text{and} \quad \tilde{\tilde{P}}_{lm} = (l-m)!P_{lm}\,.$$

Now, we define the multipole moments $\omega_{lm}^j(q_j, \mathbf{a}_j)$ for a particle at $\mathbf{a}_j$ with strength q_j about the origin $(0, 0, 0)$. The chargeless version of the multipole is defined by $O_{lm}^j(\mathbf{a}_j)$ with

$$\omega_{lm}^j(q_j, \mathbf{a}_j) = q_j O_{lm}^j(\mathbf{a}_j) = q_j a_j^l \tilde{P}_{lm}(\cos\alpha_j)e^{-im\beta_j}\,.$$

Multipole moments of multiple particles $j \in \{1, \ldots, k\}$ about a common origin can be summed, yielding

$$\omega_{lm}(q, \mathbf{a}) = \sum_{j=1}^{k}\omega_{lm}^j(q_j, \mathbf{a}_j) = \sum_{j=1}^{k} q_j O_{lm}^j(\mathbf{a}_j) = \sum_{j=1}^{k} q_j a_j^l \tilde{P}_{lm}(\cos\alpha_j)e^{-im\beta_j}\,.$$

We can also establish the coefficients of a local Taylor-like expansion of the potential at the origin due to a distant particle at $\mathbf{r}_j$. The chargeless version of the local Taylor-like

expansion is defined by $M_{lm}^j(\mathbf{r}_j)$ as

$$\mu_{lm}^j(q_j, \mathbf{r}_j) = q_j M_{lm}^j(\mathbf{r}_j) = q_j \frac{1}{r_j^{l+1}} \tilde{\bar{P}}_{lm}(\cos\theta_j) e^{im\phi_j} .$$

Again, coefficients with a common origin can be summed

$$\mu_{lm}(q, \mathbf{r}) = \sum_{j=1}^{k} \mu_{lm}^j(q_j, \mathbf{r}_j) = \sum_{j=1}^{k} q_j M_{lm}^j(\mathbf{r}_j) = \sum_{j=1}^{k} q_j \frac{1}{r_j^{l+1}} \tilde{\bar{P}}_{lm}(\cos\theta_j) e^{im\phi_j} .$$

We denote chargeless multipole expansions O_{lm} and chargeless Taylor-like expansions M_{lm} for $k = 1$ without the superscript j, subsequently. The corresponding potential $\Phi(P)$ due to a set of particles can be defined via the following two theorems.

Theorem 2.2. Multipole expansion *Suppose that k particles of strengths $q_j, j = 1, \ldots, k$ are located at the points $\mathbf{a}_j = (a_j, \alpha_j, \beta_j)$, $j = 1, \ldots, k$ with $|a_j| < \hat{a}$ inside a sphere. Then for any $P = (r, \theta, \phi) \in \mathbb{R}^3$ with $r > \hat{a}$, the potential $\Phi(P)$ is given by*

$$\begin{aligned} \Phi(P) &= \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \omega_{lm}(q, \mathbf{a}) \frac{1}{r^{l+1}} \tilde{\bar{P}}_{lm}(\cos\theta) e^{im\phi} \\ &= \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \omega_{lm}(q, \mathbf{a}) M_{lm}(\mathbf{r}) . \end{aligned}$$

Theorem 2.3. Local expansion (Taylor-like) *Suppose that k particles of strengths $q_j, j = 1, \ldots, k$ are located at the points $R_j = (r_j, \theta_j, \phi_j)$, $j = 1, \ldots, k$ outside the sphere with radius $\hat{a}$ with $\hat{a} < r_j$. Then for any $P = (a, \alpha, \beta) \in \mathbb{R}^3$ with $a < \hat{a}$, the potential $\Phi(P)$ is given by*

$$\begin{aligned} \Phi(P) &= \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \mu_{lm}(q, \mathbf{r}) a^l \tilde{P}_{lm}(\cos\alpha) e^{-im\beta} \\ &= \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \mu_{lm}(q, \mathbf{r}) O_{lm}(\mathbf{a}) . \end{aligned}$$

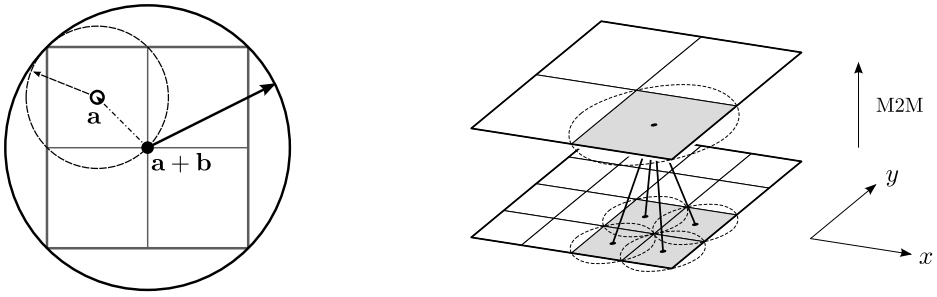
2.5 Mathematical Operators

In this section we describe the FMM operators, which can be derived from the preliminaries of the last section to obtain a fast summation scheme. We need three different operators to establish the FMM scheme. Two operators for the vertical up- and down-shifts between the different tree levels and one operator for the conversion of remote multipole expansions at each tree level.

2.5.1 Translation of a Multipole Expansion (M2M)

With help of an addition theorem[4] we are able to shift the coefficients of a multipole expansion around a point located at $\mathbf{a}$ to a point located at $\mathbf{a} + \mathbf{b}$ (Fig 5) yielding

$$O_{lm}(\mathbf{a} + \mathbf{b}) = \sum_{j=0}^{l} \sum_{k=-j}^{j} O_{jk}(\mathbf{a}) O_{l-j,m-k}(\mathbf{b}) .$$



(a) Analytic Domain (b) Tree View

Figure 5: M2M operator for a two dimensional system. M2M is a vertical operator, exchanging information between levels.

The addition theorem allows us to factorize the potential $1/|\mathbf{r}-(\mathbf{a}+\mathbf{b})|$ into a sum of triple products depending separately on $\mathbf{r}$, $\mathbf{a}$ and $\mathbf{b}$ as follows

$$\begin{aligned}\frac{1}{|\mathbf{r}-(\mathbf{a}+\mathbf{b})|} &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l} O_{lm}(\mathbf{a}+\mathbf{b})M_{lm}(\mathbf{r}) \\ &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\sum_{j=0}^{l}\sum_{k=-j}^{j} O_{jk}(\mathbf{a})M_{lm}(\mathbf{r})O_{l-j,m-k}(\mathbf{b})\,.\end{aligned}$$

The derivation for the multipole expansions ω_{lm} is straightforward

$$\omega_{lm}(\mathbf{a}+\mathbf{b}) = \sum_{j=0}^{l}\sum_{k=-j}^{j} \omega_{jk}(\mathbf{a})O_{l-j,m-k}(\mathbf{b})\,.$$

We identify operator $\mathcal{A}$ with

$$\mathcal{A}_{jk}^{lm}(\mathbf{b}) = O_{l-j,m-k}(\mathbf{b}). \tag{2}$$

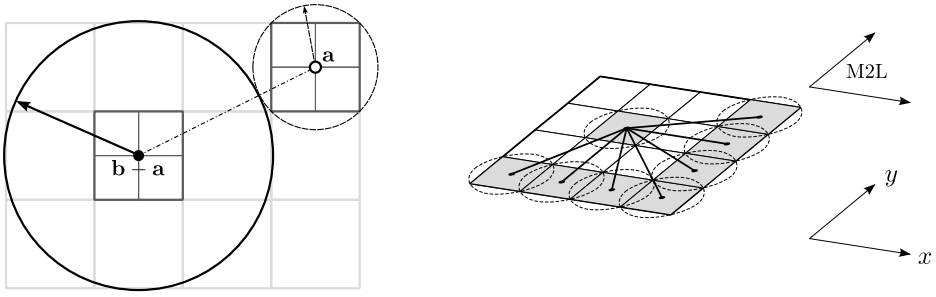
This operator is also called Multipole2Multipole operator or M2M. Operator $\mathcal{A}$ is free of errors. Independent of the length of the multipole expansion, all shifted moments are exact since the operator only includes elements up to the order of the shifted ones. The M2M operator is a vertical operator acting on boxes of different tree levels.

2.5.2 Conversion of a Multipole Expansion into a Local Expansion (M2L)

With help of another addition theorem[4] we are able to transform an external multipole expansion into a local Taylor-like expansion. The chargeless version of the operator yields

$$M_{lm}(\mathbf{b}-\mathbf{a}) = \sum_{j=0}^{\infty}\sum_{k=-j}^{j} O_{jk}(\mathbf{a})M_{j+l,k+m}(\mathbf{b})\,.$$

Similarly to the first FMM operator we can substitute the terms O_{lm} and M_{lm} to obtain the charged version of this operator. Now, we are able to transform coefficients of an external



(a) Analytic Domain (b) Tree View

Figure 6: M2L Operator for a two dimensional system. M2L is a horizontal operator, exchanging information between boxes of the same level.

multipole expansion around $\mathbf{a}$ to local Taylor-like coefficients around $(\mathbf{b}-\mathbf{a})$ as shown in figure 6. The transformed Taylor-like expansion can be represented as

$$\mu_{lm}(\mathbf{b}-\mathbf{a}) = \sum_{j=0}^{\infty}\sum_{k=-j}^{j} M_{j+l,k+m}(\mathbf{b})\omega_{jk}(\mathbf{a})$$

with Multipole2Local (M2L) operator

$$\mathcal{B}_{jk}^{lm}(\mathbf{b}) = M_{j+l,k+m}(\mathbf{b})\,. \tag{3}$$

Besides errors arising from the truncation of the expansion ω_{jk} with $j \leq p$, additional operator errors are introduced, because only $2p$ terms are considered in the sum for the elements $M_{j+l,k+m}$ with $j+l \leq 2p$. Since the operator itself is expanded up to $2p$ it allows to transform all available elements in ω_{jk}. The M2L operator is a horizontal operator acting on boxes of the same tree level.

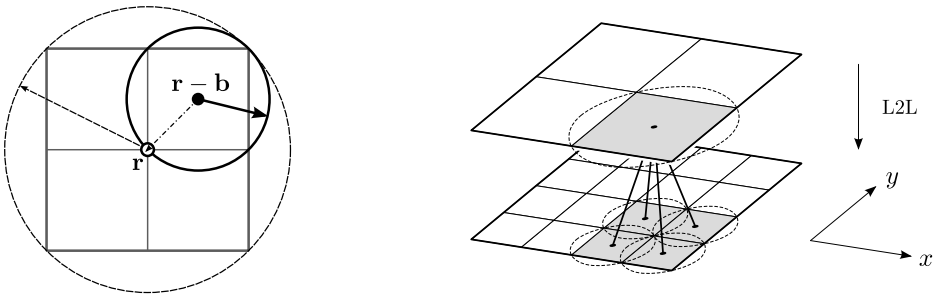
2.5.3 Translation of a Local Expansion (L2L)

The last operator can be obtained by using the first addition theorem again. With the help of the factorized potential, given by

$$\begin{aligned}\frac{1}{|\mathbf{r}-(\mathbf{a}+\mathbf{b})|} &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l} O_{lm}(\mathbf{a}+\mathbf{b})M_{lm}(\mathbf{r}) \\ &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\sum_{j=0}^{l}\sum_{k=-j}^{j} O_{jk}(\mathbf{a})M_{lm}(\mathbf{r})O_{l-j,m-k}(\mathbf{b})\,,\end{aligned}$$

and the manipulation scheme for a double summation from reference[5] with elements

$$A_{jl}(\mathbf{a},\mathbf{b},\mathbf{r}) = \sum_{m=-l}^{l}\sum_{k=-j}^{j} O_{jk}(\mathbf{a})M_{lm}(\mathbf{r})O_{l-j,m-k}(\mathbf{b})$$



(a) Analytic Domain (b) Tree View

Figure 7: L2L Operator for a two dimensional system. L2L is a vertical operator, exchanging information between levels.

we can change the order of the summation

$$\frac{1}{|\mathbf{r}-(\mathbf{a}+\mathbf{b})|} = \sum_{l=0}^{\infty}\sum_{j=0}^{l} A_{jl}(\mathbf{a},\mathbf{b},\mathbf{r}) = \sum_{j=0}^{\infty}\sum_{l=j}^{\infty} A_{jl}(\mathbf{a},\mathbf{b},\mathbf{r})\,.$$

Relabeling the indices $l \leftrightarrow j, m \leftrightarrow k$ of the last equation and resubstituting the original multipole and Taylor-like expansion yields

$$\frac{1}{|\mathbf{r}-(\mathbf{a}+\mathbf{b})|} = \sum_{l=0}^{\infty}\sum_{j=l}^{\infty}\sum_{m=-l}^{l}\sum_{k=-j}^{j} O_{lm}(\mathbf{a})M_{jk}(\mathbf{r})O_{j-l,k-m}(\mathbf{b})\,.$$

Separating the terms $O_{j-l,k-m}$ and M_{jk} results in the operator $\mathcal{C}$, which allows us to translate a Taylor-like expansion located around $\mathbf{r}$ to its center at $(\mathbf{r}-\mathbf{b})$ as shown in figure 7 with

$$M_{lm}(\mathbf{r}-\mathbf{b}) = \sum_{j=l}^{p}\sum_{k=-j}^{j} O_{j-l,k-m}(\mathbf{b})M_{jk}(\mathbf{r})\,.$$

This operator is also called Local2Local (L2L) operator. Compared to operator $\mathcal{A}$, this operator introduces errors due to the finite representation of the multipole expansion. However, compared to operator $\mathcal{B}$ no additional operator errors arise. For a finite Taylor-like expansion with a truncation at p poles the operator is exact with respect of the length of the Taylor-like expansion. The operator L2L is used to shift finite Taylor-like moments from a parent box to its children boxes' centers. The charged version of the operator is given by

$$\mu_{lm}(\mathbf{r}-\mathbf{b}) = \sum_{j=l}^{p}\sum_{k=-j}^{j} O_{j-l,k-m}(\mathbf{b})\mu_{jk}(\mathbf{r})$$

with L2L operator

$$C_{jk}^{lm}(\mathbf{b}) = O_{j-l,k-m}(\mathbf{b})\,. \tag{4}$$

The L2L operator is a vertical operator acting on boxes of different tree levels.

Operator	Purpose	Truncation Error	Additional Operator Error
$\mathcal{A}$	Multipole to Multipole	—	—
$\mathcal{B}$	Multipole to Local	•	•
$\mathcal{C}$	Local to Local	•	—

Table 1: The three FMM operators have different sources of error. The most error-prone operator is operator $\mathcal{B}$. To derive valid error estimation schemes, the influence of operator $\mathcal{B}$ must be considered additionally to the truncation errors introduced by the finite expansion.

Remark 2.4. Omitting operator M2M and L2L increases the complexity to $\mathcal{O}(N \log N)$. Such a scheme would behave like a Barnes-Hut treecode[6], but with improved prefactor.

Obviously all operators induce a complexity of $\mathcal{O}(p^4)$. Especially high precision calculations are slowed down. To overcome this problem, improvements to the operators have been proposed.

2.5.4 Rotation-Based Operators

To circumvent the $\mathcal{O}(p^4)$ operator scaling, White and Head-Gordon[7] proposed a different scheme improving the operator scaling to $\mathcal{O}(p^3)$. The improved scheme is easy to implement with the proposed standard FMM operators and only needs minor modifications. The increased memory usage is negligible and no additional approximations are induced. The original error bounds are retained. The rotation based operators are predicated on the observation that the three dimensional problem is reduced to a one dimensional problem if the translation or shift is carried out along the quantization axis (z-axis). Wigner rotation matrices are applied to rotate the multipole moments. A shift along the quantization axis with $\theta = 0$ and $\phi = 0$ yields a simplified form for the representation of the chargeless multipole moments with

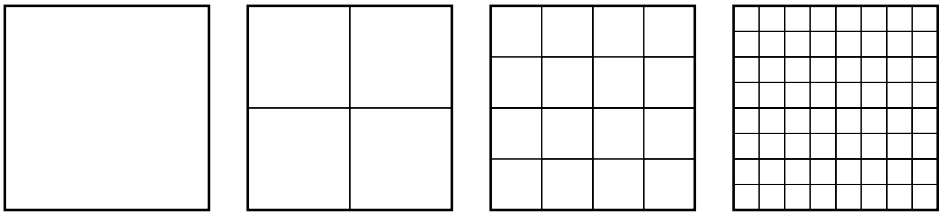
$$O_{lm}(\mathbf{a}) = \frac{1}{(l+m)!} a^l \delta_{m0}$$

$$M_{lm}(\mathbf{r}) = (l-m)! \frac{1}{r^{l+1}} \delta_{m0} \,.$$

Simple rotations preserve the total angular momentum. Therefore, any rotated spherical harmonic will be given as a linear combination of other spherical harmonics having the same order p. The implementation of the FMM presented here is based on these rotation-based operators.

2.6 $\mathcal{O}(N \log N)$ Algorithm

In this section we describe the details of the simpler algorithm not using any translation operators which yields an overall complexity of $\mathcal{O}(N \log N)$. However the final FMM scheme can be based on this approach. We start to enclose our given particles inside a cube and call this box *simulation box*. Next, we introduce a hierarchy of boxes by subdividing this simulation box at half along each axis. The refinement level of the entire simulation



(a) Level 1 – Depth 0 (b) Level 2 – Depth 1 (c) Level 3 – Depth 2 (d) Level 4 – Depth 3

Figure 8: The FMM tree is subdivided until a certain number of particles on the lowest level is reached. Level $L = 1$ corresponds to the highest level, level $L_{\max}$ corresponds to the lowest level. The subdivision shown, adds four times the number of boxes to the tree for each level. In three dimensions, the number of boxes increases by a factor of 8.

box is named level one ($L = 1$). An arbitrary level of refinement $L + 1$ can be obtained by subdividing each box at level L into eight equal subboxes. The 8^L subboxes at level $L + 1$ are called child boxes of their parent box at refinement level L. The scheme corresponds to the construction of an oct-tree depicted in figure 8.

Now we can derive subsets of boxes via the following definitions.

Definition 2.1. *Two boxes A and B are called* next neighbors *if they are at the same tree level and box B is enclosed by a box of size* $(2ws + 1)^3$ *around the center of A with a box edge length of one. Next neighbors interact in the near field (NF).*

The former definition allows us to increase the range for the near field part to the full simulation box ($ws \to \infty$) yielding a direct interaction scheme with $\mathcal{O}(N^2)$ complexity. A definition for a minimal $ws = 1$ criterion can be written as follows:

Definition 2.2. *Two boxes are called* next neighbors *if they are at the same tree level and share a boundary point (see figure 9b). Next neighbors interact in the near field (NF).*

After defining the near field, the far field has to be defined accordingly.

Definition 2.3. *Two boxes A and B are called* well separated *if they are at the same tree level and are not next neighbors (see figure 9c). Well separated boxes interact in the far field (FF).*

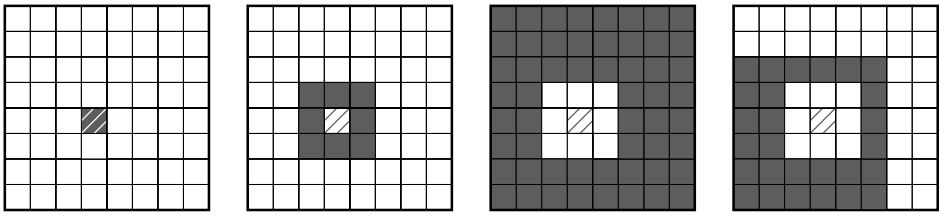
The last definition does not limit the number of interactions in the far field, therefore we have to set up a confined interaction list.

Definition 2.4. *An* interaction list i *is associated with each box A, consisting only of children of the next ws neighbors of A's parent which are well separated from box A (see figure 9d). The interaction list limits the number of far field (FF) interaction for each box.*

These definitions allow us to compute interactions between a constant number of boxes on each level. The spatial refinement enables us to cluster together particles from one and the same box into multipole moments. The expansion of the particles is performed around the box center. With the help of definition 2.4 it is possible to define interaction sets for multipole expansions of these boxes.

Definitions 2.3 and figure 8 show clearly that there are no interactions on level 1 and level 2 since these levels do not contain two separated boxes, hence all boxes are nearest neighbors.

Starting on level 3 with its 64 boxes we can use multipole expansions to compute



(a) Box – Near field (b) Nearest Neighbors interaction set (c) Full far field interaction (d) Far field interaction set

Figure 9: Different interaction sets. (a) All particles within one box (on the lowest level) interact directly. (b) Next neighbor boxes (on the lowest level) interact directly, since the convergence of the expansions can not be guaranteed or may converge slowly. (c) Boxes within the interaction list (on each level) interact via multipoles.

interactions between particles of a source box with a multipole expansion of particles in a remote box. The error bound connected to these interactions will be discussed in more detail in reference[10]. After calculating the interactions on level 3, we can use a recursion scheme to include boxes on the next refinement level (level 4). After subdividing all boxes on level 3, we again identify the interaction set for each box. Since we already accounted for all interactions of boxes outside the parent boxes' next neighbors we must ignore these interactions. The scheme is repeated for every tree level until we reach the lowest level. On the lowest level we calculate the missing interactions for the neighboring boxes via a direct calculation. In the original work of Rokhlin and Greengard the recursive process of refining is halted roughly after $\log_8 N$ levels assuming a homogeneous particle distribution.

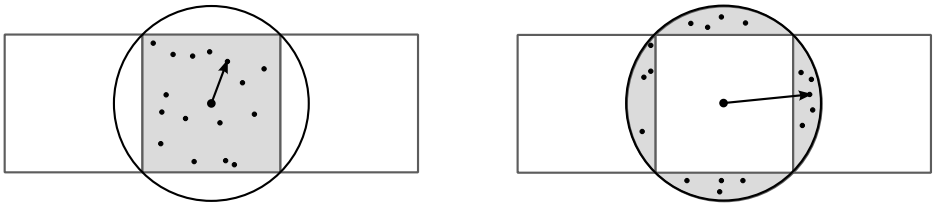
2.7 $\mathcal{O}(N)$ Algorithm

With the help of the introduced operators M2M, M2L and L2L it is possible to derive a scheme with overall complexity $\mathcal{O}(N)$. A multipole expansion of a source box does not have to be evaluated for any particle in the target box. Instead we transform all far field multipole expansions into a single local expansion in the target box. Then, the local expansion is evaluated at each individual target particle position.

General Workflow

The original FMM proposed by Greengard and Rokhlin[8,9] can be sectioned into five main steps. The determination of the FMM parameter set is influenced only by a worst-case error scheme, the actual positions of the particles are not taken into account. We briefly want to outline the single steps, also depicted in figure 11:

- Preprocessing steps
  - Define the separation criterion ws
  - Define the order of multipoles p for a given precision ΔE
  - Define the depth of the FMM tree d
  - Expand particles into multipole moments ω_{lm} on the lowest level



(a) Particles inside a box A

(b) Particles inside A's neighbor boxes

Figure 10: In general it is not possible to set ws to zero, since neighboring boxes may have particles overlapping with the multipole expansion of a considered box. However, the convergence of the expansion demands separation. Therefore, next neighbors must not interact via a multipole expansion.

- Pass 1 ▷ Translate multipole moments ω_{lm} up the tree
- Pass 2 ▷ Transform multipole moments ω_{lm} into Taylor moments μ_{lm}
- Pass 3 ▷ Translate Taylor moments μ_{lm} down the tree
- Pass 4 ▷ Compute the far field contributions of the system
  - Compute potentials $\Phi_{\text{FF}}(x, y, z)$, forces $\mathbf{F}_{\text{FF}}(x, y, z)$ and energy E_{FF}
- Pass 5 ▷ Compute the near field contributions of the system
  - Compute potentials $\Phi_{\text{NF}}(x, y, z)$, forces $\mathbf{F}_{\text{NF}}(x, y, z)$ and energy E_{NF}

The outlined workflow is now described in detail in the following sections.

Preliminary Steps

The coordinates are scaled into a $[0, 1] \times [0, 1] \times [0, 1]$ unit simulation box to guarantee numerical stability of the computations and to simplify the involved FMM operators. Since the FMM allows *a priori* error bounds the order of poles p can be set-up for a given separation criterion. Let us assume for the moment a separation criterion $ws = 1$. A higher value of ws would yield a faster convergence of the expansion (less multipole terms are needed) but higher costs in the near field computations. Since this part scales quadratically in N, we want to eliminate as many direct interactions as possible, therefore ws has to be set to one; it's minimal value. The reason for this minimum ws is shown in figure 10.

After defining the order of poles p, we are able to set up the FMM tree. The refinement of the tree stops, once a certain number of particles in the lowest level is reached (e.g. k particles per box, $k \ll N$). The particles are sorted via a radix sort[11] into the lowest level boxes. Now we expand all particles in each lowest level box into multipole moments about the center of the same box.

Pass 1

Assuming we have more than 3 levels (for $ws = 1$) in the oct-tree, we now shift the multipole coefficients at the lowest level to the center of the parent box via the M2M operator. Since each parent box consists of 8 child boxes (in 3D) the moments of the expansion can be summed up at the new center and are stored as moments of the parent box. The scheme is repeated until level 3 is reached. Now we have a multipole expansion for each box on every level (starting at level 3).

Each child expansion at the center of the box at $\mathbf{a}_i$ is shifted to the center of the common parent box at $\mathbf{a}+\mathbf{b}$ and then summed up with the shifted expansion of the other seven child boxes.

$$\omega^i_{lm}(\mathbf{a}+\mathbf{b}) = \sum_{j=0}^{l}\sum_{k=-j}^{j} \mathcal{A}^{lm}_{jk}(\mathbf{b}_i)\omega^i_{jk}(\mathbf{a}_i)$$

$$\omega_{lm}(\mathbf{a}+\mathbf{b}) = \sum_{i=1}^{8} \omega^i_{lm}(\mathbf{a}+\mathbf{b})$$

Pass 2

In the second pass we apply a modification of the already known scheme from the described $\mathcal{O}(N \log N)$ scheme. Instead of bringing every particle in a given target box to interaction with all multipoles in the interaction set, we transform (at most) 189 source multipole moments within each interaction set into local Taylor-like moments for each target box on each level.

$$\mu^i_{lm}(\mathbf{b}-\mathbf{a}) = \sum_{j=0}^{p}\sum_{k=-j}^{j} \mathcal{B}^{lm}_{jk}(\mathbf{b}_i)\omega^i_{jk}(\mathbf{a}_i)$$

$$\mu_{lm}(\mathbf{b}-\mathbf{a}) = \sum_{i=1}^{\text{ilist}} \mu^i_{lm}(\mathbf{b}-\mathbf{a})$$

Pass 3

The third pass shifts the Taylor-like moments starting from level 3 to the lowest level via the L2L operator. On the way downwards, the Taylor-like moments of the actual level are summed up with the shifted interactions from a higher level. The following equations show a shift of a Taylor-like expansion μ_{jk} at level L to expansions μ^i_{lm} at level $L+1$.

$$\mu^i_{lm}(\mathbf{r}-\mathbf{b}_i) = \sum_{j=0}^{p}\sum_{k=-j}^{j} \mathcal{C}^{lm}_{jk}(\mathbf{b}_i)\mu_{jk}(\mathbf{r})$$

Pass 4

The fourth pass finally computes the interactions between the Taylor-like moments representing all effects of all well separated particles and the particles at positions $\mathbf{a}_i = (a_i, \alpha_i, \beta_i)$ inside the actual target box. For the far field part of the potential Φ we obtain

$$\Phi_{\text{FF}}(\mathbf{a}_i) = \sum_{l=0}^{p}\sum_{m=-l}^{l} \mu_{lm}(\mathbf{r}) a_i^l \tilde{P}_{lm}(\cos\alpha_i) e^{-im\beta_i}\,.$$

Pass 5

The fifth pass calculates the neglected interactions, hence all interactions which cannot be evaluated via far field expansions because of the chosen separation criterion ws. Since we ensured that the number of particles M on the lowest level is independent of the total number of particles N the costs of this pass is $\mathcal{O}(MN)$ with $M \ll N$. Pass 5 contains all interactions from particles within one box M_{ibox} and particles from $2ws$ neighbor boxes M_{jbox} in each dimension. For the near field part of the potential Φ we obtain

$$\Phi_{\text{NF}}(\mathbf{r}_j) = \sum_{\substack{i=1 \\ i \neq j}}^{M_{\text{ibox}}} \frac{q_i^{\text{ibox}}}{r_{ij}} + \sum^{\text{jbox}} \sum_{i=1}^{M_{\text{jbox}}} \frac{q_i^{\text{jbox}}}{r_{ij}} \,.$$

2.8 Enhanced Workflow

The original scheme was modified to allow for an error estimation scheme. Details can be found in figure 11 and reference[10].

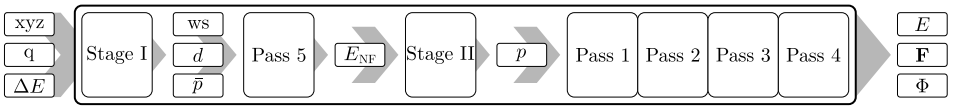


Figure 11: FMM workflow with automatic parameter estimation.

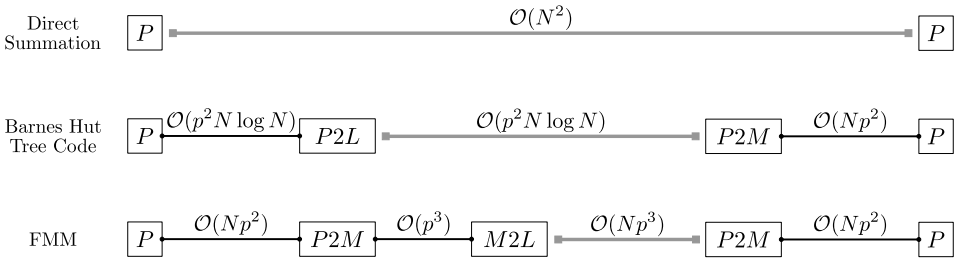


Figure 12: Comparison of the complexity of three different methods. Direct Interaction, Barnes-Hut treecodes and the FMM. The black lines represent steps in the algorithm where particles or particle collections are transformed into expansions. The gray lines represent the actual interaction.

3 Fast Multipole Method for Periodic Boundaries

The presented fast multipole scheme in section 2 for open boundaries can be extended to periodic boundary conditions (PBC) as well. The periodic boundaries allow to derive

macroscopic bulk properties of the simulated particle system. A huge (even infinite) ensemble of particles is reduced to a smaller (finite) ensemble inside a finite-sized simulation box. This simulation box then is replicated in all spatial directions. Hence, the influence of the boundary on the particles enclosed in the finite simulation box vanishes and only the influence of the bulk properties remains. As a particle moves through a boundary into a neighboring image box, it will enter on the opposite site of the central simulation box, thus the number density is conserved.

To fill the entire 3D space, several different shapes of the central simulation box may be considered[12]. However, we only derive the algorithm for the most common shape, i.e. the cubic box. The presented scheme may also be applied together with other shapes like parallelepipeds, hexagonal prisms, octahedrons or dodecahedrons.

Some simulations even demand a mixed boundary condition with periodic boundaries in only one or two dimensions, such as electrolyte solutions, membranes, nanopores or nanotubes[13]. The presented FMM scheme can also be applied to such mixed boundary systems. However, the derivation of the algorithm in this section is performed for three dimensional periodicity if not stated otherwise.

3.1 Definition of the Boundary Condition

The definition of the boundary conditions can be performed in a general manner. We only assume that the simulation cell must be translationally symmetric in order to fill the entire $\mathbb{R}^n$ space. No additional assumptions on the basis vectors forming the periodic lattice are necessary.

3.2 Three Dimensional Periodicity

We define a simulation cell $\Gamma(\mathbf{0})$ in three dimensions and the basis vectors $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ with

$$\Gamma(\mathbf{0}) = \{\mathbf{r} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + x_3\mathbf{a}_3 : \, -1/2 \leq x_i \leq 1/2, \text{ for } i = 1, 2, 3\} \, .$$

Additionally, we chose $\mathbf{a}_i$ in such a way that the volume V_Λ of the simulation cell is defined by $V_\Lambda = \mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3) > 0$. Now, we set up a lattice Λ of translationally symmetric copies $\Gamma(\mathbf{n})$ with $\mathbf{n} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3$ and $n_i \in \mathbb{Z}$ of the original central simulation cell $\Gamma(\mathbf{0})$ by

$$\Gamma(\mathbf{n}) = \{\mathbf{r} : \mathbf{r} - \mathbf{n} \in \mathbf{\Gamma}(\mathbf{0})\} \, .$$

Each replica cell $\Gamma(\mathbf{n})$ contains the exact same numbers of particles as the simulation cell $\Gamma(\mathbf{0})$. A particle with position $\mathbf{r}_i$ in the central simulation cell $\Gamma(\mathbf{0})$ has a replica particle at $\mathbf{r}_i + \mathbf{n}$ in the lattice cell $\Gamma(\mathbf{n})$.

For the periodicity in two dimensions and one dimension, we follow the derivation of the three dimensional case.

3.3 Convergence of Lattice Sums

Since the potential $\Phi(\mathbf{R})$ does not satisfy

$$|\Phi(\mathbf{R})| \leq A \, |\mathbf{r}|^{-d-\epsilon} \tag{5}$$

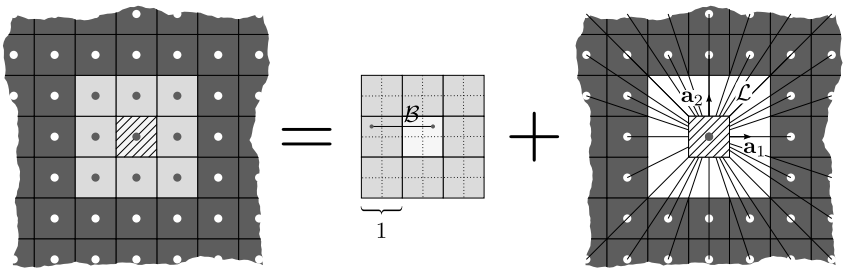


Figure 13: (left) The simulation box and its replicas fill the entire space. The space is divided in two regions, a near field region (light gray) and a lattice region (dark gray).

with d being the periodicity and $A, \epsilon > 0$ for an arbitrary set of point particles with non-zero monopole, dipole- or quadrupole moments, we have to add additional constraints. The following interactions do not obey the inequality in (5):

- Charge–Charge Interactions (1D, 2D, 3D) $|\mathbf{r}|^{-1}$
- Charge–Dipole Interactions (2D, 3D) $|\mathbf{r}|^{-2}$
- Dipole–Dipole Interactions (3D) $|\mathbf{r}|^{-3}$
- Charge–Quadrupole Interactions (3D) $|\mathbf{r}|^{-3}$

Therefore, we add the additional constraint for all particles in the simulation cell

$$Q = \sum_{i=1}^{N} q_i = 0\,.$$

Hence, with box net charge $Q = 0$ only the dipole terms do not converge absolutely. We neglect the dipole-dipole contributions temporarily, but will get back to them later.

3.4 Parameter-Free Renormalization Approach

We follow the approach proposed by Kudin and Scuseria[14] and incorporate the algorithm later on into our error control scheme.

The idea for the evaluation of lattice sums with a renormalization approach was first proposed by Berman and Greengard[15]. The scheme reduces the infinite summation of lattice sites to a rapidly converging finite summation yielding the lattice operator $\mathcal{L}$. The potential $\Phi(\mathbf{0})$ at the center of the central box $(0,0,0)$ can be computed by adding up contributions from lattice supercells of size $(2ws+1)^j \times (2ws+1)^j \times (2ws+1)^j$ with $j \in \mathbb{N}$ going to infinity. Since we want to translate the multipoles from lattice cells into Taylor-like (local) expansions around the common center $(0,0,0)$ of the central box, we can add up all contributions into a single translation operator $\mathcal{L}$. We do not add up an infinite number of lattice sites, however the fast convergence allows to precompute the lattice operator in machine precision. Since the available number of digits is fixed, we can call a result below machine precision numerically exact and therefore no additional runtime parameter is introduced.

3.4.1 Mathematical Operators

Since this approach is based on the original multipole scheme, we can reuse all introduced FMM translation and conversion operators from section 2. The FMM operators are valid on multipole moments ω_{lm} and Taylor-like expansions μ_{lm} and their chargeless counterparts O_{lm} and M_{lm}. For simplicity we only use the chargeless multipole moments O_{lm} and chargeless Taylor-like coefficients M_{lm} in this section.

3.4.2 Translation of a Multipole Expansion

Let us recall the translation of a multipole expansion $O_{jk}(\mathbf{a})$ at $\mathbf{a}$ to a multipole expansion around a new center $\mathbf{a}+\mathbf{b}$ with

$$O_{lm}(\mathbf{a}+\mathbf{b}) = \sum_{j=0}^{l} \sum_{k=-j}^{j} \mathcal{A}_{jk}^{lm}(\mathbf{b}) O_{jk}(\mathbf{a}) \,.$$

We are dropping the indices and abbreviate the operator for the following lattice sum algorithm via $\triangleleft$ into the form

$$O(\mathbf{a}+\mathbf{b}) = \mathcal{A}(\mathbf{b}) \triangleleft O(\mathbf{a}) \,.$$

3.4.3 Conversion of a Multipole Expansion into a Local Expansion

A multipole expansion at $\mathbf{a}$ transforms into a Taylor-like local expansion at $\mathbf{b}-\mathbf{a}$ via

$$M_{lm}(\mathbf{b}-\mathbf{a}) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \mathcal{B}_{jk}^{lm}(\mathbf{b}) O_{jk}(\mathbf{a}) \,.$$

Again, we abbreviate the operator into the form

$$M(\mathbf{b}-\mathbf{a}) = \mathcal{B}(\mathbf{b}) \otimes O(\mathbf{a}) \,.$$

3.4.4 Rescaling of a Multipole Expansion

Since we need a hierarchy of boxes and do not want to recompute multipole moments for each hierarchy level we introduce a scaling operator for a multipole expansion with

$$\mathcal{S}_O(O_{lm}(\mathbf{a})) = 3^l \cdot O_{lm}(\mathbf{a}) \,.$$

3.4.5 Rescaling of a Local Expansion

For similar reasons, the rescaling has to be performed for the local expansion M_{lm} as well. Therefore, we introduce a scaling operator for a Taylor-like expansion by

$$\mathcal{S}_L(M_{lm}(\mathbf{b})) = \frac{M_{lm}(\mathbf{b})}{3^{l+1}} \,.$$

We limit ourselves to the case where $ws = 1$. For a larger separation criterion ws, we have to substitute 3^l and 3^{l+1} with $(2ws+1)^l$ and $(2ws+1)^{l+1}$ accordingly.

3.4.6 Operator Properties

The operators derived in section 2 and the additionally introduced operators $\mathcal{S}_L$ and $\mathcal{S}_O$ are linear operators. Therefore, the following properties apply for infinite expansions

$$\begin{aligned} M \otimes [O_1 \triangleleft O_2] &= [M \otimes O_1] \otimes O_2 \\ O_1 \triangleleft [O_2 \triangleleft O_3] &= [O_1 \triangleleft O_2] \triangleleft O_3 \\ \mathcal{S}_L(M) \otimes \mathcal{S}_O(O) &= \mathcal{S}_L(M \otimes O) \end{aligned} \tag{6}$$

where M represents any Taylor-like local expansion and O represents a certain multipole expansion which still gives a convergent expansion when shifted or transformed. The hierarchy of lattice boxes introduced in the next section guarantees that the speed of convergence and precision is not degraded by the use of the operator properties.

3.5 Lattice Sum Algorithm

Let us derive the local moment in the central cell $\Gamma(\mathbf{0})$ at the center $(0,0,0)$ due to an arbitrary lattice cell $\Gamma(\mathbf{n})$ with $\mathbf{n} \neq \mathbf{0}$ and $\mathbf{n} = j\mathbf{a}_1 + k\mathbf{a}_2 + l\mathbf{a}_3$, $j,k,l \in \mathbb{Z}$. We define the contribution with the help of the $\mathcal{B}$ operator as

$$L^{(j,k,l)} = \mathcal{B}(\mathbf{n}) \otimes \omega$$

with ω being the total multipole moment of the lattice cell. Since all boxes are images of the original simulation box at the center, the multipole expansion of each image box is given by

$$\omega(\mathbf{n}) \equiv \omega(\mathbf{0})\,.$$

The contribution from all lattice cells for 1D-, 2D- and 3D-periodic systems reads

$$\begin{aligned} \text{1D}: \quad L^{\infty} &= \sum_{|\max(0,0,l)|>1} \mathcal{B}((0,0,l)) \otimes \omega(\mathbf{a}) \\ \text{2D}: \quad L^{\infty} &= \sum_{|\max(j,k,0)|>1} \mathcal{B}((j,k,0)) \otimes \omega(\mathbf{a}) \\ \text{3D}: \quad L^{\infty} &= \sum_{|\max(j,k,l)|>1} \mathcal{B}((j,k,l)) \otimes \omega(\mathbf{a})\,. \end{aligned}$$

All nearest neighbors for the central box, i.e. the first layer are excluded to guarantee convergence of the expansions. Since the $\mathcal{B}$ operator is a linear operator, it allows us to precalculate $\mathcal{B}$ for the entire lattice before applying the unique multipole expansion ω. However, applying a direct space summation like in the last set of equations is difficult, because the convergence of the lattice sum is very slow. But, the evaluation of the lattice sum can be rapidly achieved by introducing a hierarchy of supercells with variable size $(2ws+1)^j \times (2ws+1)^j \times (2ws+1)^j$. Since we are only interested in the case where $ws = 1$, we use a hierarchy of size $3^j \times 3^j \times 3^j$. We define an interaction set as follows. For the first interaction set Λ_0 we include all lattice cells in the far field of the central (0,0,0) cell but ignore all cells in the near field of the supercell $3^1 \times 3^1 \times 3^1$. This rule is equivalent to the partitioning rule in section 2.7 for open boundaries. We write

$$L_* = \sum_{\Lambda_0} \mathcal{B}((j,k,l))\,.$$

The far field contribution from region Λ_0 is then

$$L^0 = L_* \otimes \omega \,.$$

We proceed on the next supercell level for $j = 1$ with cell size $3^1 \times 3^1 \times 3^1$. Again we only add cells which are in the near field of the $3^2 \times 3^2 \times 3^2$ supercell. The total contribution to the lattice operator L after this step reads

$$L^1 = L_* \otimes \omega + \mathcal{S}_L(L_*) \otimes [O_* \triangleleft \omega] \,.$$

The second contribution demands scaled local moments L_*. At the same time we combine multipole moments from a $3^0 \times 3^0 \times 3^0$ supercell into multipoles of a $3^1 \times 3^1 \times 3^1$ supercell. O_* is defined for each periodic boundary condition by

$$\begin{aligned}
\text{1D}: \quad O_* &= \sum_{\Omega_1 = -1 \leq l \leq 1} \mathcal{A}((0,0,l)) \\
\text{2D}: \quad O_* &= \sum_{\Omega_1 = -1 \leq j,k \leq 1} \mathcal{A}((j,k,0)) \\
\text{3D}: \quad O_* &= \sum_{\Omega_1 = -1 \leq j,k,l \leq 1} \mathcal{A}((j,k,l)) \,.
\end{aligned}$$

Now, we move on to the next level of supercells given by

$$\begin{aligned}
L^2 = & L_* \otimes \omega \\
& + \mathcal{S}_L(L_*) \otimes [O_* \triangleleft \omega] \\
& + \mathcal{S}_L(\mathcal{S}_L(L_*)) \otimes [\mathcal{S}_O(O_*) \triangleleft [O_* \triangleleft \omega]] \,.
\end{aligned}$$

Applying the operator properties from (6) yield expressions of the form $\mathcal{L}^n = L^n \otimes \omega$. We identify the following partial sums

$$\begin{aligned}
\mathcal{L}^0 &= L_* \\
\mathcal{L}^1 &= L_* + \mathcal{S}_L(L_*) \otimes O_* \\
\mathcal{L}^2 &= L_* + \mathcal{S}_L(L_*) \otimes O_* + \mathcal{S}_L(\mathcal{S}_L(L_*)) \otimes [\mathcal{S}_O(O_*) \triangleleft O_*] \\
&= L_* + \mathcal{S}_L(L_* + \mathcal{S}_L(L_*) \otimes O_*) \otimes O_* \\
&= L_* + \mathcal{S}_L(\mathcal{L}_1) \otimes O_* \,.
\end{aligned}$$

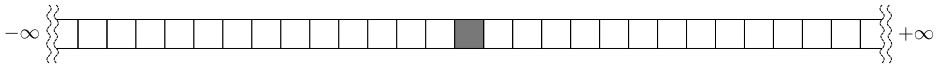
Again, the operator properties allow us to establish an infinite recursion scheme to set up the lattice sum $\mathcal{L}^n$ for increasing n with

$$\begin{aligned}
\mathcal{L}^0 &= L_* \\
\mathcal{L}^{n+1} &= \mathcal{S}_L(\mathcal{L}^n) \otimes O_* + L_* \,.
\end{aligned}$$

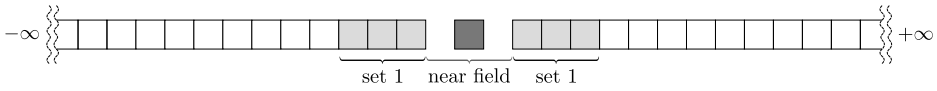
The full infinite lattice is defined via the lattice operator

$$\mathcal{L} = \mathcal{L}^\infty \,.$$

Since we want to use the result on a limited precision machine, we halt the recursion after a certain precision ε has been reached. The convergence of the lattice sum $\mathcal{L}$ increases with increasing multipole order. Therefore, we have to perform the precision check for the

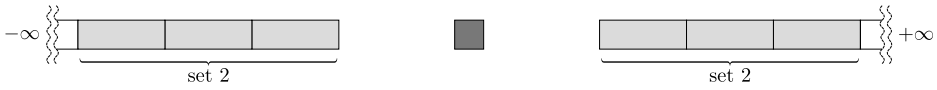


(a) Expand simulation box (gray) to infinity in all periodic dimensions.

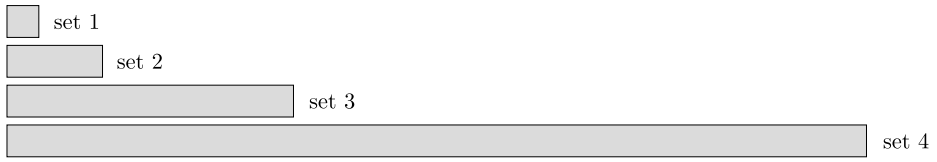


(b) Define first interaction set (light gray) and drop near field

(c) Combine multipole expansions



(d) Define second interaction set (light gray)



(e) Size of interaction sets

Figure 14: 1D representation for the computation of the lattice sum. Figure (a) shows the expansion of the (grayed) central simulation box to infinity. Figure (b) shows the dropped nearest neighbors of the simulation box and the first interaction set. The second interaction set (c)–(d) is generated by combining multipole expansions. Figure (e) shows the increasing size of the interaction sets used with increasing distance to the central simulation cell.

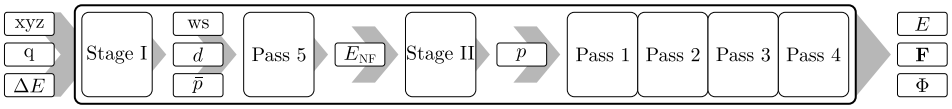
low-order elements. Since, by definition, the monopole element $\mathcal{L}_{00}$ does not converge and due to symmetry not all elements $\mathcal{L}_{lm}$ are non-zero, we halt the recursive scheme when

$$\begin{aligned} \text{1D}: &\quad |\mathcal{L}_{2,0}^{n+1} - \mathcal{L}_{2,0}^{n}| < \varepsilon \\ \text{2D}: &\quad |\mathcal{L}_{4,0}^{n+1} - \mathcal{L}_{4,0}^{n}| < \varepsilon \\ \text{3D}: &\quad |\mathcal{L}_{4,0}^{n+1} - \mathcal{L}_{4,0}^{n}| < \varepsilon\,. \end{aligned}$$

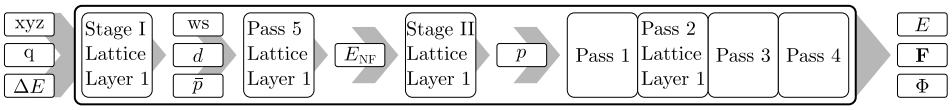
The precision goal we used to obtain the unique lattice operator was set to $\varepsilon = 10^{-64}$. To obtain this precision only 64 iterations are necessary. The lattice constants are independent of the particle distribution and can therefore be precomputed. Since these constants only depend on the lattice structure, they have to be recomputed for additional lattice shapes.

3.6 Implementation Details

To establish an FMM with periodic boundary conditions we have to add and modify certain parts of the implementation in section 2.7. The error-controlled FMM for open boundaries can be described by the following flow chart:



The periodic FMM needs changes to the scheme. In addition to the original approach we have to add contributions from the lattice and the first layer around the central box. The changes occur in the first stage of the error control, pass 5, the second stage of the error control and pass 2.



3.7 Additional FMM Pass for the Lattice Operator

Our current implementation uses precomputed values for the lattice operator $\mathcal{L}$ in pass 2. Since the implementation is based on cubic boxes, currently no other cell shapes are available. The extension to different lattice cell shapes demands extra effort for the lattice operator $\mathcal{L}$. It is possible to precompute $\mathcal{L}$ for all required lattices or compute $\mathcal{L}$ directly in the simulation. Therefore, an additional pass has to be added prior to the error control to compute the missing $\mathcal{L}$ terms. The workload of the additional pass is independent of the number of particles. It has a complexity of $\mathcal{O}(p^4)$ with respect to the order of poles p.

3.8 Modifications of FMM Pass 1–5

Pass 1: All multipole expansions have to be shifted to the highest level $l = 1$. The total charge Q of the simulation cell (ω_{00}) has to be zero in order to guarantee convergence.

Pass 2: In addition to the interactions inside the simulation box, far field contributions from image boxes have to be taken into account. These interactions will occur on all tree levels. Furthermore, interactions from the lattice with the multipole expansion of the simulation box have to be calculated.

Pass 3: The computed Taylor-like expansions in pass 2 have to be shifted into the lowest level boxes. Since these expansions were generated on all levels, pass 3 translates expansions starting at level $l = 1$.

Pass 4: No changes occur in pass 4. All computed and shifted local moments are combined with the multipole moments to yield the far field energy, forces and potentials of the system.

Pass 5: Similarly to pass 2, the near field computation has to take into account interactions from replica particles in image boxes.

3.9 Dipole Correction

Until now we neglected the conditionally convergent dipole-dipole interactions. However, since we added the lattice in a spherical manner the element $\mathcal{L}_{20}$ vanishes. A slabwise summation would yield a different result with $\mathcal{L}_{20} \neq 0$. Unfortunately, the results of such a periodic FMM computation are still not comparable with a standard Ewald summation scheme at this stage. To compare results we have to transform the obtained extrinsic energy into an intrinsic energy obtained by an Ewald summation via

$$E_{\text{in}} = E_{\text{ex}} - \frac{2\pi}{3}\mathbf{d} \cdot \mathbf{d}$$

with $\mathbf{d} = \sum_i q_i \mathbf{a}_i$ being the dipole moment of the simulation cell. The same correction has to be applied for the potential

$$\Phi_{\text{in}}(\mathbf{R}) = \Phi_{\text{ex}}(\mathbf{R}) - \frac{4\pi}{3}(\mathbf{R} - \mathbf{R}_0) \cdot \mathbf{d} + \frac{2\pi}{3}Q$$

with $Q = \sum_i q_i \mathbf{a}_i \cdot \mathbf{a}_i$ being the trace of the Cartesian quadrupole tensor and $\mathbf{R}_0$ being the origin of the coordinate system.

4 Error Control

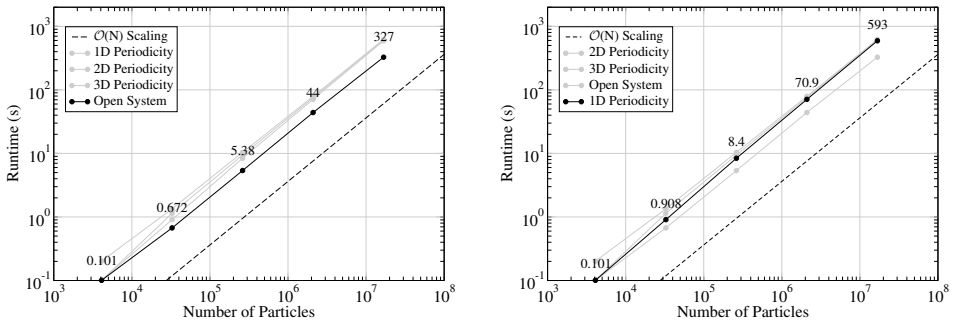
The implemented Fast Multipole method allows to tune the algorithm specific parameter set on the fly to obtain the optimal computation time for a user-given energy error bound. The corresponding error estimation scheme is very complex and therefore cannot be discussed in the scope of this article. More details on the scheme for open boundaries can be found in reference[10] and in an upcoming article for the periodic case.

5 Benchmark

In the following section, we present the scaling of the derived algorithm with respect to the number of particles N. We use five homogeneously distributed simulation sets. The smallest set contains only 4096 particles. Since the computation time for this configuration is around 0.1 seconds, an even smaller test case will not give any additional information concerning the scaling. The largest configuration contains about 16 million particles. All five sets are used in a computation with mixed or full boundary conditions. The results are illustrated in figure 15. All results were determined on the IBM Power6 JUMP[16] cluster at JSC if not stated otherwise.

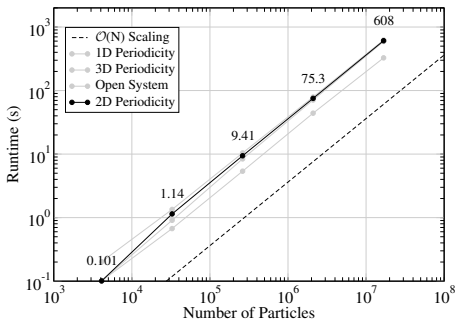
5.1 Crossover Point with Direct Summation

The crossover point between an FMM and a direct summation can only be specified for open boundaries, since a direct summation is not possible for the periodic case due to the infinite number of particles in such systems. For open systems, the crossover point is determined at around 500 particles for low precision, i.e. $\Delta E = 10^{-3}$, and 4000 particles for high precision calculations, i.e. $\Delta E = 10^{-12}$. The computation time of the error control scheme is included therein.

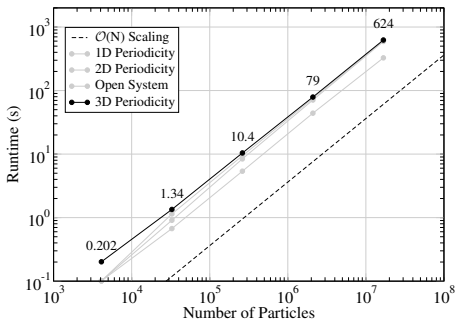


(a) Open boundaries

(b) 1D periodic boundaries



(c) 2D periodic boundaries

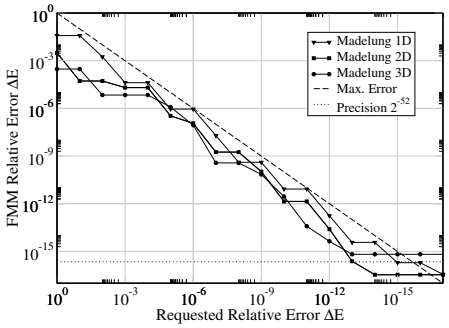


(d) 3D periodic boundaries

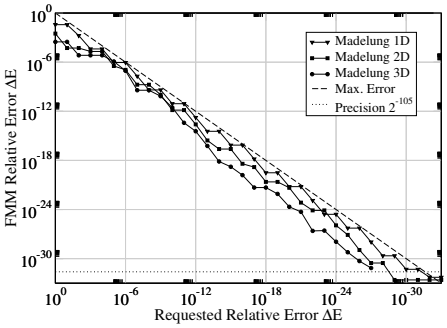
Figure 15: Scaling of the FMM with open, 1D, 2D and 3D periodic boundary conditions. The solid black lines represent the actual periodicity. The remaining shaded lines show, that the impact of the periodic boundary is limited to a factor of two, independent of the number of particles. All results show the optimal $\mathcal{O}(N)$ scaling compared to the reference scaling denoted by a dashed line. When increasing the number of particles for each plotted point by a factor of eight, the computation time increases roughly by a factor of eight. The difference in the runtime between open, 1D, 2D and 3D periodic systems originates from the increasing number of boxes at the boundary of the central box.

5.2 Multi-Billion Particle Testcase

Especially simulations in the field of astrophysics demand huge particle numbers. Therefore, we performed a computation with more than 22 billion particles to check whether the implementation is capable of dealing with such an amount of data. The test set contained more than twice the number of particles used in the Millennium simulation[17]. The computation of the energy and forces took 2 days and 16 hours for one time step on a 1.6 GHz Itanium CPU with 1 TB of main memory. The precision was set to $\Delta E = 10^{-2}$, which is sufficient for most astrophysical calculations. This test example shows that this FMM implementation is capable of computing even very large particle ensembles with limited resources.



(a) Energy error bound with double precision



(b) Energy error bound with quadruple precision

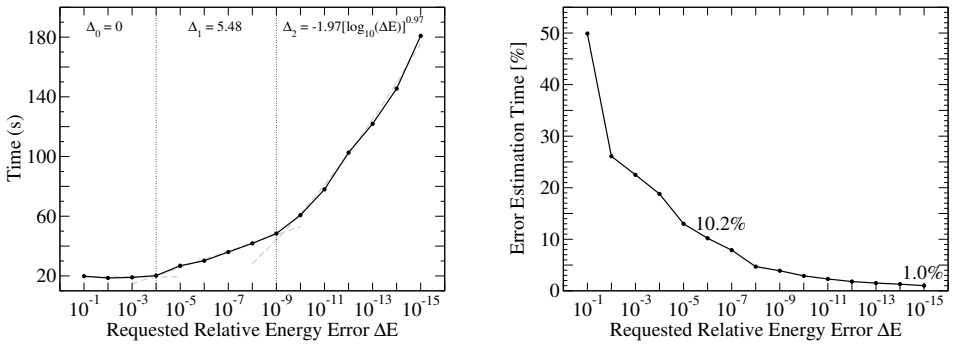
Figure 16: Precision of FMM calculations with periodic boundary conditions. The requested energy error bounds, pictured as dashed lines in a double precision (a) and quadruple precision (b) computation are never exceeded. The tuning of the optimal parameter set is done automatically. Due to symmetry the achieved energy error bound for 3D periodic systems is slightly higher compared to 2D or 1D systems.

5.3 Precision Verification

The precision of calculations with periodic boundaries can be verified directly against analytically known or rapidly converging solutions. That is, a Madelung particle system[18] serves as reference system. The Madelung constant characterizes the potential Φ at the origin due to the periodic lattice. The reference energy is determined by adding up all contributions $\frac{1}{2}\sum_{i=1}^{N} q_i \Phi(\mathbf{r}_i)$ inside the simulation box. The size of the simulation box for any periodicity can be increased by adding more and more particles from replica boxes. Thereby, it is possible to verify the precision of the algorithm even for millions of particles. All precision checks are performed for several simulation box sizes up to approximately 16 million particles. The results do not show any additional errors with increasing system size, except for minor fluctuations at machine precision for 8^8 particles. The data plotted in figure 16 is taken from test runs with the smallest possible Madelung particle sets.

5.4 Precision Scaling

The FMM allows optimal computation time for low as well as high precision simulations. Depending on the user-requested energy error bound the computation time will increase or decrease. As depicted in figure 17a the FMM computes a system with an error bound of 10^{-1} nine times faster compared to a high precision error bound of 10^{-15}. Since the error estimation scheme can be regarded as a FMM itself the computation time for low precision 10^{-1} includes a large amount of tuning time contributing 49.9%. Since this tuning overhead is almost constant for a given particle system, hence independent of the requested precision, the percentage will decrease with increasing precision. To obtain an even faster computation for low precision simulations, the error estimation can be disabled for several time steps.



(a) Computation time with decreasing error bound

(b) Error estimation time with decreasing error bound

Figure 17: The computation time for 2097152 particles (including the time to tune the FMM parameter) is depicted in (a). The graph can be divided in three sections. A low precision section ($10^{-1} \dots 10^{-4}$), where the computation time is constant (Δ_0). A section ($10^{-4} \dots 10^{-9}$) where the computation time increases linearly (Δ_1). And a high precision section (Δ_2) where the computation time follows a power law. The percentage of computation time of the error estimation scheme is shown in (b). The tuning can be disabled for several time steps during a simulation, since a (small) number of time steps most likely won't change the optimal FMM parameter set.

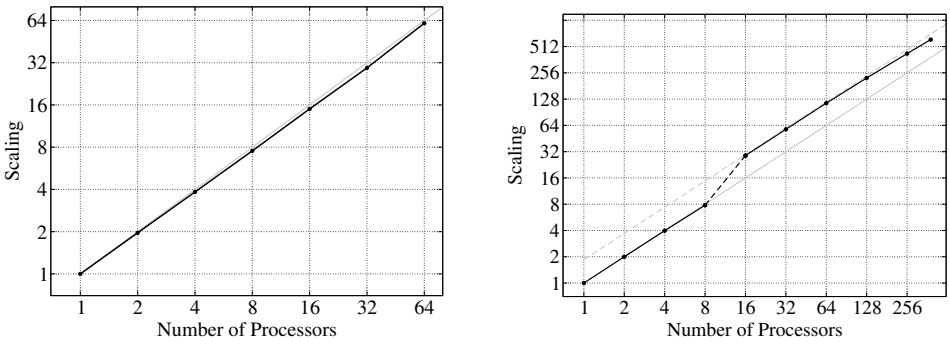
Parallel Scaling And Efficiency

The FMM shows almost ideal scaling for homogeneously distributed particle systems. For a system consisting of 2097152 particles and open boundary conditions a strong scaling test (see figure 18a) was performed up to 64 processors. The computation time with 64 processors was 0.24 seconds. The parallel efficiency at 64 processors achieved 95.6%. A second example with one billion particles is shown in figure 18b. The parallel efficiency at 256 processors is 95.2% and the runtime 34.5 seconds. The improvement at 16 processors was due to memory consumption. Using less than 16 processors resulted in non-local memory access across a SMP node and therefore slowed down the computation.

6 Main Features

The main features of the current FMM implementation can be summarized as follows

- allows the computation of energy, potential and forces,
- comes with full energy error control,
- includes a on-the-fly runtime minimization,
- can handle open, 1D, 2D and 3D periodic boundary conditions,
- computes 50k – 150k particles per second and core,
- has a small memory footprint,
- is available in single/double/quadruple precision,
- allows exchangeable cusp potential in the near field,
- can handle homogeneous and clustered particle systems,
- is available on Intel/Linux, IBM Power6/AIX, BlueGeneP, HP/Linux machines.



(a) Scaling with 2097152 particles

(b) Scaling with 1073741824 particles

Figure 18: Strong scaling for two homogeneously distributed particles systems on JUMP[16].

7 Summary

We described an $\mathcal{O}(N)$ implementation of the Fast Multipole Method for open and periodic boundary conditions in one, two and three dimensions. The attached error estimation scheme was verified for several precisions with one-, two- and three-dimensional periodicity and revealed tight error bounds never exceeding the requested threshold.

Finally, we can conclude that the presented FMM implementation provides an ideal Coulomb solver for open and periodic boundary conditions. The scientific community may benefit from this implementation, since algorithmic details, like tuning the FMM parameters are hidden from the user and will be adjusted automatically without affecting the computation time. Even high precision calculations up to machine precision can be performed without degrading the runtime. Furthermore, the developed FMM library is independent of third-party libraries and has a small memory footprint.

The developed code provides a firm framework for simulations in the field of molecular dynamics and astrophysics. It is well tested with real world examples and is used in a production environment at Max Planck Institute for the Physics of Complex Systems in Dresden[19–22].

References

1. O. D. Kellogg, *Foundations of Potential Theory*, Springer, 1967.
2. I. Gradshteyn and I. W. Ryzhik, *Tables of Integrals, Series and Products* , Academic Press, 1965.
3. C. A. White and M. Head-Gordon, *Derivation and efficient implementation of the fast multipole method*, J. Chem. Phys., **101**, 6593–6605, 1994.
4. M. J. Caola, *Solid harmonics and their addition theorems*, J. Phys. A Math. Gen., **11**, 2, L23-L25, 1978.
5. J. Choi, *Notes on formal manipulations of double series*, Commun. Korean Math. Soc., **18**, 4, 781–789, 2003.

6. J. Barnes and P. Hut, *A hierarchical O(N log N) force-calculation algorithm*, Nature, **324**, 446–449, 1986.
7. C. A. White and M. Head-Gordon, *Rotating around the quartic angular momentum barrier in fast multipole method calculations*, J. Chem. Phys., **105**, 12, 5061–5067, 1996.
8. L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., **73**, 2, 1987, 325–348.
9. L. Greengard and V. Rokhlin, *A new version of the Fast Multipole Method for the Laplace equation in three dimensions*, Acta Numer., **6**, 229–269, 1997.
10. H. Dachsel, *An error-controlled fast multipole method*, J. Chem. Phys., **132**, 11, 119901, 2010.
11. H. Dachsel and M. Hofmann and G. Rünger, *Library Support for Parallel Sorting in Scientific Computations*, Euro-Par 2007 Parallel Processing, 695–704, 2007.
12. H. S. M. Coxeter, *Regular Polytopes*, Dover Publications, 1973.
13. A. Leach, *Molecular Modelling: Principles and Applications (2nd Edition)*, Prentice Hall, 2001.
14. K. N. Kudin and G. E. Scuseria, *Revisiting infinite lattice sums with the periodic fast multipole method*, J. Chem. Phys., **121**, 7, 2886–2890, 2004.
15. C. L. Berman and L. Greengard, *A renormalization method for the evaluation of lattice sums*, J. Math. Phys., **35**, 11, 6036–6048, 1994.
16. `http://www2.fz-juelich.de/jsc/jump`
17. V. Springel et al., *Simulating the joint evolution of quasars, galaxies and their large-scale distribution*, Nature, **435**, 629, 2005.
18. R. E. Crandall and J. F. Delord, *The potential within a crystal lattice*, J. Phys. A Math. Gen., **20**, 9, 2279, 1987.
19. A. Mikaberidze and U. Saalmann, and J. M. Rost, *Laser-Driven Nanoplasmas in Doped Helium Droplets: Local Ignition and Anisotropic Growth*, Phys. Rev. Lett., **102**, 12, 128102, 2009.
20. U. Saalmann, *Electron emission from laser-irradiated atomic clusters*, Laser Phys., **19**, 2, 202–207, 2009.
21. C. Gnodtke, and U. Saalmann, and J. M. Rost, *Ionization and charge migration through strong internal fields in clusters exposed to intense x-ray pulses*, Phys. Rev. A, **79**, 4, 041201, 2009.
22. G. Bannasch and T. Pohl, *Formation of Atoms in Strongly Correlated, Ultracold Plasmas*, MPIPKS Technical Report, 2008.

Multigrid Methods for Long-Range Interactions

Matthias Bolten

Fachbereich Mathematik und Naturwissenschaften
Bergische Universität Wuppertal
42097 Wuppertal, Germany
E-mail: bolten@math.uni-wuppertal.de

Multigrid methods are optimal methods for the solution of certain PDEs and therefore widely used in different simulation codes in science and engineering. Starting from the formula for calculating the electrostatic potential we derive a formulation of this and related problems that allows for the application of multigrid methods in the calculation of these long-range interactions. The resulting methods are optimal, if the particles are almost evenly distributed and can be parallelized in a very efficient way.

1 Introduction

Electrostatic and gravitational interactions are long-ranged interactions that are important for many applications in e.g. biophysics or astrophysics. Due to their close relation to the Poisson equation

$$-\Delta u(x) = f(x), \text{ for } x \in \Omega,$$

for a suitable domain Ω and appropriate boundary conditions, these interactions can be equivalently treated by the solution of this elliptic partial differential equation (PDE) with a certain right hand side. Under the assumption that the particles are evenly distributed or at least only mildly clustered, numerical methods can be designed that are optimal, if the solution of the partial differential equation is available.

The numerical solution of the Poisson equation is well studied as it is the prototype of an elliptic PDE, in fact it serves as a model problem in most textbooks on numerical solution of PDEs, and it is needed in different fields. As a consequence fast numerical methods have been developed, fast Poisson solvers include specialized direct methods like the cyclic reduction method or FFT-based techniques, as well as iterative methods like SOR, the CG method or multigrid methods. While the first are specifically tailored to the matrices arising while using a special discretization, the latter often suffer from the fact that the number of iterations necessary to solve the system up to a given accuracy grows with the system size. This is not the case for multigrid methods, which yield a solution up to discretization accuracy in optimal, i.e. $\mathcal{O}(N)$, complexity.

The combination of the approach using the solution of the Poisson equation outlined above combined with an efficient multigrid method yields an optimal method for the computation of energies and forces due to electrostatic and gravitational interactions.

In the following, we will first introduce multigrid methods in general in Section 2. In Section 3 the application of multigrid methods in methods involving electrostatic and gravitational interactions is discussed in detail. For that purpose the relation between $1/r$-potentials and the Poisson equation is presented in Section 3.1, then the basic numerical method is derived in Section 3.2 and finally the relevant boundary conditions are treated

in Section 3.4 and Section 3.3. The parallelization of the method is described in Section 4 and a conclusion is drawn in Section 5.

2 Multigrid Methods

The development of multigrid methods goes back at least to the works of Fedorenko[1] and Bakhvalov[2]. Later the work of Brandt[3] unleashed the full potential of multigrid methods. The following introduction is very brief, so we omit proofs, and focussed onto the case of the Poisson equation. An introduction that goes beyond the following can be found in the work of Briggs et al.[4], the book of Trottenberg et al.[5] contains the theoretical foundation of multigrid methods as well as descriptions of advanced multigrid techniques.

2.1 Motivation

As motivation we consider the following Poisson equation

$$\begin{aligned} -\Delta u(x) &= f(x), \text{ for } x \in \Omega \text{ and} \\ u(x) &= 0 \text{ for } x \in \partial\Omega, \end{aligned}$$

where $\Omega = [0,1]^2$. This can be discretized on a rectangular grid with $N = (n+1)\cdot(n+1)$ grid points using the standard 5-point scheme, yielding the linear system of equations of the form

$$\frac{1}{h^2}(4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}) = f_{i,j}, \text{ for } i,j = 1,\ldots,n_k,$$

where $h = 1/n$ and $u_{i,j} = 0$ for $i \in \{0, n+1\}$ or $j \in \{0, n+1\}$. For the sake of simplicity, in the following we write the vectors u, f etc. with two indices, where necessary, to emphasize that they represent quantities on the grid. An actual implementation also often uses a two-dimensional field for each of these vectors.

The resulting linear system is denoted by

$$Lu = f, \tag{1}$$

where $L \in \mathbb{R}^{N\times N}$ and $u, f \in \mathbb{R}^N$. One easily verifies that the eigenvalues $\lambda_{l,m}$ and eigenvectors $\varphi_{l,m}$ of the system matrix L are given by

$$\lambda_{l,m} = 4 - 2\cos(l\pi h) - 2\cos(m\pi h), \tag{2}$$

$$(\varphi_{l,m})_{i,j} = \sin(l\pi i h)\sin(m\pi j h), \tag{3}$$

for $l,m = 1,\ldots,n$. The system can be solved using a simple iterative method, e.g. with the Jacobi method.

Definition 2.1 (Jacobi method)**.** *Let $A \in \mathbb{R}^{N\times N}$, let $b \in \mathbb{R}^N$ and let the solution $x \in \mathbb{R}$ of the linear system*

$$Ax = b$$

be sought for. Let $D \in \mathbb{R}^{N\times N}$ be a diagonal matrix containing the main diagonal of A. Then the Jacobi method *is defined as*

$$\begin{aligned} \phi_{\textit{Jacobi}} : \mathbb{R}^N \times \mathbb{R}^N &\to \mathbb{R}^N \\ (x^{(k)}, b) &\mapsto \phi_{\textit{Jacobi}}(x^{(k)}, b) = x^{(k+1)}, \end{aligned}$$

where

$$x^{(k+1)} = x^{(k)} - D^{-1}(Ax^{(k)} + b), k = 1, 2, \ldots,$$

and $x^(0)$ is an initial guess. The iteration matrix is given by $M_{Jacobi} = -D^{-1}(D - A)$.

The following theorem states a sufficient condition for the convergence Jacobi method:

Theorem 2.2. *Let both A be symmetric positive definite and let the relation*

$$2D > A > 0$$

hold. Then the Jacobi method converges and its convergence rate is given by

$$\rho(M_{Jacobi}) = \|M_{Jacobi}\|_A = \|M_{Jacobi}\|_D < 1.$$

With the help of this theorem and the observation that the smallest of the eigenvalues of L given by (2) is

$$\lambda_{\min} = 4(1 - \cos(\pi h))$$

we obtain that

$$\rho(M_{\text{Jacobi}}) = \cos(\pi h),$$

so the Jacobi method converges slowly for larger systems, i.e. smaller h. Examination of the error

$$e^{(k)} = u^* - u^{(k)},$$

where $u^* = L^{-1}$. With the help of the iteration matrix M_{Jacobi} we obtain that the eigenvector $\varphi_{l,m}$ is damped by a factor of $|\frac{1}{2}(\cos(l\pi h)\cos(m\pi h))|$, so error components co-linear to eigenvectors with indices l, m in the middle of $1, \ldots, n$ are damped efficiently, while components co-linear to eigenvectors with large or small indices are not damped efficiently. By introducing a *relaxation parameter ω* we obtain the ***JOR method*** given by the following definition:

Definition 2.3 (Jacobi method)**.** *Under the same assumptions as in Definition 2.1 the* JOR method *is defined as*

$$\begin{aligned} \phi_{JOR} : \mathbb{R}^N \times \mathbb{R}^N &\to \mathbb{R}^N \\ (x^{(k)}, b) &\mapsto \phi_{JOR}(x^{(k)}, b) = x^{(k+1)}, \end{aligned}$$

where

$$x^{(k+1)} = x^{(k)} - \omega D^{-1}(Ax^{(k)} + b), k = 1, 2, \ldots,$$

and $x^(0)$ is an initial guess. The iteration matrix is given by $M_{JOR,\omega} = -\omega D^{-1}(D - A)$.

Similar to Theorem 2.2 one obtains:

Theorem 2.4. *Let $A \in \mathbb{R}^{N \times N}$ be symmetric and and positive definite and let ω fulfill*

$$0 < \omega < 2/\rho(D^{-1}A).$$

Then the JOR method converges, and its convergence rate is given by

$$\rho(M_{JOR,\omega}) = \|M_{JOR,\omega}\|_A = \|M_{JOR,\omega}\|_D < 1.$$

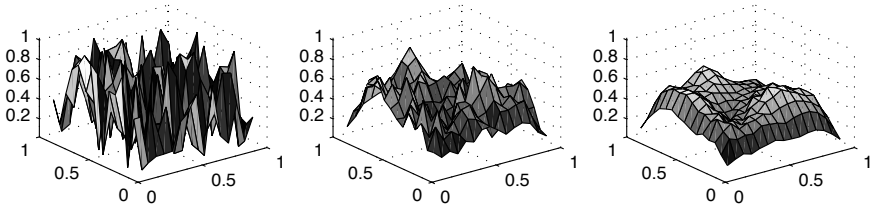


Figure 1: Error of an arbitrarily chosen initial approximation and right hand side of the Laplacian discretized on the unit square using 15^2 grid points before and after application of one and three iterations of a damped Jacobi method with $\omega = 4/5$.

So we obtain that error components co-linear to the eigenvector $\varphi_{l,m}$ are damped by a factor of $|1 - \frac{\omega}{2}(2 - \cos(l\pi h_k) - \cos(m\pi h_k))|$. Now, by choosing ω appropriately, error components that are co-linear to $\lambda_{l,m}$ with l and m close to n can also be reduced, but the behavior for components corresponding to l, m almost 0 is not changed. This behavior can be observed in Figure 1: Obviously, the error components that belong to highly oscillating parts of the error are reduced efficiently, while components that vary slowly are merely affected at all. This is one important observation for the derivation of multigrid methods. The other observation is that a slowly varying error can be represented on a coarser grid.

2.2 Two-grid methods

Guided by the ideas of the previous section, we will now derive two-grid methods. For that purpose, we assume that the system to solve is such that $n = n_\ell = 2^\ell - 1$ for some $\ell \in \mathbb{N}$. Consequently we denote the associated system matrix by L_ℓ and the grid spacing by h_ℓ. The right hand side, current approximation to the solution etc. also are added an ℓ to denote that they belong to this level. Next, another, coarser level $\ell - 1$ with $n_{\ell-1} = 2^{\ell-1} - 1$ is introduced. To define a two-grid method we now need three ingredients:

1. smoother
2. grid-transfer operators
3. coarse-grid correction operator

2.2.1 Smoother

Due to the above observation, iterative methods like the Jacobi method are called *smoothers* in the multigrid setting. For the analysis of the smoother we first define *low* and *high* frequencies:

Definition 2.5. *Let L be given by* (1)*. The eigenvector $\varphi_{\ell;l,m}$ as given by* (3) *is called*

$$\begin{aligned} &\textit{low frequency,} && \textit{if } \max(l,m) < (n_\ell + 1)/2, \\ &\textit{high frequency,} && \textit{if } (n_\ell + 1)/2 \leq \max(l,m). \end{aligned}$$

Now, the smoothing factor of the JOR method is defined as the worst factor by which a high frequency is damped:

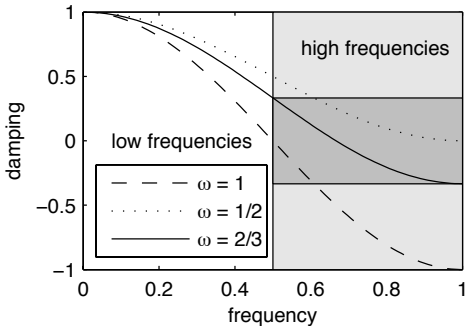


Figure 2: Damping factors $\chi_{l,m}$ for $h \to 0$ of the JOR method for the 1D analogon to our model problem for different relaxation parameters ω. The choice $\omega = 2/3$ is optimal and all high frequency components are damped by a factor of at least $1/3$.

Definition 2.6. *Let L be given by* (1) *and let*

$$\chi_{\ell,l,m}(\omega) := 1 - \frac{\omega}{2}(2 - \cos(l\pi h_\ell) - \cos(m\pi h_\ell))$$

be the factor by which the eigenvector $\varphi_{\ell;l,m}$ is damped by the JOR method. Then the smoothing factor $\mu_(\omega)$ *of the JOR method is defined as*

$$\mu_\ell(\omega) := \max\{|\chi_{\ell;l,m}(\omega)| : (n_\ell + 1)/2 \leq \max(l, m) \leq n_\ell\},$$

Further:

$$\mu(\omega) = \sup_{\ell \in \mathbb{N}} \mu_\ell(\omega).$$

Thus the relaxation parameter is optimal if ω as the minimizer of $\mu(\omega)$. In the 2D case $\omega = 4/5$ is optimal. For the 1D analogon of our problem the choice $\omega = 2/3$ is optimal, as depicted in Figure 2.

2.2.2 Grid-transfer operators

For the transfer from the fine level ℓ to the coarse level $\ell - 1$ and vice versa *restriction* and *prolongation operators* are needed. An example for a restriction operator, that is widely used and that is used in our examples is the *full-weighting operator*. This operator just does an averaging over neighboring components on the grid.

Definition 2.7. *The* full-weighting operator $I_\ell^{\ell-1}$ *transfers a quantity v_ℓ from level ℓ to level $\ell - 1$ by*

$$\begin{aligned}(v_\ell - 1)_{i,j} &= I_\ell^{\ell-1} v_\ell \\ &= \frac{1}{16}(4(v_\ell)_{2i,2j} + 2(v_\ell)_{2i-1,2j} + 2(v_\ell)_{2i+1,2j} + 2(v_\ell)_{2i,2j-1} + 2(v_\ell)_{2i,2j+1} + \\ &\qquad (v_\ell)_{2i-1,2j-1} + (v_\ell)_{2i-1,2j+1} + (v_\ell)_{2i+1,2j-1}(v_\ell)_{2i+1,2j+1}),\end{aligned}$$

for $i, j = 1, \ldots, n_{\ell-1}$. *An alternative description is the* stencil

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{\ell}^{\ell-1}$$

As we deal with zero Dirichlet boundary conditions in our model problem, the values on the boundaries of the grid are always defined to vanish. For prolongation, bilinear interpolation is widely used.

Definition 2.8. *The* bilinear interpolation operator *transfers a quantity* $v_{\ell-1}$ *from level* $\ell - 1$ *to level* ℓ *by*

$$\begin{aligned}(v_\ell)_{i,j} &= I_{\ell-1}^{\ell} v_{\ell-1} \\ &= \frac{1}{4}\begin{cases} 4(v_{\ell-1})_{i/2,j/2}, & \text{for } i \text{ even and } j \text{ even,} \\ 2((v_{\ell-1})_{(i-1)/2,j/2} + (v_{\ell-1})_{(i+1)/2,j/2}), & \text{for } i \text{ odd and } j \text{ even,} \\ 2((v_{\ell-1})_{i/2,(j-1)/2} + (v_{\ell-1})_{(i/2,(j+1)/2}), & \text{for } i \text{ even and } j \text{ odd,} \\ (v_{\ell-1})_{(i-1)/2,(j-1)/2} + (v_{\ell-1})_{((i+1)/2,(j+1)/2} + \\ \quad (v_{\ell-1})_{(i-1)/2,(j-1)/2} + (v_{\ell-1})_{((i+1)/2,(j+1)/2}, & \text{for } i \text{ odd and } j \text{ odd,} \end{cases}\end{aligned}$$

for $i, j = 1, \ldots, n_\ell$. *Alternatively, it is described by the stencil*

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{\ell-1}^{\ell}.$$

Note that the brackets around the stencil of prolongation operator are interchanged to represent that using this stencil means giving values to another quantity. The extension of these operators to 3D is straightforward.

2.2.3 Coarse grid correction operator

Using the restriction and prolongation operators, as well as a discretization of the problem on level $\ell - 1$, the coarse grid operator can be defined. For that purpose, we examine the residual

$$r_\ell^{(k)} = f_\ell - L_\ell u_\ell^{(k)}$$

for some approximative solution $x_\ell^{(k)}$. The residual is the solution of the system

$$L_\ell r_\ell^k = e_\ell^k,$$

where $e_\ell^{(k)} = L_\ell^{-1} f_\ell - x_\ell^{(k)}$ is the error. If the error consists is dominated by low-frequency modes, it is well approximated on the coarser grid, i.e. by $e_\ell^{(k)} \approx I_{\ell-1}^{\ell} e_{\ell-1}^{(k)}$. Similarly, for the residual we have $r_{\ell-1}^{(k)} \approx I_\ell^{\ell-1} r_\ell^{(k)}$, combining this with the operator on the coarser level yields

$$e_\ell^{(k)} \approx I_{\ell-1}^{\ell} L^{-1} I_\ell^{\ell-1} r_\ell^{(k)},$$

so, given that the actual error is governed by low modes, a better approximation is given by

$$u_\ell^{(k)} = u_\ell^k + I_{\ell-1}^\ell L_{\ell-1}^{-1} I_\ell^{\ell-1}(f_\ell - u_\ell^{(k)}).$$

Given that the solution of the coarse grid system is cheaper than that of the fine grid system, this saves computational time.

2.2.4 Two-grid cycle

Given a smoothing method $\phi_{\mathrm{S};\ell}$ that is applied ν_1 times before and ν_2 times after the coarse grid correction, the complete two-grid cycle is performed by Algorithm 2.1. For

Algorithm 2.1 Two-grid cycle $u_\ell \leftarrow \phi_{\mathrm{TGM};\ell}(u_\ell, f_\ell)$

$u_\ell \leftarrow \phi_{\mathrm{S};\ell}^{\nu_1}(u_\ell, f_\ell)$
$r_\ell \leftarrow f_\ell - L_\ell u_\ell$
$r_{\ell-1} \leftarrow I_\ell^{\ell-1} r_\ell$
$e_{\ell-1} \leftarrow L_{\ell-1}^{-1} r_{\ell-1}$
$e_\ell \leftarrow I_{\ell-1}^\ell e_{\ell-1}$
$u_\ell \leftarrow u_\ell + e_\ell$
$u_\ell \leftarrow \phi_{\mathrm{S};\ell}^{\nu_2}(u_\ell, f_\ell)$

the convergence proof basically two properties are needed:

Definition 2.9 (Smoothing property)**.** *An iterative method $\phi_{S;\ell}$ with iteration matrix S_ℓ fulfills the* smoothing property*, if there exists a function $\eta(\nu)$, such that*

$$\|L_\ell S_\ell^\nu\|_2 \leq \eta(\nu)\|L_\ell\|_2 \text{ for all } 0 \leq \nu \leq \infty \text{ with } \ell \geq 0,$$
$$\lim_{\nu\to\infty} \eta(\nu) = 0.$$

It can be shown that for our model problem the damped JOR method[6] satisfies the smoothing property with $\eta(\nu) = c/(\nu + \frac{1}{2})$.

Since the inverse of the operator is approximated on the coarse level, the approximation property is defined as a measure for the quality of this approximation.

Definition 2.10 (Approximation property)**.** *Let $I_{\ell-1}^\ell$ and $I_\ell^{\ell-1}$ be the interpolation and restriction operators and let L_ℓ be the discretization of the underlying partial differential equation as defined above. The twogrid method using these operators is said to fulfill the* approximation property*, if there exists a constant c, such that for all $\ell \in \mathbb{N}$ we have*

$$\|L_\ell^{-1} - I_{\ell-1}^\ell L_{\ell-1}^{-1} I_\ell^{\ell-1}\|_2 \leq \frac{c}{\|L_\ell\|_2}.$$

Various problems arising from the discretization of partial differential equations fulfill the approximation property, for details we refer to the work of Hackbusch[6–12].

Given the smoothing and the approximation property the twogrid method converges, as stated by the following theorem:

Theorem 2.11. *Let the twogrid method $\phi_{TGM;\ell}$ with ν presmoothing iterations and no postsmoothing iterations of the iterative method $\phi_{S;\ell}$ fulfill the smoothing and the approximation property and by T_ℓ denote the iteration matrix of the two-grid method, i.e.*

$$T_\ell = I - I_{\ell-1}^{\ell} L_{\ell-1}^{-1} I_{\ell}^{\ell-1} L_\ell.$$

Then for all $0 < \zeta < 1$ there exists a lower bound $\tilde{\nu}$, such that for all $\nu > \tilde{\nu}$ and for all $h < h_{max}$ we have

$$\|T_\ell S_\ell^\nu\|_2 \leq c\eta(\nu) \leq \zeta.$$

2.3 Multigrid methods

Inside the two-grid method still a linear system has to be solved that, while it is much smaller than the original, could be expensive to solve. An obvious idea is to use another two-grid method again, to solve the smaller linear system. This is the key idea of multigrid: A hierarchy of grids is introduced to approximate the error by solving for the residual on the next coarser grid. On each of the levels, γ steps of the method are performed to approximate the solution to this system, only on the coarsest level of the hierarchy a direct solve is performed. In the following, we assume that we go down to level $\ell = 1$, so the "system" to solve on the coarsest level consists of one variable, only. The result is Algorithm 2.2. Often $\gamma = 1$ is chose, in that case the multigrid cycle is called a "V-cycle" for obvious

Algorithm 2.2 Multigrid cycle $u_\ell \leftarrow \phi_{\text{MGM};\ell}(u_\ell, f_\ell)$

$u_\ell \leftarrow \phi_{\text{S};\ell}^{\nu_1}(u_\ell, f_\ell)$
$r_\ell \leftarrow f_\ell - L_\ell u_\ell$
$r_{\ell-1} \leftarrow I_\ell^{\ell-1} r_\ell$
if $\ell - 1 = 0$ **then**
  $e_0 \leftarrow L_0^{-1} r_0$
else
  $e_{\ell-1} \leftarrow 0$
  for $i = 1$ to γ **do**
    $e_{\ell-1} \leftarrow \phi_{\text{MGM};\ell\,-\,1}(e_{\ell-1}, r_{\ell-1})$
  end for
end if
$e_{\ell-1} \leftarrow L_{\ell-1}^{-1} r_{\ell-1}$
$e_\ell \leftarrow I_{\ell-1}^{\ell} e_{\ell-1}$
$u_\ell \leftarrow u_\ell + e_\ell$
$u_\ell \leftarrow \phi_{\text{S};\ell}^{\nu_2}(u_\ell, f_\ell)$

reasons. In the case $\gamma = 2$ the cycle is known as $W - cycle$, these two choices of γ are most common. In two dimensions the amount of work needed for one multigrid cycle is linear in the number of unknowns on the finest level for $\gamma = 1, 2, 3$, starting with $\gamma = 4$ the amount of work grows faster than that. As the correction is only an approximation, the convergence of the multigrid method has to be analyzed. The convergence of multigrid cycles with $\gamma \geq 2$ is straightforward to show and stated by the following theorem:

Theorem 2.12. *With the same notation as above let*

$$\|S_\ell^{\nu_2} T_\ell S_\ell^{\nu_1}\|_* \leq \sigma, \qquad \|S_\ell^{\nu_2} I_{\ell-1}^\ell\|_* \|L_{\ell-1}^{-1} I_\ell^{\ell-1} L_\ell S_k^{\nu_1}\|_* \leq c$$

hold uniformly for all levels ℓ for some norm $\|\cdot\|_*$. Then the $*$-norm of the iteration matrix M_k is bounded by η^ℓ, where η^ℓ is defined recursively as

$$\eta_0 = \sigma, \qquad \eta_k = \sigma + c\eta_{k-1}^\gamma \quad (k = 1, 2, \dots),$$

where $c, \sigma > 0$. For $\gamma = 2$ and

$$4c\sigma \leq 1$$

the $*$-norm of the iteration matrix M_ℓ is bounded from above by

$$\|M_\ell\|_* \leq \eta = \frac{1}{2c}(1 - \sqrt{1 - 4c\sigma}) \leq 2\sigma,$$

so for $\sigma < \frac{1}{2}$ the method converges with a uniformly bounded convergence rate.

As a consequence, multigrid methods are optimal methods in the sense that the convergence rate is bounded independently of the level and that one iteration of the method has linear computational cost.

2.4 Advanced multigrid techniques

While this brief introduction is relatively easily extendible to other elliptic PDEs, other domains, including more complex ones, and other boundary conditions, e.g. periodic boundary conditions, the development of multigrid methods goes much further than that. Multigrid methods have been developed for non-elliptic problems and for non-linear problems. Further on algebraic multigrid methods (AMG) allow the use of this idea without the explicit generation of the grid hierarchy, this is important e.g. when multigrid has to be used as a black box solver in a mature code. These developments go far beyond the scope of our introduction, a good overview and introduction into more advanced multigrid methods is given in the book of Trottenberg et al.[5].

3 Multigrid Methods for Long-Range Interactions

As just shown multigrid methods are optimal solvers for the Poisson equation. In the introduction we noted that the use of an optimal Poisson solver leads to an optimal method for $1/r$-potentials. In the following, we will derive the numerical method. The idea is related to that of the P3M method that is described e.g. in the book of Hockney and Eastwood[13]. In contrast to the P3M we start deriving the method for the open space problem.

3.1 Electrostatic interactions and the Poisson equation

Assume that the electrostatic energy of a system of N particles[a] with charges q_i at positions x_i, $i = 1, \dots, N$ has to be calculated, i.e. the sum

$$E = \frac{1}{2} \sum_{i=1}^{N} q_i \sum_{j=1, j\neq i}^{N} \frac{1}{4\pi\varepsilon_0} \frac{q_j}{\|x_i - x_j\|_2}.$$

[a] Note that this N is different from the one in the previous chapter.

Obviously, the electrostatic potential U_i of particle i is given by the part inside of the outer sum, so

$$U_i = q_i \sum_{j=1, j\neq i}^{N} \frac{1}{4\pi\varepsilon_0} \frac{q_j}{\|x_i - x_j\|_2}.$$

As the Green function of the 3D Laplace operator is $G(x) = 1/(4\pi\|x\|_2)$ this is equivalent to the solution Φ of the Poisson equation

$$-\Delta\Phi_i(x) = \frac{1}{\varepsilon_0} \sum_{j=1, j\neq i}^{N} q_j \delta(\|x - x_j\|_2)$$

evaluated at the position x_i. As the right hand side consists of a superposition of δ-distributions, this equation cannot be solved numerically. To make it accessible by numerical methods, we replace the δ-distributions on the right hand side by a function $\rho_g = g(\|x\|_2)$. We choose the function $g : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that g is sufficiently smooth, such that

$$\int_{\mathbb{R}^3} \rho_g(x)dx = 1$$

and such that the solution Φ_g of the Poisson equation

$$-\Delta\Phi_g(x) = \frac{1}{\epsilon_0}\rho_g(x)$$

is known analytically. This is similar to the introduction of the error function in the Ewald summation, in contrast to this additionally we demand that g has limited support $R > 0$. So the potential of particle i is given as

$$U_i = q_i \left(\tilde{\Phi}_i(x_i) - \sum_{j\in M_i\setminus\{i\}} q_j \Phi_g(x_i - x_j) - \frac{1}{4\pi\epsilon_0} \frac{q_j}{\|x_i - x_j\|_2} \right), \tag{4}$$

where $\tilde{\Phi}_i$ is the solution of the Poisson equation

$$-\Delta\tilde{\Phi}_i(x) = \frac{1}{\varepsilon_0} \sum_{j=1, j\neq i}^{N} q_j \rho_g(\|x - x_j\|_2)$$

and where $M_i = \{j : \|x_j - x_i\|_2 < R\}$ denotes the set of all indices $j \in \{1, \ldots, N\}$ that closer to particle i than the support of g, including i. This sum is a simple near-field correction.

Now, by subtracting the self-energy correction term $q_i\Phi_g(0)$ from (4), we obtain

$$U_i = q_i \left(\tilde{\Phi}_i(x_i) - \sum_{j\in M_i} q_j \Phi_g(x_i - x_j) - \frac{1}{4\pi\epsilon_0} \frac{1_j}{\|x_i - x_j\|_2} \right)$$

and we can replace the $\tilde{\Phi}_i$, $i = 1, \ldots, N$ by $\tilde{\Phi}$, the solution of the single Poisson equation

$$-\Delta\tilde{\Phi}(x) = \frac{1}{\varepsilon_0} \sum_{j=1}^{N} q_j \rho_g(\|x - x_j\|_2) \tag{5}$$

and the energy is given by

$$E = \frac{1}{2} \sum_{i=1}^{N} U_i.$$

We like to note that no additional errors have been introduced by this transformation. In molecular dynamics simulations not the energy, but the forces are of interest. Obviously the force acting on particle i is given by

$$F_i = q_i \nabla \tilde{\Phi}(x_i) + \text{near-field correction}.$$

A self-energy correction is not necessary, here.

3.2 Basic numerical method

With the previous chapter, a numerical method can be derived easily. For finding the neighbors that are necessary in the near-field correction, the linked-cell algorithm is used, as it is described in the book of Hockney and Eastwood[13]. Here we assume that a cell spacing h is used. The initialization of the necessary data structures is given by Algorithm 3.1. To keep the algorithm simpler by bold indices we denote multi-indices.

Algorithm 3.1 Linked-list initialization

```
for i ∈ 𝒢 do
  HOC(i) ← 0
end for
for i = 1 to N do
  j ← round(x/h)
  LL(i) ← HOC(j)
  HOC(j) ← i
end for
```

The data structure initialized by this algorithm is used in the basic numerical method. Further on the method uses the solution of the Poisson equation (5) on a mesh, for ease of implementation that mesh can coincide with the mesh introduced due to the linked-list data structure. Finally, the basic method is given by Algorithm 3.2.

At this point, given that the numerical solution of is exactly available on the grid, the only error in the method is due to the interpolation of the potential surface.

3.3 Open boundary conditions

In the open boundary case the Poisson equation (5) has to be solved with boundary conditions

$$\tilde{\Phi}(x) \to 0, \text{ as } \|x\|_2 \to \infty.$$

Multigrid methods are well-suited for the numerical solution of this problem. The basic idea is to extend and coarsen the grid successively and impose Dirichlet boundary conditions with the help of the Green function on the coarsest level. This scheme is depicted in

Algorithm 3.2 Basic numerical method

for $\mathbf{i} \in \mathcal{G}$ **do**
  for $\mathbf{j} \in \{\mathbf{j} : \|\mathbf{i} - \mathbf{j}\|_\infty \leq R/h\}$ **do**
    $k = HOC(\mathbf{j})$
    while $k \neq 0$ **do**
      $f(x_i) = q_k \rho_{g_a}(x_i - x_k)$
      $k = LL(k)$
    end while
  end for
end for
Solve $\Delta\Phi_{g_a,\mathcal{P}} = f$ numerically using Poisson solver
$E = 0$
for $k = 1, \ldots, N$ **do**
  Approximate $\Phi_{g_a,\mathcal{P}}(x_k)$ by interpolating the potential surface
  $E = E + q_k(\Phi_{g_a,\mathcal{P}}(x_k) - \Phi_{g_a}(0))$
  for $\mathbf{j} \in \{\mathbf{j} : \|\text{round}(x_k/h) - \mathbf{j}\|_\infty \leq R/h\}$ **do**
    $k = HOC(\mathbf{j})$
    while $k \neq 0$ **do**
      $E = E - q_k\tilde{\Phi}(x_i - x_k) + \frac{1}{4\pi\epsilon_0}\frac{q_i q_k}{\|x_i - x_k\|_2}$
      $k = LL(k)$
    end while
  end for
end for

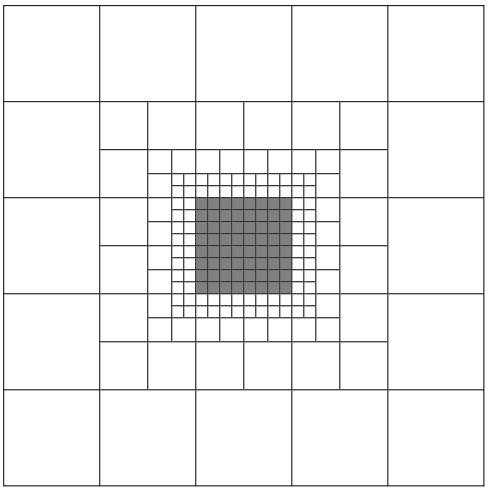


Figure 3: Coarsened grid in 2D.

2D in Figure 3. It can be shown that the error of the numerical method is of the required order and that the numerical complexity stays linear. The details are beyond the scope of

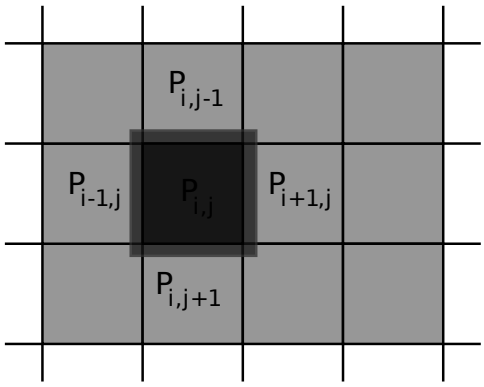


Figure 4: Domain splitting in the 2D case. Highlighted is the part of the domain for which processor $P_{i,j}$ is responsible, the part that is highlighted lighter is the *ghost area*.

this introduction, for details we refer to the work of the author[14, 15].

3.4 Periodic boundary conditions

For periodic boundary conditions (5) has to be solved on the torus $\mathbb{R}^3/\mathbb{Z}^3$, so the setting of explicit boundary conditions is not necessary. The basic numerical method given by Algorithm 3.2 needs only slight modifications, namely in the linked list data structure it has to be taken into account that the periodic images exist.

In contrast to the simple multigrid method described above, to simplify implementation the grid sizes should be chosen such that on level ℓ the grid has $n_\ell = 2^\ell$ grid points. The full-weighting operator given by Definition 2.7 and the linear operation operator given by Definition 2.8 are then easily adopted to take into account the grid points in the periodic images. Not that the linear system is singular with the constant vector, i.e. the vector consisting of ones, only, in the kernel.

4 Parallelization

The resulting method can be parallelized efficiently with the help of a standard domain splitting scheme[b], i.e. the physical domain is distributed onto the available processors. This situation is depicted for the 2D case in Figure 4. The physical domain Ω is mapped onto a threedimensional processor grid, such that processor $P_{i,j,k}$ contains subdomain $\Omega_{i,j,k}$, where

$$\bigcup_{i,jk} \Omega_{i,j,k} = \Omega, \qquad \Omega_{i,j,k} \cap \Omega_{i',j',k'} = \emptyset \text{ for } i \neq i', j \neq j', k \neq k'.$$

On each processor the domain is enlarged by a *ghost area*, such that the necessary data is available during computation. The ghost area for the multigrid method contains the grid

[b]The term *domain splitting* is used here to distinguish this data distribution scheme from the numerical method that is called *domain decomposition*.

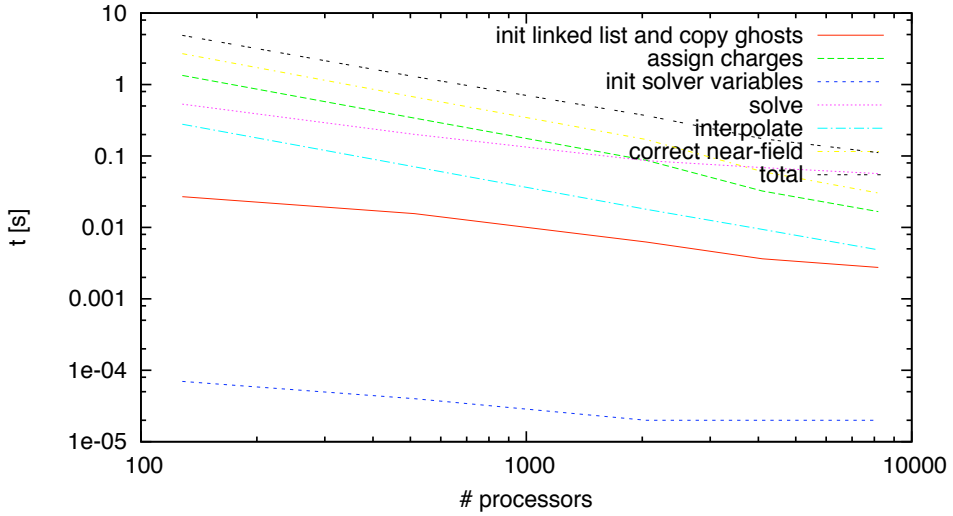


Figure 5: Timings for 2097152 particles on a $128 \times 128 \times 128$ grid on the Blue Gene/P system JUGENE in Jülich.

points needed to calculate the local matrix-vector product and to interpolate and restrict the data. The ghost area for the particle part of the method has to contain the particles located in the ghost area, the necessary data structures, i.e. the linked list entries, and the meshed solution of the PDE. These ghost areas do not have to be the same and they are updated whenever it is needed, i.e. the particles in the ghost area have to be exchanged at the beginning of the computation and the solution of the PDE has to be updated after it has been computed in the particle part of the method, in the multigrid method an update is necessary after each smoothing step and after applying restriction and interpolation. Using this parallelization scheme, parallelization of the basic numerical method is straightforward.

The parallelization of multigrid methods needs more attention. As the problem size is reduced on the coarser grids, at some point there are fewer variables, than processors to distribute the work to. There are basically two options to deal with this situation. First, it is possible to let the processors that are not assigned a variable stay idle during the computation on these coarser levels. The second option is to stop coarsening on this level and to solve the system directly on this level. The second option needs a parallel direct solver, while the first introduces a load-imbalance. As in practice the problems on these levels are relatively small compared to the finer levels, the first option is in many cases a decent choice. An overview over parallelization approaches for multigrid methods can be found in the survey paper of Chow et al.[16].

Our implementation of a multigrid-based method for the calculation of electrostatic interactions is based on the described 3D-domain splitting approach with idle processors on the coarser grids of the multigrid hierarchy. The timings of the different steps can be found in Figure 5. The method has been implemented using C99 and FORTRAN and the timing results are obtained for 2097152 particles on a $128 \times 128 \times 128$ grid on the Blue Gene/P system JUGENE in Jülich. The speedup and efficiency plots for this test can be

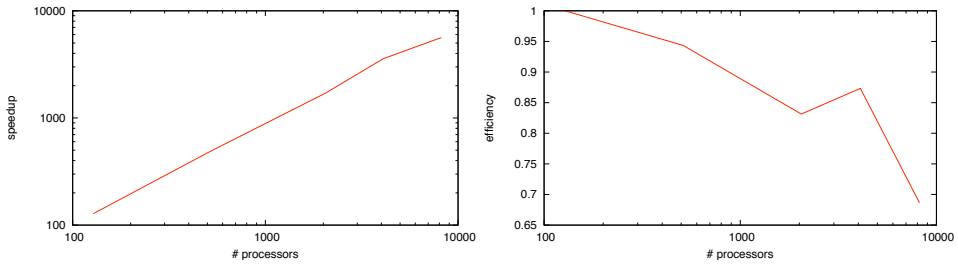


Figure 6: Speedup and efficiency for 2097152 particles on a $128 \times 128 \times 128$ grid on the Blue Gene/P system JUGENE in Jülich.

found in Figure 6. We like to note that on two racks of Blue Gene/P, i.e. 8192 processors, each processor is responsible for 256 grid points on the finest grid, only.

5 Conclusion and Outlook

The use of multigrid methods for the calculation of long-ranged interactions due to the electrostatic or gravitational potential is possible by reformulating the original problem as a PDE in a consistent way. Using this formulation the actual implementation of such a method is straightforward. One big advantage of the approach is that it can be parallelized efficiently. Further on, multigrid easily allows the treatment of more complicated geometries and problems with varying dielectric constants or the implicit simulation of solutes (see e.g. the article of Holst and Saied[17]).

Because the error of the method depends solely on the used discretization of the PDE and on the interpolation within the basic numerical method, the numerical precision needed can be enhanced by using e.g. more higher order discretization schemes, e.g. the ones proposed by Sutmann and Steffen[18].

References

1. R. P. Fedorenko, *The speed of convergence of one iterative process*, USSR Comp. Math. Math. Phys., **4**, no. 3, 227–235, 1964.
2. N. S. Bakhvalov, *On the convergence of a relaxation method with natural constraints on the elliptic operator*, USSR Comp. Math. Math. Phys., **6**, 101–135, 1966.
3. A. Brandt, *Multi-Level Adaptive Solutions to Boundary-Value Problems*, Math. Comp., **31**, no. 138, 333–390, April 1977.
4. W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, Philadelphia, 2000.
5. U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, 2001.
6. W. Hackbusch, *Multi-grid convergence theory*, in: Multigrid methods, W. Hackbusch and U. Trottenberg, (Eds.), vol. 960 of *Lecture Notes in Mathematics*, pp. 177–219, Springer-Verlag, Berlin, 1982.

7. W. Hackbusch, *Ein iteratives Verfahren zur schnellen Auflösung elliptischer Randwertprobleme*, Rep. 76-12, Institute for Applied Mathematics, University of Cologne, West Germany, Cologne, 1976.
8. W. Hackbusch, *On the convergence of a multi-grid iteration applied to finite element equations*, Rep. 77-8, Institute for Applied Mathematics, University of Cologne, West Germany, Cologne, 1977.
9. W. Hackbusch, *Convergence of multi-grid iterations applied to difference equations*, Math. Comp., **34**, 425–440, 1980.
10. W. Hackbusch, *On the convergence of multi-grid iterations*, Beiträge Numer. Math., **9**, 213–239, 1981.
11. W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.
12. W. Hackbusch, *Iterative solution of large sparse systems of equations*, Number 95 in Applied Mathematical Sciences. Springer-Verlag, New York, 1994.
13. R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*, Institute of Physics, Bristol, 1988.
14. M. Bolten, *Hierarchical grid coarsening for the solution of the Poisson equation in free space*, Electron. Trans. Numer. Anal., **29**, 70–80, 2008.
15. M. Bolten, *Multigrid methods for structured grids and their application in particle simulation*, PhD thesis, Bergische Universität Wuppertal, Wuppertal, 2008.
16. E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. M. Yang, "A survey of parallelization techniques for multigrid solvers", in: Parallel Processing for Scientific Computing, M. A. Heroux, P. Raghavan, and H. D. Simon, (Eds.), SIAM Series on Software, Environments, and Tools, chapter 10. SIAM, Philadelphia, 2006.
17. M. Holst and F. Saied, *Multigrid Solution of the Poisson-Boltzmann Equation*, J. Comp. Chem., **14**, no. 1, 105–113, 1993.
18. G. Sutmann and B. Steffen, *High-Order Compact Solvers for the Three Dimensional Poisson Equation*, J. Comp. Appl. Math., **187**, 142–170, 2006.

Particle Simulation Based on Nonequispaced Fast Fourier Transforms

Michael Pippig and Daniel Potts

Chemnitz University of Technology
Department of Mathematics
09107 Chemnitz, Germany
E-mail: {*michael.pippig, daniel.potts*}*@mathematik.tu-chemnitz.de*

The fast calculation of long-range interactions is a demanding problem in particle simulation. The main focus of our approach is the decomposition of the problem in building blocks and present efficient numerical realizations for these blocks. For that reason we recapitulate the fast Fourier transform at nonequispaced nodes and the fast summation method. We describe the application of these algorithms to the evaluation of long-range potentials and compare our methods with the existing fast multipole method.

Keywords and Phrases. fast discrete summation, fast Fourier transform at nonequispaced nodes, NFFT, fast multipole method, FMM, Ewald method, FFT-accelerated Ewald sum, particle-particle particle-mesh (P^3M), particle-mesh Ewald (PME), smooth particle-mesh Ewald (SPME).

1 Introduction

An important concern of applied mathematics is the development of efficient algorithms for frequently recurring problems. On the other hand one should decompose the practical problems such that one can use the efficient algorithms and consequently incorporate advanced implementations.

The aim of this tutorial is to decompose problems from particle simulation into building blocks. These blocks are the fast Fourier transform (FFT), the fast Fourier transform at nonequispaced nodes (NFFT) and the fast summation method.

In Section 2 we summarize the main ideas of the NFFT. A severe shortcoming of traditional Fourier schemes in recent applications is the need for equispaced sampling. During the last two decades that problem has attracted much attention. The nonequispaced fast Fourier transform[19] overcomes these disadvantages while keeping the number of floating point operations at $N \log N$. The concept of NFFT is the trade of exactness for efficiency. Instead of precise computations (up to machine precision for actual implementations), the proposed methods guarantee a user specified target accuracy. An early review of several algorithms for nonuniform Fourier transforms[3,6] has been given by A. Ware[32]. A unified approach to fast algorithms for the NFFT has been obtained by G. Steidl in[30]. The main idea is the use of a window function which is well localized in space as well as in frequency domain. Then one is able to use an approximation by translates of the scaled window function and estimate the approximation error, see the tutorial paper[29]. It became clear, that this approach is related to the gridding algorithm, which was known in image processing context and astrophysics years ago. Similar methods where used in particle simulation. A widely used implementation is available as part of the NFFT package[19] and is based on the FFTW[13].

In Section 3 we summarize the main ideas of the fast summation method based on NFFT. This method can be interpreted as nonequispaced convolution. For equispaced nodes the discrete convolution and its fast computation is typically realized by FFT exploiting the basic property $\mathrm{e}^{2\pi\mathrm{i}(y-x)} = \mathrm{e}^{2\pi\mathrm{i}y}\mathrm{e}^{-2\pi\mathrm{i}x}$. Following these lines, we propose to compute the "convolution at nonequispaced nodes" by Fourier methods as well, more precisely by the NFFT. This new method includes convolutions, e.g., with kernels of the form $1/\|x\|_2$.

In Section 4 we describe how to use the building blocks from Section 2 and Section 3 for particle simulation. Here we focus on the Coulomb potential for open and periodic systems. We remark that some FFT-accelerated Ewald[17,15] methods contain similar steps as the fast summation based on NFFT.

Finally Section 5 contains various different numerical examples, where we compare our methods with the fast multipole method.

2 Nonequispaced Fourier Transforms

This section summarizes the mathematical theory and ideas behind the NFFT based on[29,19–21]. Let the dimension $d \in \mathbb{N}$, the torus $\mathbb{T}^d := \mathbb{R}^d/\mathbb{Z}^d \sim [-\frac{1}{2}, \frac{1}{2})^d$ and the sampling set $\mathcal{X} := \{\mathbf{x}_j \in \mathbb{T}^d \colon j = 1, \ldots, M\}$ with $M \in \mathbb{N}$ be given. Furthermore, let the multi degree $\mathbf{N} = (N_0, N_1, \ldots, N_{d-1})^\top \in 2\mathbb{N}^d$ and the index set for possible frequencies $I_{\mathbf{N}} := \{-\frac{N_0}{2}, \ldots, \frac{N_0}{2} - 1\} \times \ldots \times \{-\frac{N_{d-1}}{2}, \ldots, \frac{N_{d-1}}{2} - 1\}$ be given. We define the space of d-variate trigonometric polynomials $T_{\mathbf{N}}$ of multi degree $\mathbf{N}$ by

$$T_{\mathbf{N}} := \operatorname{span}\left\{\mathrm{e}^{-2\pi\mathrm{i}\mathbf{k}\cdot} \colon \mathbf{k} \in I_{\mathbf{N}}\right\}.$$

The dimension of this space and hence the total number of Fourier coefficients is $N_\pi = N_0 \cdot \ldots \cdot N_{d-1}$. Note, that we abbreviate the inner product between the frequency $\mathbf{k}$ and the time/spatial node $\mathbf{x}$ by $\mathbf{kx} = \mathbf{k}^\top\mathbf{x} = k_0x_0 + k_1x_1 + \ldots + k_{d-1}x_{d-1}$. For clarity of presentation the multi index $\mathbf{k}$ addresses elements of vectors and matrices as well.

2.1 Nonequispaced discrete Fourier transform (NDFT)

For a finite number N_π of given Fourier coefficients $\hat{f}_{\mathbf{k}} \in \mathbb{C},\ \mathbf{k} \in I_{\mathbf{N}}$, one wants to evaluate the trigonometric polynomial

$$f(\mathbf{x}) := \sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{f}_{\mathbf{k}}\mathrm{e}^{-2\pi\mathrm{i}\mathbf{kx}} \in T_{\mathbf{N}} \tag{1}$$

at given nonequispaced nodes $\mathbf{x}_j \in \mathbb{T}^d$, $j = 1, \ldots, M$. Thus, our concern is the computation of the matrix vector product

$$\mathbf{f} = \mathbf{A}\hat{\mathbf{f}} \tag{2}$$

where

$$\mathbf{f} := (f(\mathbf{x}_j))_{j=1,\ldots,M}, \quad \mathbf{A} := \left(\mathrm{e}^{-2\pi\mathrm{i}\mathbf{kx}_j}\right)_{j=1,\ldots,M;\ \mathbf{k}\in I_{\mathbf{N}}}, \quad \hat{\mathbf{f}} := \left(\hat{f}_{\mathbf{k}}\right)_{\mathbf{k}\in I_{\mathbf{N}}}.$$

The straightforward algorithm for this matrix vector product, which is called NDFT, takes $\mathcal{O}(MN_\pi)$ arithmetical operations. A related matrix vector product is the adjoint NDFT

$$\hat{\mathbf{f}} = \mathbf{A}^{\mathsf{H}}\mathbf{f}, \qquad \hat{f}_{\mathbf{k}} = \sum_{j=1}^{M} f_j \mathrm{e}^{+2\pi \mathrm{i}\mathbf{k}\mathbf{x}_j},$$

where $\mathbf{A}^{\mathsf{H}} = \overline{\mathbf{A}}^{\top}$. Note furthermore, that the inversion formula $\mathbf{F}^{-1} = \mathbf{F}^{\mathsf{H}}$ for the (equispaced and normalized) Fourier matrix $\mathbf{F}$ does **not** hold in the general situation of arbitrary sampling nodes for the matrix $\mathbf{A}$.

2.2 Nonequispaced fast Fourier transform (NFFT)

The most successful approach for the fast computation of (2), cf.[6,3,30,29,11,14], is based on the usage of an oversampled FFT and a window function φ which is simultaneously localized in time/space and frequency. Basically, the scheme utilizes the convolution theorem in the following three informal steps:

1. deconvolve the trigonometric polynomial $f \in T_{\mathbf{N}}$ in (1) with a window function in frequency domain,
2. compute an oversampled FFT on the result of step 1.,
3. convolve the result of step 2. with the window function in time/spatial domain, i.e., evaluate this convolution at the nodes $\mathbf{x}_j$.

Throughout the rest of the paper we denote by $\sigma > 1$ the oversampling factor and by $n = \sigma N \in \mathbb{N}$ the FFT size. Furthermore, for $d > 1$ let $\boldsymbol{\sigma} = (\sigma_0, \ldots, \sigma_{d-1})^{\top} \in \mathbb{R}^d$, $\sigma_0, \ldots, \sigma_{d-1} > 1$, $\mathbf{n} = \boldsymbol{\sigma} \odot \mathbf{N}$, and $n_\pi = n_0 \cdot \ldots \cdot n_{d-1}$ denote the oversampling factor, the FFT size, and the total FFT size, respectively. For notational convenience we use the pointwise product $\boldsymbol{\sigma} \odot \mathbf{N} := (\sigma_0 N_0, \sigma_1 N_1, \ldots, \sigma_{d-1} N_{d-1},)^{\top}$ and the pointwise inverse $\mathbf{N}^{-1} := \left(\frac{1}{N_0}, \frac{1}{N_1}, \ldots, \frac{1}{N_{d-1}}\right)^{\top}$.

The window function

Starting with a window function $\varphi \in L_2(\mathbb{R})$, which is well localized in the time/spatial domain $\mathbb{R}$ and in the frequency domain $\mathbb{R}$, respectively, one assumes that its 1-periodic version $\tilde{\varphi}$, i.e.,

$$\tilde{\varphi}(x) := \sum_{r \in \mathbb{Z}} \varphi(x + r)$$

has an uniformly convergent Fourier series and is well localized in the time/spatial domain $\mathbb{T}$ and in the frequency domain $\mathbb{Z}$, respectively. Thus, the periodic window function $\tilde{\varphi}$ may be represented by its Fourier series

$$\tilde{\varphi}(x) = \sum_{k \in \mathbb{Z}} \hat{\varphi}(k) \mathrm{e}^{-2\pi \mathrm{i} k x}$$

with the Fourier coefficients

$$\hat{\varphi}(k) := \int_{\mathbb{T}} \tilde{\varphi}(x)\, \mathrm{e}^{+2\pi \mathrm{i} k x}\, \mathrm{d}x = \int_{\mathbb{R}} \varphi(x)\, \mathrm{e}^{+2\pi \mathrm{i} k x}\, \mathrm{d}x\,, \qquad k \in \mathbb{Z}.$$

We truncate this series at the FFT length n, which causes an aliasing error.

If φ is furthermore well localized in time/spatial domain $\mathbb{R}$, it can be truncated with truncation parameter $m \in \mathbb{N}$, $m \ll n$ and approximated by the function $\varphi \cdot \chi_{[-\frac{m}{n}, \frac{m}{n}]}$ which has compact support within the interval $[-\frac{m}{n}, \frac{m}{n}]$. Furthermore, the periodic window function can be approximated by the periodic version of the truncated window function. For $d > 1$, univariate window functions $\varphi_0, \ldots, \varphi_{d-1}$, and a node $\mathbf{x} = (x_0, \ldots, x_{d-1})^\top$ the multivariate window function is simply given by

$$\varphi(\mathbf{x}) := \varphi_0(x_0)\, \varphi_1(x_1) \ldots \varphi_{d-1}(x_{d-1})$$

and $\tilde{\varphi}(\mathbf{x}) = \sum_{\mathbf{r} \in \mathbb{Z}^d} \varphi(\mathbf{x} + \mathbf{r})$ again denotes the 1-periodic version; an immediate observation is

$$\hat{\varphi}(\mathbf{k}) := \int_{\mathbb{R}^d} \varphi(\mathbf{x})\, \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}}\, \mathrm{d}\mathbf{x} = \hat{\varphi}_0(k_0)\, \hat{\varphi}_1(k_1) \ldots \hat{\varphi}_{d-1}(k_{d-1})\,.$$

For a single truncation parameter $m \in \mathbb{N}$ the window function is truncated to the cube $\mathbf{n}^{-1} \odot [-m, m]^d$.

We follow the general approach of[30,29] and approximate the complex exponentials in the trigonometric polynomial (1) by

$$\mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{x}} \approx \frac{1}{n_\pi \hat{\varphi}(\mathbf{k})} \sum_{\mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x})} \tilde{\varphi}\left(\mathbf{x} - \mathbf{n}^{-1} \odot \mathbf{l}\right) \mathrm{e}^{-2\pi \mathrm{i} \left(\mathbf{n}^{-1} \odot \mathbf{l}\right) \mathbf{k}} \tag{3}$$

where the set

$$I_{\mathbf{n},m}(\mathbf{x}) := \{\mathbf{l} \in I_{\mathbf{n}} \colon \mathbf{n} \odot \mathbf{x} - m\mathbf{1} \leq \mathbf{l} \leq \mathbf{n} \odot \mathbf{x} + m\mathbf{1}\}$$

collects these indices where the window function is mostly concentrated (the inequalities have to be fulfilled modulo $\mathbf{n}$ and for each component). After changing the order of summation in (1) we obtain for $\mathbf{x}_j \in \mathbb{T}^d$, $j = 1, \ldots, M$, the approximation

$$f(\mathbf{x}_j) \approx \sum_{\mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x}_j)} \left(\sum_{\mathbf{k} \in I_{\mathbf{N}}} \frac{\hat{f}_{\mathbf{k}}}{n_\pi \hat{\varphi}(\mathbf{k})} \mathrm{e}^{-2\pi \mathrm{i} \left(\mathbf{n}^{-1} \odot \mathbf{l}\right) \mathbf{k}} \right) \tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right)\,.$$

This causes a truncation and an aliasing error, see[29,26] for details. As can be readily seen, after an initial deconvolution step, the expression in brackets can be computed via a d-variate FFT of total size n_π. The final step consists of the evaluation of sums having at most $(2m+1)^d$ terms where the window function is sampled only in the neighborhood of the node $\mathbf{x}_j$.

The algorithm and its matrix notation

The proposed scheme reads in matrix vector notation as

$$\mathbf{A}\hat{\mathbf{f}} \approx \mathbf{B}\mathbf{F}\mathbf{D}\hat{\mathbf{f}},$$

where $\mathbf{B}$ denotes the real $M \times n_\pi$ sparse matrix

$$\mathbf{B} := \left(\tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right) \cdot \chi_{I_{\mathbf{n},m}(\mathbf{x}_j)}(\mathbf{l})\right)_{j=1,\ldots,M;\, \mathbf{l} \in I_{\mathbf{n}}}, \tag{4}$$

where $\mathbf{F}$ is the d-variate Fourier matrix of size $n_\pi \times n_\pi$,

$$\mathbf{F} := \left(\mathrm{e}^{-2\pi \mathrm{i}\left(\mathbf{n}^{-1} \odot \mathbf{l}\right)\mathbf{k}}\right)_{\mathbf{l} \in I_{\mathbf{n}};\, \mathbf{k} \in I_{\mathbf{n}}}, \tag{5}$$

and where $\mathbf{D}$ is the real $n_\pi \times N_\pi$ 'diagonal' matrix

$$\mathbf{D} := \bigotimes_{t=0}^{d-1} \left(\mathbf{O}_t \mid \operatorname{diag}\left(\frac{1}{n_t \hat{\varphi}_t(k_t)}\right)_{k_t \in I_{N_t}} \mid \mathbf{O}_t\right)^\top \tag{6}$$

with zero matrices $\mathbf{O}_t$ of size $N_t \times \frac{n_t - N_t}{2}$. Obviously, the approximate matrix splitting can be applied to the adjoint matrix as $\mathbf{A}^\mathsf{H} \approx \mathbf{D}^\top \mathbf{F}^\mathsf{H} \mathbf{B}^\top$, where the multiplication with the sparse matrix $\mathbf{B}^\top$ is implemented in a 'transposed' way, summation as outer loop and only using the index sets $I_{\mathbf{n},m}(\mathbf{x}_j)$.

Algorithm 2.1 NFFT

Input: $d \in \mathbb{N}$ dimension,
$M \in \mathbb{N}$ number of nodes, nodes $\mathbf{x}_j \in [-\frac{1}{2}, \frac{1}{2})^d,\ j = 1, \ldots, M$,
$\mathbf{N} \in 2\mathbb{N}^d$ multi degree, Fourier coefficients $\hat{f}_\mathbf{k} \in \mathbb{C},\ \mathbf{k} \in I_\mathbf{N}$,
$\boldsymbol{\sigma} \in \mathbb{R}^d$ oversampling factor with $\sigma_t > 1$, $t = 0, \ldots, d-1$,
$m \in \mathbb{N}$ window size of $\tilde{\varphi}$.

1. For $\mathbf{k} \in I_\mathbf{N}$ compute

$$\hat{g}_\mathbf{k} := \frac{\hat{f}_\mathbf{k}}{n_\pi \hat{\varphi}(\mathbf{k})}.$$

2. For $\mathbf{l} \in I_\mathbf{n}$ with $\mathbf{n} = \boldsymbol{\sigma} \odot \mathbf{N}$ compute by d-variate (forward) FFT

$$g_\mathbf{l} := \sum_{\mathbf{k} \in I_\mathbf{N}} \hat{g}_\mathbf{k}\, \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\left(\mathbf{n}^{-1} \odot \mathbf{l}\right)}.$$

3. For $j = 1, \ldots, M$ compute

$$s_j := \sum_{\mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x}_j)} g_\mathbf{l}\, \tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right).$$

Output: approximate values $s_j \approx f(\mathbf{x}_j)$, $j = 1, \ldots, M$.

Evaluation techniques for window functions

To keep the aliasing error and the truncation error small, several univariate functions φ with good localization in time and frequency domain were proposed. For an oversampling factor $\sigma > 1$, a degree $N \in 2\mathbb{N}$, the FFT length $n = \sigma N$, and a cut-off parameter $m \in \mathbb{N}$, the following window functions are considered:

Algorithm 2.2 NFFT$^{\vdash}$

Input: $d \in \mathbb{N}$ dimension,
$M \in \mathbb{N}$ number of nodes,
nodes $\mathbf{x}_j \in [-\frac{1}{2}, \frac{1}{2})^d$, coefficients $f_j \in \mathbb{C}$, $j = 1, \ldots, M$,
$\mathbf{N} \in 2\mathbb{N}^d$ multi degree,
$\boldsymbol{\sigma} \in \mathbb{R}^d$ oversampling factor with $\sigma_t > 1$, $t = 0, \ldots, d-1$,
$m \in \mathbb{N}$ window size of $\tilde{\varphi}$.

1. For $\mathbf{l} \in I_{\mathbf{n}}$ with $\mathbf{n} = \boldsymbol{\sigma} \odot \mathbf{N}$ compute

$$g_{\mathbf{l}} := \sum_{\substack{j=1 \\ \mathbf{l} \in I_{\mathbf{n},m}(\mathbf{x}_j)}}^{M} f_j \, \tilde{\varphi}\left(\mathbf{x}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right).$$

2. For $\mathbf{k} \in I_{\mathbf{n}}$ compute by d-variate (backward) FFT

$$\hat{g}_{\mathbf{k}} := \sum_{\mathbf{l} \in I_{\mathbf{n}}} g_{\mathbf{l}} \, \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k}\left(\mathbf{n}^{-1} \odot \mathbf{l}\right)}.$$

3. For $\mathbf{k} \in I_{\mathbf{N}}$ compute

$$\hat{s}_{\mathbf{k}} := \frac{\hat{g}_{\mathbf{k}}}{n_\pi \hat{\varphi}(\mathbf{k})}.$$

Output: approximate values $\hat{s}_{\mathbf{k}} \approx \hat{f}_{\mathbf{k}}$, $\mathbf{k} \in I_{\mathbf{N}}$.

1. for a shape parameter $b = \frac{2\sigma}{2\sigma-1}\frac{m}{\pi}$ the dilated *Gaussian window*[6,30,5]

$$\begin{aligned} \varphi(x) &= (\pi b)^{-1/2} \, \mathrm{e}^{-(nx)^2/b}, \\ \hat{\varphi}(k) &= \frac{1}{n} \, \mathrm{e}^{-(\frac{\pi k}{n})^2 b}, \end{aligned} \tag{7}$$

2. for M_{2m} denoting the centered cardinal B-Spline of order $2m$ the dilated *B-Spline window*[3,30]

$$\begin{aligned} \varphi(x) &= M_{2m}(nx), \\ \hat{\varphi}(k) &= \frac{1}{n} \begin{cases} 1 & \text{for } k = 0, \\ \operatorname{sinc}^{2m}\left(\frac{k\pi}{n}\right) & \text{otherwise}, \end{cases} \end{aligned} \tag{8}$$

3. the dilated *Sinc window*[26]

$$\begin{aligned} \varphi(x) &= \frac{N\,(2\sigma-1)}{2m} \operatorname{sinc}^{2m}\left(\frac{\pi N x\,(2\sigma-1)}{2m}\right), \\ \hat{\varphi}(k) &= M_{2m}\left(\frac{2mk}{(2\sigma-1)\,N}\right), \end{aligned} \tag{9}$$

with $\operatorname{sinc}(x) := \sin(x) \,/\, x$ for $x \neq 0$ and $\operatorname{sinc}(0) := 1$

4. and for a shape parameter $b = \pi(2 - \frac{1}{\sigma})$ the dilated *Kaiser-Bessel window*[27]

$$\varphi(x) = \frac{1}{\pi} \begin{cases} \dfrac{\sinh(b\sqrt{m^2 - n^2x^2})}{\sqrt{m^2 - n^2x^2}} & \text{for } |x| \leq \frac{m}{n}, \\ \dfrac{\sin(b\sqrt{n^2x^2 - m^2})}{\sqrt{n^2x^2 - m^2}} & \text{otherwise,} \end{cases} \tag{10}$$

$$\hat{\varphi}(k) = \begin{cases} \frac{1}{n} I_0\left(m\sqrt{b^2 - (2\pi k/n)^2}\right) & \text{for } k = -n\left(1 - \frac{1}{2\sigma}\right), \ldots, n\left(1 - \frac{1}{2\sigma}\right), \\ 0 & \text{otherwise,} \end{cases}$$

where I_0 denotes the modified zero-order Bessel function.

Note, that the latter two have compact support in frequency domain while the second one has compact support in time domain. Further references on the usage of (generalized) Kaiser-Bessel window functions include[18,23], where some authors prefer to interchange the role of time and frequency domain. For these univariate window functions φ, the approximation error of Algorithm 2.1 obeys

$$|f(x_j) - s_j| \leq C_{\sigma,m} \|\hat{\mathbf{f}}\|_1 ,$$

where

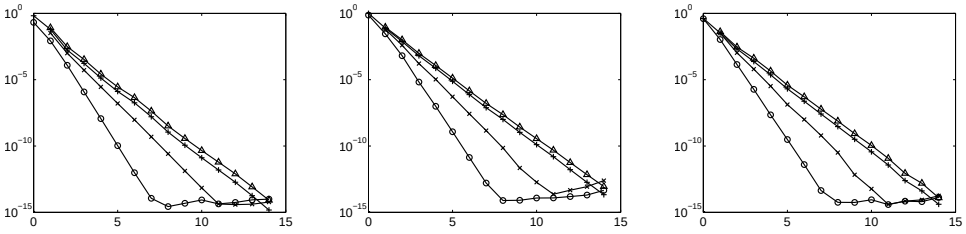
$$C_{\sigma,m} := \begin{cases} 4\,\mathrm{e}^{-m\pi(1-1/(2\sigma-1))} & \text{for (7), cf.[30],} \\ 4\left(\frac{1}{2\sigma-1}\right)^{2m} & \text{for (8), cf.[30],} \\ \frac{1}{m-1}\left(\frac{2}{\sigma^{2m}} + \left(\frac{\sigma}{2\sigma-1}\right)^{2m}\right) & \text{for (9), cf.[26],} \\ 4\pi\left(\sqrt{m} + m\right)\sqrt[4]{1 - \frac{1}{\sigma}}\,\mathrm{e}^{-2\pi m\sqrt{1-1/\sigma}} & \text{for (10), cf.[26].} \end{cases}$$

Thus, for fixed $\sigma > 1$, the approximation error introduced by the NFFT decays exponentially with the number m of terms in sum (3). Using the tensor product approach, the error estimates above have been generalized for the multivariate setting in[7,5].

In summary, the whole algorithm for the fast approximate computation of (1) consists of the computation of N_π multiplications, the computation of a d-variate FFT of total size n_π and the sparse summations. Therefore it requires $\mathcal{O}(n_\pi + n_\pi \log n_\pi + (2m+1)^d M) = \mathcal{O}(n_\pi \log n_\pi + m^d M)$ arithmetic operations.

In[21] we suggest different methods for the compressed storage and application of matrix $\mathbf{B}$, which are all available within our NFFT library[19] by choosing particular flags in a simple way during the initialization phase. These methods include fully precomputed window function, tensor product based precomputation, linear interpolation from a lookup table, fast Gaussian gridding (see also[14]) and no precomputation of the window function. The choice of precomputation does not yield a different asymptotic performance but rather yields a lower constant in the amount of computation.

Finally we remark, that similar approximations of the exponentials as in (3) are used in various particle methods, see [4, Formula (24)], [8, Formula (3.5)] or [15, Formula (7.57)]. However, mainly splines are used as window functions. We note that, e.g., Kaiser-Bessel functions lead to better results, while using the same window size m, see Figure 1 and[20],



(a) $d = 1$, $N = 2^{12}$. (b) $d = 2$, $N = 2^6$. (c) $d = 3$, $N = 2^4$.

Figure 1: Error E_∞ for increasing cut-off parameter $m = 0, \ldots, 14$ and $d = 1, 2, 3$. In each case, the degree N was chosen to be equal along each dimension such that $|I_{\mathbf{N}}| = 2^{12}$. We fixed the oversampling factor $\sigma = 2$ and $M = 10000$. Shown are results for the Kaiser-Bessel ($\circ$), the Sinc ($\times$), the B-spline ($+$), and the Gaussian window function ($\triangle$).

where the accuracy is measured by

$$E_\infty := \max_{1 \le j \le M} |f_j - s_j| / \sum_{\mathbf{k} \in I_{\mathbf{N}}} |\hat{f}_{\mathbf{k}}| \ .$$

For further NFFT approaches see [20, Appendix D].

3 Fast Summation Algorithms

We are interested in the fast evaluation of sums

$$h(\mathbf{y}) := \sum_{l=1}^{L} \alpha_l \mathcal{K}(\mathbf{y} - \mathbf{x}_l) = \sum_{l=1}^{L} \alpha_l K(\|\mathbf{y} - \mathbf{x}_l\|_2), \tag{11}$$

at M different target nodes $\mathbf{y}_j$, $j = 1, \ldots, M$, where $\|\mathbf{x}\|_2 := (x_0^2 + \ldots + x_{d-1}^2)^{1/2}$ denotes the Euclidean norm. This approach was suggested in[27,28,9]. The original idea for our algorithm came from the consideration of (11) for equispaced source nodes $\mathbf{x}_l$ and target nodes $\mathbf{y}_j$. In this case we have simply to compute the multiplication of a vector with a Toeplitz matrix or a block–Toeplitz–Toeplitz–block matrix in the multivariate setting. The standard algorithm to do this uses the FFT, see [27, Section2]. Here we propose the summation algorithm based on NFFT for arbitrary distributed source and target nodes.

The kernel function K is in general a non-periodic function, while the use of Fourier methods requires to replace K by a periodic version. Without loss of generality we may assume that the source and target nodes are scaled, such that $\|\mathbf{x}_l\|_2, \|\mathbf{y}_j\|_2 < \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2}$ and consequently $\|\mathbf{y}_j - \mathbf{x}_l\|_2 < \frac{1}{2} - \varepsilon_{\mathrm{B}}$. The parameter $\varepsilon_{\mathrm{B}} > 0$, which we specify later, guarantees that K has to be evaluated only at nodes in the ball with radius $\frac{1}{2} - \varepsilon_{\mathrm{B}}$. This simplifies the later consideration of a 1-periodic version of K.

We deal with kernels $\mathcal{K}$ which are C^∞ except for the origin, where $\mathcal{K}$ or its derivatives may have singularities. Examples of such kernels are

$$\|\mathbf{x}\|_2^2 \log \|\mathbf{x}\|_2, \ \log \|\mathbf{x}\|_2 \quad \text{and} \quad \frac{1}{\|\mathbf{x}\|_2^\beta} \quad (\beta \in \mathbb{N}).$$

For the sake of simplicity we define $\mathcal{K}(\mathbf{0}) = 0$ whenever a singularity appears at the origin. This enables us to evaluate (11) at source nodes which coincide with a target node. Of course, our algorithm can be modified for other kernels frequently used in the approximation by radial basis functions, e.g., the Gaussian[22] or the (inverse) multiquadric[10] $(x^2 + c^2)^{\pm 1/2}$. Our algorithm, in particular our regularization procedure, is simply structured, can easily be adapted to different kernels K and requires $\mathcal{O}(L \log \sqrt[d]{L} + M)$ or $\mathcal{O}(M \log \sqrt[d]{M} + L)$ arithmetic operations for uniformly distributed source nodes $\mathbf{x}_l$ or target nodes $\mathbf{y}_j$, respectively.

Beyond a special treatment of $\mathcal{K}$ near the boundary, we have to be concerned about the singularity of $\mathcal{K}$ at the origin. We regularize $\mathcal{K}$ near $\mathbf{0}$ and near the boundary $\{\mathbf{x} \in \mathbb{T}^d : \|\mathbf{x}\|_2 = \frac{1}{2}\}$ as follows

$$\mathcal{K}_{\mathrm{R}}(\mathbf{x}) := \begin{cases} T_{\mathrm{I}}(\|\mathbf{x}\|_2) & \text{if} \quad \|\mathbf{x}\|_2 \le \varepsilon_{\mathrm{I}}, \\ T_{\mathrm{B}}(\|\mathbf{x}\|_2) & \text{if} \quad \frac{1}{2} - \varepsilon_{\mathrm{B}} < \|\mathbf{x}\|_2 < \frac{1}{2}, \\ T_{\mathrm{B}}(\frac{1}{2}) & \text{if} \quad \frac{1}{2} \le \|\mathbf{x}\|_2, \\ K(\|\mathbf{x}\|_2) & \text{otherwise}, \end{cases} \tag{12}$$

where $0 < \varepsilon_{\mathrm{I}} < \frac{1}{2} - \varepsilon_{\mathrm{B}} < \frac{1}{2}$. The functions T_{I} and T_{B} will be chosen such that $\mathcal{K}_{\mathrm{R}}$ is in the Sobolev space $H^p(\mathbb{T}^d)$ for an appropriate parameter $p \in \mathbb{N}$. Several regularizations of $\mathcal{K}$ are possible, e.g., by algebraic polynomials, splines or trigonometric polynomials, see[10].

Here we focus our attention on two point Taylor interpolation, i.e., we are interested in the unique polynomial of degree $2p-1$ which satisfies the interpolation conditions

$$P^{(j)}(m-r) = a_j, \quad P^{(j)}(m+r) = b_j, \quad j = 0, \dots, p-1 \tag{13}$$

at the endpoints of an interval $[m-r, m+r], r > 0$.

Lemma 3.1. *(see [9, Proposition 2.2]) For given a_j, b_j $(j = 0, \dots, p-1)$ there exists a unique polynomial P of degree $2p-1$ which satisfies* (13)*. With $y := \frac{z-m}{r}$ this polynomial can be written as*

$$P(z) = \frac{1}{2^p} \sum_{j=0}^{p-1} \sum_{k=0}^{p-1-j} \binom{p-1+k}{k} \frac{r^j}{j! 2^k} \Big((1+y)^{j+k}(1-y)^p a_j + (1-y)^{j+k}(1+y)^p (-1)^j b_j \Big). \tag{14}$$

Lemma 3.2. *The derivative of the polynomial P from* (14) *is given by*

$$\begin{aligned} P'(z) = & \frac{1}{2^p} \sum_{j=0}^{p-1} \sum_{\substack{k=0 \\ j+k \neq 0}}^{p-1-j} \binom{p-1+k}{k} \frac{r^{j-1}}{j! 2^k} \\ & \Big((1+y)^{j+k-1}(1-y)^{p-1} \big[(j+k)(1-y) - p(1+y) \big] a_j \\ & + (1-y)^{j+k-1}(1+y)^{p-1} \big[(j+k)(1+y) - p(1-y) \big] (-1)^{j-1} b_j \Big). \end{aligned} \tag{15}$$

We exploit (14) with $a_j = K^{(j)}(\varepsilon_{\mathrm{I}}), b_j = (-1)^j a_j, j = 0, \dots, p-1, m = 0, r = 2\varepsilon_{\mathrm{I}}$ to obtain T_{I} and with $a_j = K^{(j)}(\frac{1}{2} - \varepsilon_{\mathrm{B}}), b_j = \delta_{0,j} K(\frac{1}{2}), j = 0, \dots, p-1, m = \frac{1-\varepsilon_{\mathrm{B}}}{2}, r = \varepsilon_{\mathrm{B}}$ to obtain T_{B}.

Next we approximate the smooth function $\mathcal{K}_{\mathrm{R}}$ by the Fourier series

$$\mathcal{K}_{\mathrm{R}} \approx \mathcal{K}_{\mathrm{RF}}(\mathbf{x}) := \sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\mathbf{x}}, \tag{16}$$

where

$$\hat{b}_{\mathbf{k}} := \frac{1}{N_\pi} \sum_{\mathbf{l}\in I_{\mathbf{N}}} \mathcal{K}_{\mathrm{R}}(\mathbf{N}^{-1} \odot \mathbf{l}) \mathrm{e}^{+2\pi \mathrm{i}(\mathbf{N}^{-1}\odot \mathbf{l})\mathbf{k}}, \quad \mathbf{k} \in I_{\mathbf{N}} \,. \tag{17}$$

Then our original kernel splits as

$$\mathcal{K} = (\mathcal{K} - \mathcal{K}_{\mathrm{R}}) + (\mathcal{K}_{\mathrm{R}} - \mathcal{K}_{\mathrm{RF}}) + \mathcal{K}_{\mathrm{RF}} = \mathcal{K}_{\mathrm{NE}} + \mathcal{K}_{\mathrm{ER}} + \mathcal{K}_{\mathrm{RF}}, \tag{18}$$

where $\mathcal{K}_{\mathrm{NE}} := \mathcal{K} - \mathcal{K}_{\mathrm{R}}$ and $\mathcal{K}_{\mathrm{ER}} := \mathcal{K}_{\mathrm{R}} - \mathcal{K}_{\mathrm{RF}}$. Since $\mathcal{K}_{\mathrm{R}}$ is smooth, the approximation error $\mathcal{K}_{\mathrm{ER}}$ of its Fourier approximation $\mathcal{K}_{\mathrm{RF}}$ should be small. We neglect this error and approximate h in (11) by

$$h(\mathbf{y}) \approx \tilde{h}(\mathbf{y}) := h_{\mathrm{NE}}(\mathbf{y}) + h_{\mathrm{RF}}(\mathbf{y}),$$

where

$$h_{\mathrm{NE}}(\mathbf{y}) := \sum_{l=1}^{L} \alpha_l \mathcal{K}_{\mathrm{NE}}(\mathbf{y} - \mathbf{x}_l) \,, \tag{19}$$

$$h_{\mathrm{RF}}(\mathbf{y}) := \sum_{l=1}^{L} \alpha_l \mathcal{K}_{\mathrm{RF}}(\mathbf{y} - \mathbf{x}_l) \,. \tag{20}$$

Instead of h we evaluate $\tilde{h}$ at the target nodes $\mathbf{y}_j$. If either the source nodes $\mathbf{x}_k$ or the target nodes $\mathbf{y}_j$ are "sufficiently uniformly distributed" this can indeed be done in a fast way, namely:

Near field computation (19)

By definition (12), the function $\mathcal{K}_{\mathrm{NE}}$ has a small support contained in the ball of radius ε_{I} around $\mathbf{0}$ and in the neighborhood of the boundary. The boundary is not interesting for us since $\|\mathbf{y}_j - \mathbf{x}_l\|_2 \leq 1/2 - \varepsilon_{\mathrm{B}}$. To achieve the desired complexity of our algorithm we suppose that either the L source nodes $\mathbf{x}_l$ or the M target nodes $\mathbf{y}_j$ are "sufficiently uniformly distributed", i.e., we suppose that there exists a small constant $\nu \in \mathbb{N}$ such that every ball of radius ε_{I} contains at most ν of the nodes $\mathbf{x}_l$ or of the nodes $\mathbf{y}_j$, respectively. This implies that ε_{I} depends linearly on $1/\sqrt[d]{L}$, respectively $1/\sqrt[d]{M}$. In the following, we restrict our attention to the case

$$\varepsilon_{\mathrm{I}} \sim \sqrt[d]{\frac{\nu}{L}}.$$

Then for fixed $\mathbf{y}_j$ the sum (19) contains not more than ν terms so that its evaluation at M target nodes $\mathbf{y}_j$ requires only $\mathcal{O}(\nu M)$ arithmetic operations.

We remark that also $\mathcal{O}(\log \sqrt[d]{M})$, respectively $\mathcal{O}(\log \sqrt[d]{L})$ nodes instead of $\mathcal{O}(1)$ nodes per ball will keep a complexity of $\mathcal{O}(M \log \sqrt[d]{M} + L \log \sqrt[d]{L})$ of our whole algorithm.

Far field summation (20) **by NFFT$^{\intercal}$/NFFT**

Substituting (16) for $\mathcal{K}_{\mathrm{RF}}$ we obtain

$$h_{\mathrm{RF}}(\mathbf{y}_j) \;=\; \sum_{l=1}^{L} \alpha_l \sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}(\mathbf{y}_j-\mathbf{x}_l)} \;=\; \sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \left(\sum_{l=1}^{L} \alpha_l \mathrm{e}^{+2\pi \mathrm{i}\mathbf{k}\mathbf{x}_l} \right) \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\mathbf{y}_j}.$$

The expression in the inner brackets can be computed by a d-variate NFFT$^{\intercal}$ of total size N_π. This is followed by N_π multiplications with $\hat{b}_{\mathbf{k}}$ and completed by a d-variate NFFT of total size N_π to compute the outer sum with the complex exponentials. If m is the cut-off parameter and $\sigma = (2)_{t=0,\ldots,d-1}$ the oversampling factor of the NFFT$^{\intercal}$/NFFT, then the proposed evaluation of h_{RF} at the nodes $\mathbf{y}_j$, $j = 1,\ldots,M$ requires $\mathcal{O}(m^d(L+M) + \sigma_\pi N_\pi \log(\sigma_\pi N_\pi))$ arithmetic operations. The relation between M, L and $\mathbf{N}$ is determined by the approximation error of the algorithm and is discussed in detail in[27,28,10].

Note, we can avoid the error of $\mathcal{K}_{\mathrm{ER}}$ in the near field in (18) by splitting kernel function $\mathcal{K}$ as

$$\mathcal{K} = (\mathcal{K} - \mathcal{K}_{\mathrm{RF}}) \;+\; \mathcal{K}_{\mathrm{RF}} = \mathcal{K}_{\mathrm{NE}} \;+\; \mathcal{K}_{\mathrm{ER}} \;+\; \mathcal{K}_{\mathrm{RF}}, \tag{21}$$

and setting $\mathcal{K}_{\mathrm{NE}} := (\mathcal{K} - \mathcal{K}_{\mathrm{RF}})\, \chi_{\{\|\cdot\|\le\varepsilon_I\}}$, $\mathcal{K}_{\mathrm{ER}} := \mathcal{K} - \mathcal{K}_{\mathrm{RF}} - \mathcal{K}_{\mathrm{NE}}$. The first splitting (18) is preferable, if we are able to evaluate the near field regularization T_I in a fast way. If T_I can not be computed in a fast way but the Fourier coefficients $\hat{b}_{\mathbf{k}}$ are given, the splitting (21) should be used.

The algorithm and its matrix notation

The proposed scheme reads in matrix vector notation as

$$\mathbf{K}\alpha \approx \mathbf{B}_{\mathbf{y}}\mathbf{F}\mathbf{D}\mathbf{D}_{\mathbf{b}}\mathbf{D}^{\top}\mathbf{F}^{\mathrm{H}}\mathbf{B}_{\mathbf{x}}^{\top}\alpha + \mathbf{K}_{\mathrm{NE}}\alpha\,, \tag{22}$$

where $\mathbf{B}_{\mathbf{x}}$ denotes the real $L \times n_\pi$ sparse matrix depending on the source nodes $\mathbf{x}_l$, $l = 1,\ldots,L$ as given in (4)

$$\mathbf{B}_{\mathbf{x}} := \left(\tilde{\varphi}\left(\mathbf{x}_l - \mathbf{n}^{-1} \odot \mathbf{l}\right) \cdot \chi_{I_{\mathbf{n},m}(\mathbf{x}_l)}(\mathbf{l})\right)_{l=1,\ldots,L;\, \mathbf{l}\in I_{\mathbf{n}}},$$

$\mathbf{B}_{\mathbf{y}}$ denotes the real $M \times n_\pi$ sparse matrix depending on the target nodes $\mathbf{y}_j$, $j = 1,\ldots,M$ as given in (4)

$$\mathbf{B}_{\mathbf{y}} := \left(\tilde{\varphi}\left(\mathbf{y}_j - \mathbf{n}^{-1} \odot \mathbf{l}\right) \cdot \chi_{I_{\mathbf{n},m}(\mathbf{y}_j)}(\mathbf{l})\right)_{j=1,\ldots,M;\, \mathbf{l}\in I_{\mathbf{n}}},$$

$\mathbf{F}$ is the d-variate Fourier matrix of size $n_\pi \times n_\pi$ given in (5), $\mathbf{D}$ is the real $n_\pi \times N_\pi$ 'diagonal' matrix given in (6), which contains the Fourier coefficients of the window function, $\mathbf{D}_{\mathbf{b}} = \mathrm{diag}(\hat{b}_{\mathbf{k}})_{\mathbf{k}\in I_{\mathbf{N}}}$ contains the Fourier coefficients of the regularized kernel $\mathcal{K}_{\mathrm{R}}$ given in (17). Furthermore, $\mathbf{K}_{\mathrm{NE}}$ contains the near field correction (19). From the representation (22) we see, that $\mathbf{F}\mathbf{D}\mathbf{F}^{\mathrm{H}}$ with a diagonal matrix $\mathbf{D}$ is well known as fast realization of a convolution on a mesh. The matrix $\mathbf{B}^{\top}$ is used to smear the charge density onto a grid, then one can use a convolution on a mesh and finally a back-interpolation via $\mathbf{B}$.

In summary we obtain Algorithm 3.1 for the fast evaluation of (11).

Algorithm 3.1 Fast sum

Input: $d \in \mathbb{N}$ dimension,
$p \in \mathbb{N}$ smoothness of regularized kernel K_{R},
$\mathbf{N} \in 2\mathbb{N}^d$ multi degree,
$\varepsilon_{\mathrm{I}} > 0$ nearfield size,
$\varepsilon_{\mathrm{B}} > 0$ boundary size,
$L \in \mathbb{N}$ number of source nodes,
$M \in \mathbb{N}$ number of target nodes,
source nodes $\mathbf{x}_l \in [-\frac{1}{4} + \frac{\varepsilon_{\mathrm{B}}}{2}, \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2})^d$, $l = 1, \ldots, L$,
target nodes $\mathbf{y}_j \in [-\frac{1}{4} + \frac{\varepsilon_{\mathrm{B}}}{2}, \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2})^d$, $j = 1, \ldots, M$,
coefficients $\alpha_l \in \mathbb{C},\ l = 1, \ldots, L$.

Precomputation:

i) Computation of $(\hat{b}_{\mathbf{k}})_{\mathbf{k} \in I_{\mathbf{N}}}$ by (17) and (12).

ii) Computation of $K_{\mathrm{NE}}(\mathbf{y}_j - \mathbf{x}_l)$ for all $j = 1, \ldots, M$ and $l \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)$, where $I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j) := \{l \in \{1, \ldots, L\} \colon \|\mathbf{y}_j - \mathbf{x}_l\|_2 < \varepsilon_{\mathrm{I}}\}$.

1. For $\mathbf{k} \in I_{\mathbf{N}}$ compute by Algorithm 2.2 the d-variate NFFT$^{\vdash}$

$$\hat{a}_{\mathbf{k}} := \sum_{l=1}^{L} \alpha_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l}.$$

2. For $\mathbf{k} \in I_{\mathbf{N}}$ compute the products

$$\hat{d}_{\mathbf{k}} := \hat{a}_{\mathbf{k}} \hat{b}_{\mathbf{k}}.$$

3. For $j = 1, \ldots, M$ compute by Algorithm 2.1 the d-variate NFFT

$$h_{\mathrm{RF}}(\mathbf{y}_j) := \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{d}_{\mathbf{k}} \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{y}_j}.$$

4. For $j = 1, \ldots, M$ compute the near field sums

$$h_{\mathrm{NE}}(\mathbf{y}_j) = \sum_{l \in I_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)} \alpha_l \mathcal{K}_{\mathrm{NE}}(\mathbf{y}_j - \mathbf{x}_l).$$

5. For $j = 1, \ldots, M$ compute the near field corrections

$$\tilde{h}(\mathbf{y}_j) = h_{\mathrm{NE}}(\mathbf{y}_j) + h_{\mathrm{RF}}(\mathbf{y}_j).$$

Output: approximate values $\tilde{h}(\mathbf{y}_j) \approx h(\mathbf{y}_j),\ j = 1, \ldots, M$.

Generalization to cuboidal domains

Whenever the restrictions $\|\mathbf{x}_l\|_2, \|\mathbf{y}_j\|_2 < \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2}$ are not fulfilled we must scale the nodes. We now explain the resulting changes to Algorithm 3.1. In order to handle node distributions with unequal dimensional extend we introduce $\mathbf{s} = (s_0, s_1, \ldots, s_{d-1})^{\top} \in \mathbb{N}^d$

as a component wise scaling such that $\|\mathbf{s}^{-1} \odot \mathbf{x}_l\|_2, \|\mathbf{s}^{-1} \odot \mathbf{y}_j\|_2 < \frac{1}{4} - \frac{\varepsilon_\mathrm{B}}{2}$ and therefore $\|\mathbf{s}^{-1} \odot (\mathbf{y}_j - \mathbf{x}_l)\| < \frac{1}{2} - \varepsilon_\mathrm{B}$ for all $l = 1, \ldots, L$ and $j = 1, \ldots, M$. By the substitution $\mathbf{z} := \mathbf{s}^{-1} \odot \mathbf{x}$ we obtain $\mathbf{x} = \mathbf{s} \odot \mathbf{z}$ and therefore $\mathcal{K}^\mathbf{s}(\mathbf{z}) := \mathcal{K}(\mathbf{s} \odot \mathbf{z}) = \mathcal{K}(\mathbf{x})$.

Since $\mathcal{K}^\mathbf{s}$ is not radial symmetric anymore, we define the regularized kernel function $\mathcal{K}^\mathbf{s}_\mathrm{R}$ in a slightly different way

$$\mathcal{K}^\mathbf{s}_\mathrm{R}(\mathbf{z}) := \begin{cases} T_\mathrm{I}(\|\mathbf{s} \odot \mathbf{z}\|_2) & \text{if} \quad \|\mathbf{s} \odot \mathbf{z}\|_2 \leq \varepsilon_\mathrm{I}, \\ T^\mathbf{z}_\mathrm{B}(\|\mathbf{z}\|_2) & \text{if} \quad \frac{1}{2} - \varepsilon_\mathrm{B} < \|\mathbf{z}\| < \frac{1}{2}, \\ T^\mathbf{z}_\mathrm{B}(\frac{1}{2}) & \text{if} \quad \frac{1}{2} \leq \|\mathbf{z}\|, \\ K(\|\mathbf{s} \odot \mathbf{z}\|_2) & \text{otherwise.} \end{cases}$$

While the definition of T_I remains the same as in the non-scaled case, we again exploit (14) to obtain $T^\mathbf{z}_\mathrm{B}$ with the altered parameters $a_j = K^{(j)}((\frac{1}{2} - \varepsilon_\mathrm{B})\frac{\|\mathbf{s}\odot\mathbf{z}\|_2}{\|\mathbf{z}\|_2}), b_j = \delta_{0,j} K(\frac{1}{2} s_{\max}), j = 0, \ldots, p-1, m = \frac{1}{2} - \frac{\varepsilon_\mathrm{B}}{2}, r = \varepsilon_\mathrm{B}$, where $s_{\max} := \max\{s_t \colon t = 0, \ldots, d-1\}$. Note that the interpolation polynomial $T^\mathbf{z}_\mathrm{B}$ may change for every node, since the coefficients a_j now depend on $\mathbf{z}$. We only need to evaluate $T^\mathbf{z}_\mathrm{B}$ during a precomputation step to obtain the Fourier coefficients $\hat{b}_\mathbf{k}$ of the regularized kernel function $\mathcal{K}_\mathrm{RF}$, therefore the dependency of the interpolation polynomial on $\mathbf{z}$ has no impact on the complexity of our fastsum algorithm. In order to assure differentiability of $\mathcal{K}_\mathrm{RF}$ at the border $\{\mathbf{x} \in \mathbb{T}^d : \|\mathbf{x}\|_2 \geq \frac{1}{2}\}$, we set $T^\mathbf{z}_\mathrm{B}(\frac{1}{2}) = b_0$ constant for all $\mathbf{z}$.

Let $s_\pi := s_0 \cdot \ldots \cdot s_{d-1}$. We approximate the smooth function $\mathcal{K}^\mathbf{s}_\mathrm{R}$ by the Fourier series

$$\mathcal{K}^\mathbf{s}_\mathrm{RF}(\mathbf{x}) := \sum_{\mathbf{k} \in I_{\mathbf{s}\odot\mathbf{N}}} \hat{b}_\mathbf{k} \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k}\mathbf{x}},$$

where the Fourier coefficients $\hat{b}_\mathbf{k}$ are given by the d-variate discrete Fourier transform

$$\hat{b}_\mathbf{k} = \frac{1}{s_\pi N_\pi} \sum_{\mathbf{l} \in I_{\mathbf{s}\odot\mathbf{N}}} \mathcal{K}^\mathbf{s}_\mathrm{R}(\mathbf{s}^{-1} \odot \mathbf{N}^{-1} \odot \mathbf{l}) \mathrm{e}^{+2\pi \mathrm{i} (\mathbf{s}^{-1} \odot \mathbf{N}^{-1} \odot \mathbf{l}) \mathbf{k}}, \quad \mathbf{k} \in I_{\mathbf{s}\odot\mathbf{N}}.$$

The remaining part of the fast summation algorithm can be done analogously to the non-scaled case with the scaled NFFT size $\mathbf{s} \odot \mathbf{N}$ instead of $\mathbf{N}$.

4 Application to Particle Simulation for the Coulomb Potential

In this part we apply our fast summation algorithm to the long-range potential $1/r$. The fundamental idea is to split the long-range part of the potential into a smooth long-range part and a singular short range part. We will use our splitting (19) and (20). There exists a variety of methods which use similar splittings such as the P^3M method[17]. In the following we will discuss open and periodic systems.

4.1 Open systems

In this part we apply our fast summation algorithm to a system of M charged particles located at source nodes $\mathbf{x}_l \in \mathbb{R}^3$ with charge $q_l \in \mathbb{R}$. We are interested in the evaluation of the electrostatic potential ϕ at target node $\mathbf{y} \in \mathbb{R}^3$,

$$\phi(\mathbf{y}) := \sum_{l=1}^{M} q_l \mathcal{K}(\mathbf{y} - \mathbf{x}_l), \tag{23}$$

and the force $\mathbf{F}$ acting at particle $\mathbf{y} \in \mathbb{R}^3$ with charge $q(\mathbf{y}) \in \mathbb{R}$,

$$\mathbf{F}(\mathbf{y}) := -q(\mathbf{y})\nabla\phi(\mathbf{y}) = -q(\mathbf{y}) \sum_{l=1}^{M} q_l \mathcal{K}'(\mathbf{y} - \mathbf{x}_l), \tag{24}$$

where the kernel functions $\mathcal{K}$ and $\mathcal{K}'$ corresponding to the Coulomb potential are given by

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \|\mathbf{x}\|_2 = 0, \\ 1/\|\mathbf{x}\|_2 & \text{otherwise,} \end{cases} \quad \text{and} \quad \mathcal{K}'(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \|\mathbf{x}\|_2 = 0, \\ \mathbf{x}/\|\mathbf{x}\|_2^3 & \text{otherwise.} \end{cases} \tag{25}$$

For equal sets of source and target nodes the sum (23) turns into

$$\phi(\mathbf{x}_j) = \sum_{\substack{l=1 \\ l \neq j}}^{M} \frac{q_l}{\|\mathbf{x}_j - \mathbf{x}_l\|_2}, \quad j = 1, \ldots, M,$$

and (24) becomes

$$\mathbf{F}(\mathbf{x}_j) = -q_j \sum_{\substack{l=1 \\ l \neq j}}^{M} q_l \frac{\mathbf{x}_j - \mathbf{x}_l}{\|\mathbf{x}_j - \mathbf{x}_l\|_2^3}, \quad j = 1, \ldots, M.$$

Furthermore we are interested in the computation of the total electrostatic potential energy

$$U := \frac{1}{2} \sum_{j=1}^{M} q_j \phi(\mathbf{x}_j),$$

which can be evaluated straightforward after the computation of the potentials $\phi(\mathbf{x}_j), j = 1, \ldots, M$. To get the potentials $\phi(\mathbf{x}_j)$ we apply Algorithm 3.1 on (23) by choosing equal sets of source and target nodes and $\alpha_l = q_l$, $l = 1, \ldots, M$. For the near field corrections of this algorithm we require repeated evaluations of the near field interpolation polynomial T_{I}. Instead of using (14) it is more efficient to calculate the coefficients once explicitly for smoothness $p = 2, \ldots, 12$ and evaluate the polynomial with a Horner scheme. E.g., for smoothness $p = 5$ we get the following explicit representation

$$T_{\mathrm{I}}(\|\mathbf{x}\|_2) = \frac{315}{128\varepsilon_{\mathrm{I}}} + \left(\frac{-105}{32\varepsilon_{\mathrm{I}}^3} + \left(\frac{189}{64\varepsilon_{\mathrm{I}}^5} + \left(\frac{-45}{32\varepsilon_{\mathrm{I}}^7} + \frac{35}{128\varepsilon_{\mathrm{I}}^9} \|\mathbf{x}\|_2^2 \right) \|\mathbf{x}\|_2^2 \right) \|\mathbf{x}\|_2^2 \right) \|\mathbf{x}\|_2^2.$$

For the fast calculation of the forces $\mathbf{F}(\mathbf{y})$ we follow the main steps of Algorithm 3.1. First we observe that Algorithm 3.1 decomposes potential $\phi(\mathbf{y})$ into

$$\phi(\mathbf{y}) \approx \phi_{\mathrm{NE}}(\mathbf{y}) + \phi_{\mathrm{RF}}(\mathbf{y}), \tag{26}$$

where the near field part ϕ_{NE} is given by

$$\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1 \\ \|\mathbf{y}-\mathbf{x}_l\|_2 < \varepsilon_{\mathrm{I}}}}^{M} q_l \left(\mathcal{K}(\mathbf{y} - \mathbf{x}_l) - T_{\mathrm{I}}(\|\mathbf{y} - \mathbf{x}_l\|_2) \right)$$

and the far field part ϕ_{RF} reads as

$$\phi_{\mathrm{RF}}(\mathbf{y}) = \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l} \right) \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{y}}.$$

Combination of (24) and (26) yields

$$\mathbf{F}(\mathbf{y}) = -q(\mathbf{y})\nabla\phi(\mathbf{y}) \approx -q(\mathbf{y})(\nabla\phi_{\mathrm{NE}}(\mathbf{y}) + \nabla\phi_{\mathrm{RF}}(\mathbf{y})).$$

Taking into account that

$$\nabla\phi_{\mathrm{RF}}(\mathbf{y}) = -2\pi\mathrm{i}\sum_{\mathbf{k}\in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}}\mathbf{k}\left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi\mathrm{i}\mathbf{k}\mathbf{x}_l}\right)\mathrm{e}^{-2\pi\mathrm{i}\mathbf{k}\mathbf{y}},$$

we are able to compute the inner sum of the far field part $\nabla\phi_{\mathrm{RF}}$ with only one NFFT$^{\mathsf{H}}$ and the outer vector sum with three NFFTs, one for each component. This is a remarkable improvement to the algorithm proposed in[25] where four NFFT$^{\mathsf{H}}$s and four NFFTs are needed to calculate the far field part of the forces. The gradient of the near field part reads as

$$\nabla\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1\\ \|\mathbf{y}-\mathbf{x}_l\|_2<\varepsilon_{\mathrm{I}}}}^{M} q_l\left(\mathcal{K}'(\mathbf{y}-\mathbf{x}_l) - \nabla T_{\mathrm{I}}(\|\mathbf{y}-\mathbf{x}_l\|_2)\right).$$

This vector sum can be evaluated straightforward. One way to obtain the gradient of T_{I} is to use (15) and the chain rule with $z = \|\mathbf{x}\|_2$. We obtain for $\|\mathbf{x}\| \neq 0$

$$\begin{aligned}\nabla T_{\mathrm{I}}(\|\mathbf{x}\|_2) =& \frac{1}{2^p}\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\sum_{j=0}^{p-1}\sum_{\substack{k=0\\ j+k\neq 0}}^{p-1-j}\binom{p-1+k}{k}\frac{r^{j-1}}{j!2^k}\\ &\Big((1+y)^{j+k-1}(1-y)^{p-1}\big[(j+k)(1-y)-p(1+y)\big]a_j\\ &+(1-y)^{j+k-1}(1+y)^{p-1}\big[(j+k)(1+y)-p(1-y)\big](-1)^{j-1}b_j\Big),\end{aligned}$$

where $y = \frac{\|\mathbf{x}\|_2 - m}{r}$ and $\nabla T_{\mathrm{I}}(0) = \mathbf{0}$. Alternatively we represent the gradient of T_{I}, e.g., for smoothness $p = 5$, by

$$\nabla T_{\mathrm{I}}(\|\mathbf{x}\|_2) = \left(\frac{-105}{16\varepsilon_{\mathrm{I}}^3} + \left(\frac{189}{16\varepsilon_{\mathrm{I}}^5} + \left(\frac{-1080}{128\varepsilon_{\mathrm{I}}^7} + \frac{35}{16\varepsilon_{\mathrm{I}}^9}\|\mathbf{x}\|_2^2\right)\|\mathbf{x}\|_2^2\right)\|\mathbf{x}\|_2^2\right)\|\mathbf{x}\|_2^2\right)\mathbf{x}.$$

In summary we can apply Algorithm 3.1 with the matrix representation given in (22).

4.2 Periodic systems

In this section we present a straightforward method, that accelerates the traditional Ewald summation technique by NFFT. This combination was first presented in[16] and is very similar to the FFT-accelerated Ewald sum, namely, the so-called particle-particle particle-mesh (P^{3}M), particle-mesh Ewald (PME) and smooth particle-mesh Ewald (SPME), see[4]. Additionally we will see, that the accelerated Ewald summation can be reinterpreted into a method very similar to our fastsum Algorithm 3.1.

We consider a system of charged particles coupled via the Coulomb potential, a cubic simulation box with edge length B, containing M charged particles, each with a charge $q_l \in \mathbb{R}$, located at $\mathbf{x}_l \in B\mathbb{T}^3$. Periodic boundary conditions in a system without cut-off is

represented by replicating the simulation box in all directions of space. The electrostatic potential ϕ at $\mathbf{y} \in B\mathbb{T}^3$, can be written as a lattice sum, see [12, Chapter 12] and[31],

$$\phi(\mathbf{y}) := \sum_{l=1}^{M} q_l \tilde{\mathcal{K}}(\mathbf{y} - \mathbf{x}_l), \tag{27}$$

and the force $\mathbf{F}$ at particle $\mathbf{y} \in B\mathbb{T}^3$ with charge $q(\mathbf{y}) \in \mathbb{R}$ is given by

$$\mathbf{F}(\mathbf{y}) := -q(\mathbf{y})\nabla\phi(\mathbf{y}) = -q(\mathbf{y}) \sum_{l=1}^{M} q_l \tilde{\mathcal{K}}'(\mathbf{y} - \mathbf{x}_l), \tag{28}$$

where the kernel functions

$$\tilde{\mathcal{K}}(\mathbf{x}) = \sum_{\mathbf{r}\in\mathbb{Z}^3} \mathcal{K}(\mathbf{x} + \mathbf{r}B) \quad \text{and} \quad \tilde{\mathcal{K}}'(\mathbf{x}) = \sum_{\mathbf{r}\in\mathbb{Z}^3} \mathcal{K}'(\mathbf{x} + \mathbf{r}B)$$

are periodizations of the kernel functions (25) corresponding to the Coulomb potential. For equal sets of source and target nodes the evaluation of the potential (27) and the force (28) reads as

$$\phi(\mathbf{x}_j) = \sum_{\mathbf{r}\in\mathbb{Z}^3} \sum_{\substack{l=1 \\ l\neq j \text{ for } \mathbf{r}=\mathbf{0}}}^{M} \frac{q_l}{\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2}, \quad j = 1, \ldots, M \tag{29}$$

and

$$\mathbf{F}(\mathbf{x}_j) = -q_j \sum_{\mathbf{r}\in\mathbb{Z}^3} \sum_{\substack{l=1 \\ l\neq j \text{ for } \mathbf{r}=\mathbf{0}}}^{M} q_l \frac{\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B}{\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2^3}, \quad j = 1, \ldots, M,$$

respectively. Furthermore we are interested in the computation of the total electrostatic potential energy

$$U := \frac{1}{2} \sum_{j=1}^{M} q_j \phi(\mathbf{x}_j),$$

which can be evaluated straightforward after the computation of the potentials $\phi(\mathbf{x}_j)$, $j = 1, \ldots, M$ like in the non-periodic case. The well-known Ewald formula for the computation of (29) splits the electrostatic potential ϕ into three parts

$$\phi = \phi^{\text{real}} + \phi^{\text{reci}} + \phi^{\text{self}}, \tag{30}$$

where the contribution from real space ϕ^{real}, reciprocal space ϕ^{reci} and the self-energy ϕ^{self} are given by

$$\begin{aligned}
\phi^{\text{real}}(\mathbf{x}_j) &= \sum_{\mathbf{r}\in\mathbb{Z}^3} \sum_{\substack{l=1 \\ l\neq j \text{ for } \mathbf{r}=\mathbf{0}}}^{M} q_l \frac{\operatorname{erfc}(\alpha\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2)}{\|\mathbf{x}_j - \mathbf{x}_l + \mathbf{r}B\|_2}, \\
\phi^{\text{reci}}(\mathbf{x}_j) &= \frac{1}{\pi B^3} \sum_{\mathbf{k}\in\mathbb{Z}^3\setminus\{\mathbf{0}\}} \frac{\mathrm{e}^{-\pi^2\|\mathbf{k}\|_2^2/(\alpha B)^2}}{\|\mathbf{k}\|_2^2} \sum_{l=1}^{M} q_l \mathrm{e}^{-2\pi\mathrm{i}\mathbf{k}(\mathbf{x}_j - \mathbf{x}_l)}, \\
\phi^{\text{self}}(\mathbf{x}_j) &= -2q_j \frac{\alpha}{\sqrt{\pi}}.
\end{aligned} \tag{31}$$

Thereby, the complementary error function is defined by $\operatorname{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty \mathrm{e}^{-t^2} \mathrm{d}t$. Choosing optimal parameters, Ewald summation scales as $\mathcal{O}(M^{3/2})$. In order to overcome this increase in time we apply the NFFT for the calculation of the reciprocal-space potential ϕ^{reci} and we obtain a method similar as our fast summation method. To this end, we compute the Fourier transformed charge density

$$S(\mathbf{k}) = \sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l}$$

by NFFT$^{\mathsf{H}}$ and after truncation of the sum (31) we obtain by NFFT

$$\phi^{\mathrm{reci}}(\mathbf{x}_j) \approx \frac{1}{\pi B^3} \sum_{\mathbf{k} \in I_{\mathbf{N}} \setminus \{\mathbf{0}\}} \frac{\mathrm{e}^{-\pi^2 \|\mathbf{k}\|_2^2/(\alpha B)^2}}{\|\mathbf{k}\|_2^2} S(\mathbf{k}) \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{x}_j} .$$

We now reinterpret the approximation steps of Ewald summation in order to use the steps of our Algorithm 3.1. Following (12) we define the slightly different regularization $\mathcal{K}_{\mathrm{R}}$ of the kernel function $\mathcal{K}$ by

$$\mathcal{K}_{\mathrm{R}}(\mathbf{x}) := T_{\mathrm{I}}(\|\mathbf{x}\|_2) \approx \begin{cases} T_{\mathrm{I}}(\|\mathbf{x}\|_2) \text{ if } \quad \|\mathbf{x}\|_2 \leq \varepsilon_{\mathrm{I}}, \\ K(\|\mathbf{x}\|_2) \text{ otherwise,} \end{cases}$$

where the near field regularization T_{I} is given by

$$T_{\mathrm{I}}(z) = \begin{cases} 2\alpha/\sqrt{\pi} \quad \text{if} \quad z = 0, \\ \operatorname{erf}(\alpha z)/z \text{ otherwise.} \end{cases}$$

Here the error function is defined by $\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z \mathrm{e}^{-t^2} \mathrm{d}t$ and the parameter α must be chosen large enough to ensure $T_{\mathrm{I}}(\mathbf{x}) \approx \mathcal{K}(\mathbf{x})$ for all $\|\mathbf{x}\|_2 \geq \varepsilon_{\mathrm{I}}$. Since the periodic regularization $\tilde{\mathcal{K}}_{\mathrm{R}}(\mathbf{x}) := \sum_{\mathbf{r} \in \mathbb{Z}^3} \mathcal{K}_{\mathrm{R}}(\mathbf{x} + \mathbf{r})$ is smooth at the domain border, we do not need a regularization T_{B}. Instead of (30) we use (18) to split the electrostatic potential ϕ from (27) into the two parts

$$\phi(\mathbf{y}) \approx \phi_{\mathrm{NE}}(\mathbf{y}) + \phi_{\mathrm{RF}}(\mathbf{y}),$$

where the near field part ϕ_{NE} is given by

$$\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1 \\ \|\mathbf{y} - \mathbf{x}_l\|_2 < \varepsilon_{\mathrm{I}}}}^{M} q_l \left(\tilde{\mathcal{K}}(\mathbf{y} - \mathbf{x}_l) - T_{\mathrm{I}}(\|\mathbf{y} - \mathbf{x}_l\|_2) \right)$$

and the far field part ϕ_{RF} reads as

$$\phi_{\mathrm{RF}}(\mathbf{y}) = \sum_{\mathbf{k} \in I_{\mathbf{N}}} \hat{b}_{\mathbf{k}} \left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi \mathrm{i} \mathbf{k} \mathbf{x}_l} \right) \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k} \mathbf{y}}.$$

Since the Fourier coefficients $\hat{b}_{\mathbf{k}}$ of $\tilde{\mathcal{K}}_{\mathrm{R}}$ can be calculated analytically,

$$\hat{b}_{\mathbf{k}} = \begin{cases} 0 & \text{if} \quad \|\mathbf{k}\|_2 = 0, \\ \frac{\mathrm{e}^{-\pi^2 \|\mathbf{k}\|_2^2/(\alpha B)^2}}{\|\mathbf{k}\|_2^2} & \text{otherwise,} \end{cases}$$

we do not need to compute an inverse discrete Fourier transformation of $\mathcal{K}_{\mathrm{R}}$ in the pre-computation step as in (17). Analogously to Algorithm 3.1 the far field part ϕ_{RF} can be evaluated by one NFFT$^{\intercal}$ and one NFFT, while the near field part only includes nearest neighbors of every target node. Therefore we get an algorithm with the same complexity as our fastsum algorithm for non-periodic systems. For equal sets of source and target nodes the resulting algorithm coincides with the NFFT-accelerated Ewald summation.

As in the non-periodic case we get

$$\mathbf{F}(\mathbf{y}) = -q(\mathbf{y})\nabla\phi(\mathbf{y}) \approx -q(\mathbf{y})(\nabla\phi_{\mathrm{NE}}(\mathbf{y}) + \nabla\phi_{\mathrm{RF}}(\mathbf{y})),$$

where the far field part $\nabla\phi_{\mathrm{RF}}(\mathbf{y})$ can be computed by one NFFT$^{\intercal}$ and three NFFTs. The near field part $\nabla\phi_{\mathrm{NE}}(\mathbf{y})$ reads as

$$\nabla\phi_{\mathrm{NE}}(\mathbf{y}) = \sum_{\substack{l=1 \\ \|\mathbf{y}-\mathbf{x}_l\|_2<\varepsilon_{\mathrm{I}}}}^{M} q_l \left(\mathcal{K}'(\mathbf{y}-\mathbf{x}_l) - \nabla T_{\mathrm{I}}(\|\mathbf{y}-\mathbf{x}_l\|_2)\right).$$

This vector sum can be evaluated straightforward. Since

$$\frac{\mathrm{d}}{\mathrm{d}z}\frac{\mathrm{erf}(\alpha z)}{z} = \frac{1}{z}\left(\frac{2\alpha}{\sqrt{\pi}}\mathrm{e}^{-\alpha^2 z^2} - \frac{\mathrm{erf}(\alpha z)}{z}\right)$$

the gradient of T_{I} is given by

$$\nabla T_{\mathrm{I}}(\|\mathbf{x}\|_2) = \begin{cases} 0 & \text{if} \quad \|\mathbf{x}\|_2 = 0, \\ \frac{\mathbf{x}}{\|\mathbf{x}\|_2^2}\left(\frac{2\alpha}{\sqrt{\pi}}\mathrm{e}^{-\alpha^2\|\mathbf{x}\|_2^2} - \frac{\mathrm{erf}(\alpha\|\mathbf{x}\|_2)}{\|\mathbf{x}\|_2}\right) & \text{otherwise.} \end{cases}$$

For equal sets of source and target nodes we get

$$\mathbf{F}(\mathbf{x}_j) \approx \mathbf{F}_{\mathrm{NE}}(\mathbf{x}_j) + \mathbf{F}_{\mathrm{RF}}(\mathbf{x}_j),$$

where

$$\begin{aligned}\mathbf{F}_{\mathrm{NE}}(\mathbf{x}_j) =& -q_j\nabla\phi_{\mathrm{NE}}(\mathbf{x}_j) \\ =& -q_j\sum_{\substack{l=1 \\ l\neq j}}^{M} q_l \frac{\mathbf{x}_j-\mathbf{x}_l}{\|\mathbf{x}_j-\mathbf{x}_l\|_2^2}\left(\frac{\mathrm{erfc}(\alpha\|\mathbf{x}_j-\mathbf{x}_l\|_2)}{\|\mathbf{x}_j-\mathbf{x}_l\|_2} + \frac{2\alpha}{\sqrt{\pi}}\mathrm{e}^{-\alpha^2\|\mathbf{x}_j-\mathbf{x}_l\|_2^2}\right),\end{aligned}$$

$$\begin{aligned}\mathbf{F}_{\mathrm{RF}}(\mathbf{x}_j) =& -q_j\nabla\phi_{\mathrm{RF}}(\mathbf{x}_j) \\ =& \frac{2\mathrm{i}q_j}{B^3}\sum_{\mathbf{k}\in I_{\mathbf{N}}\setminus\{\mathbf{0}\}} \frac{\mathrm{e}^{-\pi^2\|\mathbf{k}\|_2^2/(\alpha B)^2}}{\|\mathbf{k}\|_2^2}\mathbf{k}S(\mathbf{k})\mathrm{e}^{-2\pi\mathrm{i}\mathbf{k}\mathbf{x}_j}.\end{aligned}$$

This splitting coincides with the well known results of Ewald summation.

In summary we can apply Algorithm 3.1 with the matrix representation given in (22).

5 Numerical Results

5.1 Generation of pseudo random sampling sets

To guarantee a minimum distance between all nodes we used Hammersley and Halton pseudo random node distributions[33]. Let $p \in \mathbb{N}$ prime. Every non-negative integer $j \in$

$\mathbb{N} \cup \{0\}$ can be uniquely represented as

$$j = a_0 + a_1 p + a_2 p^2 + \ldots + a_r p^r, \quad a_i \in \{0, \ldots, p-1\}, \; i = 0, \ldots, r, \quad r \in \mathbb{N}.$$

We define

$$\Phi_p(j) := \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \ldots + \frac{a_r}{p^{r+1}}.$$

For d given primes $p_1 < p_2 < \ldots < p_d$ the d-variate Hammersley distribution is given by the following M nodes

$$\left(\frac{j}{M}, \Phi_{p_1}(j), \Phi_{p_2}(j), \ldots, \Phi_{p_{d-1}}(j) \right)^\top, \quad j = 0, \ldots, M-1$$

and the d-variate Halton distribution consists of the M nodes

$$\left(\Phi_{p_1}(j), \Phi_{p_2}(j), \ldots, \Phi_{p_d}(j) \right)^\top, \quad j = 0, \ldots, M-1.$$

Note that it is possible to increase a given set of Halton distributed nodes, while the number of Hammersley distributed nodes must be fixed because of the dependency of the first component of every node on M. Both distributions were implemented in[2]. The software package called *HAMMERSLEY* contains algorithms to generate Hammersley and Halton distributions on the square $[0,1]^2$ and the cube $[0,1]^3$. Furthermore it includes routines for mapping the square $[0,1]^2$ to the sphere $\{\mathbf{x} \in \mathbb{R}^3 \colon \|\mathbf{x}\|_2 = 1\}$ and mapping the cube $[0,1]^3$ to the ball $\{\mathbf{x} \in \mathbb{R}^3 \colon \|\mathbf{x}\|_2 \leq 1\}$.

5.2 Error definitions

We performed the computations of the total energy U, the potentials $\phi(\mathbf{x}_j)$ and the forces $\mathbf{F}(\mathbf{x}_j)$ for different node distributions with the direct algorithm, the fast multipole method from FCS software library[1] and the NFFT based algorithms. In this section we use the names of the applied methods to label numerical results, e.g., $U^{\mathrm{direct}}, U^{\mathrm{FMM}}$ and U^{NFFT} holds the results of the energy computations based on the three algorithms, respectively. We used the following error measurements to compare the results produced by the three methods.

The relative errors of the total potential energy with respect to the applied method read as

$$E_U^{\mathrm{FMM}} := \frac{U^{\mathrm{FMM}}}{U^{\mathrm{direct}}}, \quad E_U^{\mathrm{NFFT}} := \frac{U^{\mathrm{NFFT}}}{U^{\mathrm{direct}}}.$$

Let $\phi = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_M))^\top$. The relative errors of the potential with respect to the applied method are given by

$$E_\phi^{\mathrm{FMM}} := \frac{\|\phi^{\mathrm{direct}} - \phi^{\mathrm{FMM}}\|_2}{\|\phi^{\mathrm{direct}}\|_2}, \quad E_\phi^{\mathrm{NFFT}} := \frac{\|\phi^{\mathrm{direct}} - \phi^{\mathrm{NFFT}}\|_2}{\|\phi^{\mathrm{direct}}\|_2}.$$

For $\mathbf{F}(\mathbf{x}) = (F_0(\mathbf{x}), F_1(\mathbf{x}), F_2(\mathbf{x}))^\top$, and $\mathbf{F}_t := (F_t(\mathbf{x}_0), \ldots, F_t(\mathbf{x}_M))^\top$, $t = 0, 1, 2$, we define the average relative errors of the forces with respect to the applied method via

$$E_{\mathbf{F}}^{\mathrm{FMM}} := \frac{1}{3} \sum_{t=0}^{2} \frac{\|\mathbf{F}_t^{\mathrm{direct}} - \mathbf{F}_t^{\mathrm{FMM}}\|_1}{\|\mathbf{F}_t^{\mathrm{direct}}\|_1}, \quad E_{\mathbf{F}}^{\mathrm{NFFT}} := \frac{1}{3} \sum_{t=0}^{2} \frac{\|\mathbf{F}_t^{\mathrm{direct}} - \mathbf{F}_t^{\mathrm{NFFT}}\|_1}{\|\mathbf{F}_t^{\mathrm{direct}}\|_1}.$$

5.3 Test cases

We computed all test cases on the nowadays retired JUMP cluster at Research Center Jülich. Each of its 41 nodes got 32 Power4+ processors at 1.7 GHz and 128 GB memory. Our test runs were performed on one processor of one completely allocated node. The FCS software library[1] (timestamp 16.08.2007) and the NFFT library were compiled with IBM's *xlf* and *xlc* compilers at optimization level *-O5* and with the flag *-q64* to support the 64 bit architecture of JUMP.

We performed the NFFT based fastsum algorithm with equal oversampling factors $\sigma_t = 2, t = 0, 1, 2$, the truncation parameter $m = 2$, choose the Kaiser-Bessel window function, the regularization parameter $p = 5$ and set the initialization flags *PRE_PHI_HUT*, *PRE_LIN_PSI*, *FFT_OUT_OF_PLACE*, *FFTW_MEASURE* and *FFTW_DESTROY_INPUT* to obtain the following results.

The upper bound of the relative error of the total energy U^{FMM} was set to 0.001 for the FMM based calculations. For the NFFT based computations we tried to adjust the parameters to obtain the same upper bound on U^{NFFT}. All measured times include the computation of the energy, the potentials and the forces.

Hammersley distribution within a cube

For this test case M nodes satisfy a Hammersley distribution with parameters $p_1 = 2, p_2 = 3$ on $[0, 1]^3$. The randomly chosen charges $q_l \in \{-1, 1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1, 0, 1\}.$$

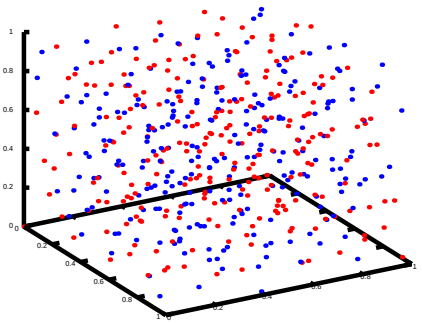


Figure 2: Hammersley distribution within a cube for 500 nodes.

Hammersley distribution within a ball

For this test case a Hammersley distribution with parameters $p_1 = 2, p_2 = 3$ on $[0, 1]^3$ is mapped to the ball

$$(x - 0.5)^2 + (y - 0.5)^2 + (z - 0.5)^2 \leq (0.5)^2.$$

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3/32	0.01	0.01	0.11
5 000	$(32,32,32)^\top$	3/32	0.86	0.25	0.38
50 000	$(64,64,64)^\top$	3/64	87.09*	3.00	4.63

Table 1: Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution in a cube. Times with $*$ are estimated.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	1.544e-15	3.094e-04	1.603e-15	8.086e-04	1.689e-15	1.917e-03
5 000	1.668e-07	9.205e-05	2.571e-06	5.626e-04	7.270e-06	9.513e-04
50 000	2.388e-08	3.006e-04	3.222e-07	5.454e-04	8.060e-07	1.240e-03

Table 2: Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution in a cube.

The randomly chosen charges $q_l \in \{-1, 1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1, 0, 1\}\,.$$

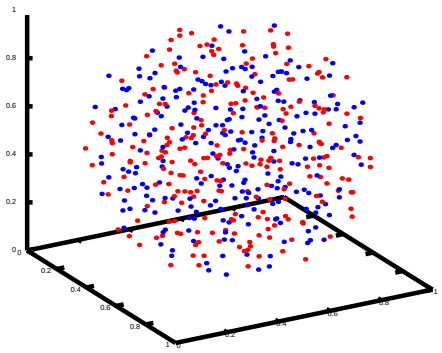


Figure 3: Hammersley distribution within a ball for 500 nodes.

Cylindrical Halton distribution

For this test case a Halton distribution with parameters $p_1 = 2, p_2 = 3, p_3 = 5$ on the cylinder

$$0 \le x_0 \le 10, \quad (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \le 1$$

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3.5/32	0.01	0.01	0.11
5 000	$(32,32,32)^\top$	3.5/32	0.86	0.24	0.34
50 000	$(64,64,64)^\top$	3.5/64	87.06	3.47	4.20
500 000	$(128,128,128)^\top$	3.5/128	8903.23	46.77	59.48

Table 3: Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution in a ball.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	9.677e-16	5.584e-05	1.751e-15	4.189e-04	2.328e-15	1.092e-03
5 000	1.339e-06	4.903e-05	5.471e-06	3.190e-04	1.454e-05	6.404e-04
50 000	1.113e-07	4.087e-04	2.579e-07	2.507e-04	7.650e-07	7.416e-04
500 000	1.075e-09	3.726e-04	7.366e-08	2.962e-04	2.195e-07	7.613e-04

Table 4: Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution in a ball.

is scaled into the cube $[0,1]^3$. The randomly chosen charges $q_l \in \{-1,1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1,0,1\}\,.$$

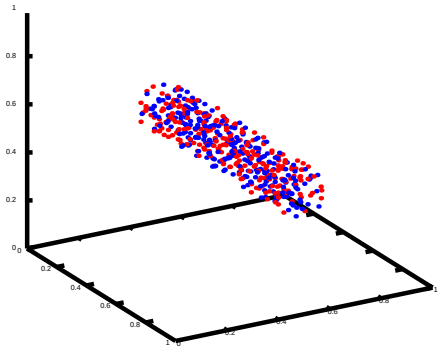


Figure 4: Cylindrical Halton distribution for 500 nodes.

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3/32	0.01	0.04	0.12
5 000	$(32,32,32)^\top$	3/32	0.86	0.25	0.60
50 000	$(256,64,64)^\top$	3.5/64	87.19	3.80	9.94

Table 5: Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Halton distribution in a cylinder.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	1.401e-06	1.201e-04	1.295e-05	4.374e-04	1.535e-05	3.973e-04
5 000	2.938e-07	2.161e-04	2.030e-06	3.914e-04	4.497e-06	2.491e-04
50 000	6.025e-08	1.065e-04	2.663e-08	3.011e-04	7.379e-08	1.302e-03

Table 6: Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Halton distribution in a cylinder.

Spherical Hammersley distribution

For this test case a two-dimensional Hammersley distribution on the square $[0,1]^2$ with parameter $p_1 = 2$ is mapped to the sphere

$$(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2 = (0.5)^2$$

The randomly chosen charges $q_l \in \{-1,1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1,0,1\}\,.$$

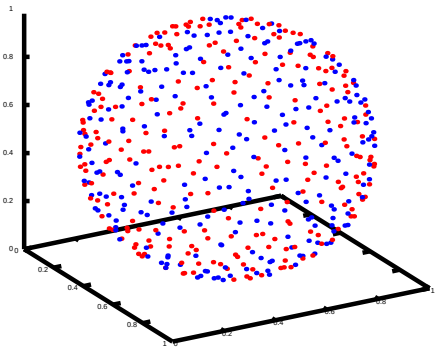


Figure 5: Spherical Hammersley distribution for 500 nodes.

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	3/32	0.01	0.01	0.11
5 000	$(32,32,32)^\top$	3/32	0.86	0.17	0.23
50 000	$(64,64,64)^\top$	3/64	87.13	2.58	3.41

Table 7: Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution on a sphere.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	0.000e+00	1.338e-04	1.740e-15	6.655e-04	2.983e-15	1.697e-03
5 000	9.167e-08	1.957e-04	1.202e-06	4.101e-04	1.478e-06	3.644e-04
50 000	2.460e-08	4.866e-04	5.522e-08	2.422e-04	5.200e-08	1.545e-04

Table 8: Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution on a sphere.

Two Hammersley distributed balls

For this test case $\frac{63}{64}M$ Hammersley distributed nodes on the cube $[0,1]^3$ with parameters $p_1 = 2, p_2 = 3$ are mapped to the ball $(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2 \leq (0.5)^2$. A second set of $\frac{1}{64}M$ Hammersley distributed nodes on the cube $[0,1]^3$ with parameters $p_1 = 2, p_2 = 3$ is mapped to a smaller ball such that the density of nodes is equal to the first. We set the distance between the balls to 10. Finally the whole set of nodes is scaled into the cube $[0,1]^3$. The random charges $q_l \in \{-1, 1\}$ fulfill

$$\sum_{l=1}^{M} q_l \in \{-1, 0, 1\}\,.$$

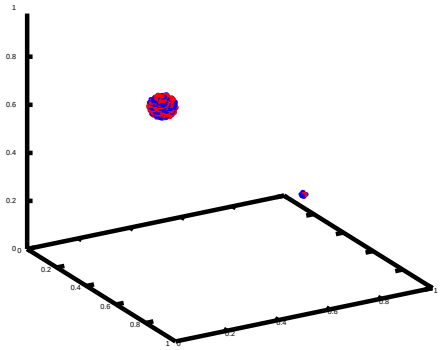


Figure 6: Hammersley distribution `hammersley_two_balls` with 500 nodes.

$\mathbf{M}$	$\mathbf{N}$	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
500	$(32,32,32)^\top$	2/32	0.01	0.01	0.12
5 000	$(32,32,32)^\top$	2/32	0.86	0.26	1.15
50 000	$(384,48,48)^\top$	3.2/48	87.21	22.44	17.01

Table 9: Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes satisfy a Hammersley distribution in two balls.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
500	6.111e-15	2.410e-04	9.988e-15	4.439e-04	1.794e-14	2.286e-04
5 000	1.188e-07	5.444e-04	8.945e-07	6.288e-04	8.189e-08	2.322e-04
50 000	5.916e-08	7.673e-04	1.115e-07	6.624e-04	7.228e-09	1.033e-03

Table 10: Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes satisfy a Hammersley distribution in two balls.

NaCl grid structure

Let $M \in \mathbb{N} \setminus \{1\}$ be a cubic number of nodes. For $u, v, w \in \{0, \dots, \sqrt[3]{M} - 1\}$ we set

$$l = \left(\sqrt[3]{M}u + v\right)\sqrt[3]{M} + w + 1$$

and the nodes $\mathbf{x}_l$ and charges q_l are given by

$$\mathbf{x}_l = \frac{1}{\sqrt[3]{M} - 1}(u, v, w)^\top, \quad q_l = (-1)^{u+v+w+1}.$$

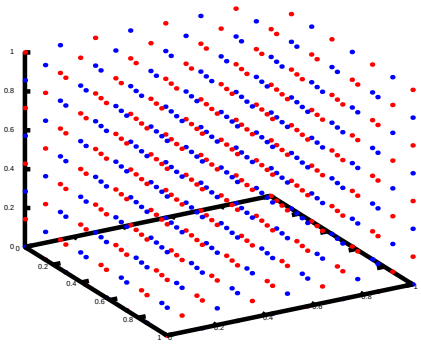


Figure 7: NaCl grid structure for 512 nodes.

M	N	$\varepsilon_I = \varepsilon_B$	t^{direct}	t^{FMM}	t^{NFFT}
512	$(32,32,32)^\top$	4/32	0.01	0.01	0.12
5 832	$(32,32,32)^\top$	2.5/32	1.17	0.13	0.33
50 653	$(64,64,64)^\top$	2.5/64	106.73*	2.41	3.21

Table 11: Runtimes t in seconds of direct, FMM and NFFT based computations, respectively. The nodes are located like a NaCl grid. Times with $*$ are estimated.

M	E_U^{FMM}	E_U^{NFFT}	E_ϕ^{FMM}	E_ϕ^{NFFT}	$E_{\mathbf{F}}^{\text{FMM}}$	$E_{\mathbf{F}}^{\text{NFFT}}$
512	1.648e-15	2.316e-05	1.454e-15	2.892e-05	8.933e-15	2.799e-03
5 832	3.132e-07	4.491e-05	1.406e-05	5.430e-05	5.102e-04	9.325e-04
50 653	3.094e-07	5.422e-05	1.054e-05	7.071e-05	9.372e-04	1.451e-03

Table 12: Errors E_U of total potential energy, E_ϕ of the potential and $E_{\mathbf{F}}$ of the forces for FMM and NFFT, respectively. The nodes are located like a NaCl grid.

5.4 Conclusions

We computed the total energy, the potentials and the forces for several charged particle distributions with open boundary conditions. The runtimes of our NFFT based fast summation showed that it is able to compete with the highly optimized, kernel dependent FMM. We want to emphasize that the NFFT based algorithm is not restricted to the Coulomb potential since it can be easily adapted to various kernel functions. Testcases with equally distributed systems are the cubic Hammersley distribution, the Hammersley distribution within a ball and the NaCl grid structure. Especially for the last one we see in Table 12 that the errors of both algorithms are comparable. As examples for unequally distributed systems we chose the Halton distribution within a cylinder, the Hammersley distribution on a sphere and two Hammersley distributed balls. Although we did not optimize our algorithm for such inhomogeneous systems, the runtimes show an remarkable improvement in comparison to the direct algorithm. The generalization to cuboidal domains of the NFFT based fast summation was successfully tested for the cylindrical Halton distribution and the two Hammersley distributed balls, as one can see in Table 5 and Table 9.

6 Summary

We suggested fast summation algorithms of the form

$$\mathbf{K}\alpha \approx \mathbf{B_y F D D_b D}^\top \mathbf{F}^{\mathrm{H}} \mathbf{B}_\mathbf{x}^\top \alpha + \mathbf{K}_{\mathrm{NE}}\alpha$$

and applied this method to particle simulation for the Coulomb potential. This decomposition into separate building blocks simplifies the implementation. Note that particle-mesh methods are based on similar steps, see [15, Chapter 7]

- smear the charge density onto a grid, i.e., multiplication with $\mathbf{B}_\mathbf{x}^\top$, see step 1 of Algorithm 2.2
- Fourier transform the density, see step 2 of Algorithm 2.2 with FFT
- convolution in Fourier space, or solution of a differential equation in Fourier space
- Fourier transform back, see step 2 of Algorithm 2.1 with FFT

- Back-interpolation, or approximation, i.e., multiplication with $\mathbf{B_y}$, see step 3 of Algorithm 2.1
- Near field computation, i.e., multiplication with $\mathbf{K}_{\mathrm{NE}}$

A parallelization can be done step by step for the modules FFT, NFFT and finally the fast summation algorithm. A promising highly scalable FFT implementation has been tested up to 262144 cores of a BlueGene/P at Research Center Jülich[24].

Acknowledgments

The work was partly supported by the BMBF grant 01IH08001B. We are grateful to Toni Volkmer who produced the numerical results. Therefore we also thank the Jülich Supercomputing Center for providing the computational resources on JUMP. Furthermore, we thank Dr. Holger Dachsel and Dr. Ivo Kabadshow for their advise on the FCS software library. We remark that a new version of FCS is available for download.

References

1. *FCS software library*. `http://www2.fz-juelich.de/jsc/fcs`.
2. *HAMMERSLEY. The Hammersley Quasirandom Sequence.* `http://people.scs.fsu.edu/~burkardt/cpp_src/hammersley/hammersley.html`.
3. G. Beylkin: *On the fast Fourier transform of functions with singularities*. Appl. Comput. Harmon. Anal., 2:363 – 381, 1995.
4. M. Deserno and C. Holm: *How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines*. J. Chem. Phys., 109:7678 – 7693, 1998.
5. A.J.W. Duijndam and M.A. Schonewille: *Nonuniform fast Fourier transform*. Geophysics, 64:539 – 551, 1999.
6. A. Dutt and V. Rokhlin: *Fast Fourier transforms for nonequispaced data*. SIAM J. Sci. Stat. Comput., 14:1368 – 1393, 1993.
7. B. Elbel and G. Steidl: *Fast Fourier transform for nonequispaced data*. In C.K. Chui and L.L. Schumaker (eds.): *Approximation Theory IX*, pp. 39 – 46, Nashville, 1998. Vanderbilt University Press.
8. U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee, and L.G. Pedersen: *A smooth particle mesh Ewald method*. J. Chem. Phys., 103:8577 – 8593, 1995.
9. M. Fenn: *Fast Fourier transform at nonequispaced nodes and applications*. Dissertation, Universität Mannheim, 2006.
10. M. Fenn and G. Steidl: *Fast NFFT based summation of radial functions*. Sampl. Theory Signal Image Process., 3:1 – 28, 2004.
11. J.A. Fessler and B.P. Sutton: *Nonuniform fast Fourier transforms using min-max interpolation*. IEEE Trans. Signal Process., 51:560 – 574, 2003.
12. D. Frenkel and B. Smit: *Understanding molecular simulation: From algorithms to applications*. Academic Press, 2002.
13. M. Frigo and S.G. Johnson: *FFTW, C subroutine library*. `http://www.fftw.org`.
14. L. Greengard and J.Y. Lee: *Accelerating the nonuniform fast Fourier transform*. SIAM Rev., 46:443 – 454, 2004.
15. M. Griebel, S. Knapek, and G. Zumbusch: *Numerical simulation in molecular dynamics*, vol. 5 of *Texts in Computational Science and Engineering*. Springer, Berlin, 2007.

16. F. Hedman and A. Laaksonen: *Ewald summation based on nonuniform fast Fourier transform*. Chem. Phys.Lett., 425:142 – 147, 2006.
17. R.W. Hockney and J.W. Eastwood: *Computer simulation using particles*. Taylor & Francis, Inc., Bristol, PA, USA, 1988.
18. J.I. Jackson, C.H. Meyer, D.G. Nishimura, and A. Macovski: *Selection of a convolution function for Fourier inversion using gridding*. IEEE Trans. Med. Imag., 10:473 – 478, 1991.
19. J. Keiner, S. Kunis, and D. Potts: *NFFT 3.0, C subroutine library*. `http://www.tu-chemnitz.de/~potts/nfft`.
20. J. Keiner, S. Kunis, and D. Potts: *Using NFFT3 - a software library for various nonequispaced fast Fourier transforms*. ACM Trans. Math. Software, 36:Article 19, 1 – 30, 2009.
21. S. Kunis and D. Potts: *Time and memory requirements of the nonequispaced FFT*. Sampl. Theory Signal Image Process., 7:77 – 100, 2008.
22. S. Kunis, D. Potts, and G. Steidl: *Fast Gauss transform with complex parameters using NFFTs*. J. Numer. Math., 14:295 – 303, 2006.
23. S. Matej, J.A. Fessler, and I.G. Kazantsev: *Iterative tomographic image reconstruction using Fourier-based forward and back-projectors*. IEEE Trans. Med. Imag., 23:401 – 412, 2004.
24. M. Pippig: *Scaling parallel fast fourier transform on bluegene/p*. In B. Mohr and W. Frings (eds.): *Jülich BlueGene/P Scaling Workshop 2010*, pp. 27 – 30, Jülich, May 2010. Forschungszentrum Jülich. Technical Report.
25. G. Pöplau, D. Potts, and U. van Rienen: *Calculation of 3D space-charge fields of bunches of charged particles by fast summation*. In A. Anile, G. Alì, and G. Mascaly (eds.): *Scientific Computing in Electrical Engineering*, pp. 241 – 246, Berlin Heidelberg, Germany, 2006. Springer.
26. D. Potts: *Schnelle Fourier-Transformationen für nichtäquidistante Daten und Anwendungen*. Habilitation, Universität zu Lübeck, `http://www.tu-chemnitz.de/~potts`, 2003.
27. D. Potts and G. Steidl: *Fast summation at nonequispaced knots by NFFTs*. SIAM J. Sci. Comput., 24:2013 – 2037, 2003.
28. D. Potts, G. Steidl, and A. Nieslony: *Fast convolution with radial kernels at nonequispaced knots*. Numer. Math., 98:329 – 351, 2004.
29. D. Potts, G. Steidl, and M. Tasche: *Fast Fourier transforms for nonequispaced data: A tutorial*. In J.J. Benedetto and P.J.S.G. Ferreira (eds.): *Modern Sampling Theory: Mathematics and Applications*, pp. 247 – 270, Boston, MA, USA, 2001. Birkhäuser.
30. G. Steidl: *A note on fast Fourier transforms for nonequispaced grids*. Adv. Comput. Math., 9:337 – 353, 1998.
31. G. Sutmann: *Molecular dynamics - vision and reality*. In J. Grotendorst, S. Blügel, and D. Marx (eds.): *Computational Nanoscience: Do It Yourself!*, vol. 31 of *NIC Series*, pp. 159 – 194, Jülich, Feb. 2006. Forschungszentrum Jülich, John von Neumann Institute for Computing, ISBN 3-00-017350-1. Lecture Notes.
32. A.F. Ware: *Fast approximate Fourier transforms for irregularly spaced data*. SIAM Rev., 40:838 – 856, 1998.
33. T.T. Wong, W.S. Luk, and P.A. Heng: *Sampling with Hammersley and Halton Points*. Journal of graphics tools, 2:9 – 24, 1997.

Simulating Charged Systems with ESPResSo

Axel Arnold, Olaf Lenz, and Christian Holm

Institute for Computational Physics, Universität Stuttgart
Pfaffenwaldring 27, 70569 Stuttgart, Germany
olenz@icp.uni-stuttgart.de

We give an introduction to working with the MD simulation package ESPResSo , that allows to perform many-particle simulations of charged systems. The package provides different methods for the computation of long-range interactions, namely P^3M, MMM2D, MMM1D, ELC and Maggs method for Coulomb interactions, as well as the dipolar P^3M algorithm for the interaction between point-like dipoles.

1 Introduction

Nowadays computer simulations are a well established tool in theoretical physics. Here, we want to give an introduction to our simulation package ESPResSo [1–3]; ESPResSo is an acronym for **E**xtensible **S**imulation **P**ackage for **Res**earch on **So**ft matter systems.

The term soft matter, or complex fluids, as they are called in the American literature, describes a large class of materials, such as polymers, colloids, liquid crystals, glasses, hydrogels and dipolar fluids; familiar examples of such materials are glues, paints, soaps or baby diapers. Also most biological materials are soft matter – DNA, membranes, filaments and other proteins belong to this class. The research in soft matter science has been increased in the last decade due to its high potential usefulness in technology, biophysics, and nanoscience.

Many properties of soft matter emerge on the molecular rather than the atomistic level: the elasticity of rubber is the result of entropy of the long polymers molecules, and the hygroscopic materials used in modern diapers store water inside a polyelectrolyte network. To reproduce these physical effects in a computer simulation on the atomistic level, one would have to incorporate many millions of atoms, which is only possible on very small time scales even with the most powerful modern computers. However, often a much simpler description of the material is sufficient. Polymers such as polyelectrolytes or rubber often can be modeled by bead-spring models, *i.e.* (charged) spheres connected by springs, where each of the spheres represents several atoms, often a complete monomer or even larger compounds. Although these models hide most of the chemical properties, they are quite successful in the description of polymers and other soft-matter systems. The process of removing atomistic details from a system to obtain a simpler model is called *coarse-graining*.

Computer simulations of coarse-grained models still incorporate several thousands of spheres and springs and require an efficient simulation software. Furthermore, the models often need non-standard algorithms for the simulation that in some cases have been especially developed for a specific model. Therefore it is necessary that the simulation software is much more flexible than standard atomistic simulation packages (as for example GROMACS or NAMD).

ESPResSo was designed and implemented to address the requirements of such coarse-grained models, and it has a few unique features that distinguish it from any other simulation package that we know of.

2 Features of ESPResSo

Free and open source ESPResSo is an open-source program that is published under the GNU public license, and is available through our web page[1].

Extensible Users can read and modify the code to meet their own needs. Throughout the source code of ESPResSo , readability is preferred over code optimizations. This allows users to extend the code. Furthermore, ESPResSo has defined a number of interfaces that allow users to implement extensions to the ESPResSo core code (*e.g.* new potentials, new thermostats or new analysis routines)

Controlled by Tcl ESPResSo uses the scripting language Tcl[4] to control the simulations. The simulation control script determines all simulation parameters such as the number and type of particles, the type of interactions between these particles, and how the system is propagated; most of the parameters can be changed even during the simulation.

This flexibility makes it possible to perform highly complex simulation procedures, such as adapting the interaction parameters to the current configuration during the simulation, cooling down the system in a simulated annealing process, applying or removing constraints, or even complex schemes like parallel tempering. The flexibility provided by controlling the simulation via Tcl is unmatched by any other simulation packages that we know of.

Parallelized ESPResSo is parallelized code, allowing for simulations of millions of particles on hundreds of CPUs. ESPResSo scales well, it can achieve an efficiency of about 70% on 512 Power4+ processors. Since ESPResSo contains some of the fastest currently available simulation algorithms, it also scales well with the number of particles, allowing for the simulation of large systems.

Portable The code kernel is written in simple ANSI C, therefore it can be compiled on a wide variety of hardware platforms like desktop workstations, convenience clusters and high performance supercomputers based on POSIX operating systems (for example all Unices including GNU/Linux).

Nevertheless, one should be aware that the flexibility of ESPResSo also costs some performance: compared to fast MD programs such as GROMACS, ESPResSo is slower by a factor of about 2. However, most of the problems that we use ESPResSo for cannot be treated with these fast codes at all without massive changes to the simulation engine.

ESPResSo is not self-contained, but relies on other open source packages. Most prominent is the use of the Tcl[4] script language interpreter for the simulation control. For the parallelization, standard MPI[5] routines are used. P^{3}M relies on the FFTW package[6]. Besides these libraries, which are required to be able to have a running version of ESPResSo , the development process is supported heavily by the use of the CVS version

control system[7], which allows several developers to work simultaneously on the code, the documentation generation tool Doxygen[8], LaTeXand the GNU Autotools for compilation.

Of course it is also important to have a look at the actual algorithms and methods that ESPResSo provides.

Integrators and thermostats ESPResSo can perform MD simulations using a Velocity-Verlet integration scheme. Besides the microcanonical (NVE) ensemble, it is possible to obtain the canonical (NVT) ensemble via the Langevin thermostat, one can employ DPD for the NVT ensemble with hydrodynamic interactions, the NPT (constant pressure) ensemble can be obtained using an algorithm by Dünweg *et al*[9]. Furthermore, a quaternion integrator can be used for non-spherical particles (*e.g.* Gay-Berne ellipsoids) or particles that have a dipole moment.

Nonbonded potentials For nonbonded interactions between different particle species, a number of different potentials are implemented in ESPResSo , for example the Lennard-Jones, Gay-Berne, Morse and Buckingham potentials. In addition, it is possible to use tabulated potentials. To avoid overlap problems during equilibration, ESPResSo allows to cap the nonbonded potentials.

Bonded potentials ESPResSo contains a number of interactions between two or more specific particles, including the FENE and harmonic bond potentials, bond angle and dihedral interactions. Again, potentials can also be included as tables.

Long-range interactions ESPResSo has a number of algorithms for long-ranged interactions: for electrostatics, the P^3M , MMM2D, MMM1D, ELC and Maggs' method are implemented, for magnetostatic dipolar interactions, a dipolar P^3M version is contained.

Constraints ESPResSo has the ability to fix some or all coordinates of a particle, or to apply an arbitrary external force on each particle. In addition, spatial constraints such as spherical or cubic compartments, can be used. These constraints interact by any nonbonded interaction with the particles.

Analysis All ESPResSo analysis routines are available in the simulation engine, allowing for both online analysis (during the simulation) as well as offline analysis. ESPResSo can of course calculate the energy and (isotropic) pressure, and the forces acting on particles or spatial constraints can be obtained from the simulation engine. There are routines to determine particle distributions and structure factors, and some polymer-specific measures such as the end-to-end distance or the radius of gyration. For visualization ESPResSo can output files that can be read by visualization software such as VMD[10].

Lattice-Boltzmann fluid ESPResSo contains an implementation of the Lattice-Boltzmann fluid that can be coupled to particle-based models. This allows models to involve hydrodynamic interactions.

AdResS ESPResSo contains an implementation of the AdResS **Ad**aptive **Res**olution **S**cheme that allows to simulate a single simulation that contains two levels of resolution.[11]

3 Using ESPResSo

In this section we want to construct a simulation script for a simple salt melt, *e.g.* NaCl at about 1400 K. The system is to consist out of 100 (+1) ions and 100 (-1) counterions. Besides the electrostatic interaction, the ions interact via the purely repulsive soft core of a Lennard-Jones interaction (the so-called Weeks-Chandler-Anderson or WCA potential). This potential also defines the length scale of the system: the length unit is the Lennard-Jones parameter σ (which roughly corresponds to 0.6nm). We simulate at a density of $0.7\sigma^{-3}$. The Bjerrum length is set to 10σ.

We cannot give a full Tcl tutorial here; however, most of the constructs should be self-explanatory. We also assume that the reader is familiar with the concepts of an MD simulation. The code snippets can be copied into a file, which then can be run using `Espresso <file>` from the ESPResSo source directory.

Our script starts with setting up the initial configuration. Most conveniently, one would like to specify the density and the number of particles of the system as parameters:

```
set n_part 200; set density 0.7
set box_l [expr pow($n_part/$density,1./3.)]
```

These variables do not change anything in the simulation engine, but are just standard Tcl-variables; they are used to increase the readability and flexibility of the script. The box length is not a parameter of this simulation; it is calculated from the number of particles and the system density. This allows to change the parameters later easily, *e.g.* to simulate a bigger system.

The parameters of the simulation engine are modified by the **setmd** command. For example

```
setmd box_l $box_l $box_l $box_l
setmd periodic 1 1 1
```

defines a cubic simulation box of size `box_l`, and periodic boundary conditions in all spatial dimensions. We now fill this simulation box with particles

```
set q 1; set type 0
for {set i 0} { $i < $n_part } {incr i} {
  set posx [expr $box_l*[t_random]]
  set posy [expr $box_l*[t_random]]
  set posz [expr $box_l*[t_random]]
  set q [expr -$q]; set type [expr 1-$type]
  part $i pos $posx $posy $posz q $q type $type
}
```

This loop adds `n_part` particles at random positions, one by one. In this construct, only two commands are not standard Tcl-commands: the random number generator **t_random** and the **part** command, which is used to specify particle properties, here the position, the charge `q` and the type. In ESPResSo the particle type is just an integer number which allows to group particles; it does not imply any physical parameters. Here we use it to tag the charges: positive charges have type 0, negative charges have type 1.

Now we define the ensemble that we will be simulating. We want to do a canonical NVT simulation, so we need to use a thermostat, which can be activated using the **thermostat** command. We also set some integration parameters:

```
setmd time_step 0.01; setmd skin 0.4
set temp 1; set gamma 1
thermostat langevin $temp $gamma
```

This switches on the Langevin thermostat for the NVT ensemble, with temperature `temp` and friction `gamma`. The skin depth `skin` is a parameter for the that tunes ESPResSo 's performance (*i.e.* it will not influence the results), but cannot be discussed here.

Before we can really start the simulation, we have to specify the interactions between our particles. We use a simple, purely repulsive Lennard-Jones interaction to model the hard core repulsion[12], and the charges interact via the Coulomb potential:

```
set sig 1.0; set cut   [expr 1.12246*$sig]
set eps 1.0; set shift [expr 0.25*$eps]
set bjerrum 10.0
inter 0 0 lennard-jones $eps $sig $cut $shift 0
inter 1 0 lennard-jones $eps $sig $cut $shift 0
inter 1 1 lennard-jones $eps $sig $cut $shift 0
inter coulomb $bjerrum p3m tunev2 accuracy 1e-3 mesh 32
```

The first three **inter** commands instruct ESPResSo to use the same purely repulsive Lennard-Jones potential for the interaction between all combinations of the two particle types 0 and 1; by using different parameters for different combinations, one could simulate differently sized particles. The last line sets the Bjerrum length to the value 10, and then instructs ESPResSo to use P^3M for the Coulombic interaction and to try to find suitable parameters for an rms force error below 10^{-4}, with a fixed mesh size of 32. The mesh is fixed here to speed up the tuning; for a real simulation, one will also tune this parameter.

Now we would be ready for integration. However, if we would start integating now, the simulation would crash: ESPResSo would complain about particle coordinates being out of range. The reason for this is simple: Due to the random setup, some of the particles are overlapping strongly, which would generate extremely high forces and catapult the particles out of the system if we would start the integration. To get rid of these extreme forces, ESPResSo allows to cap the generated forces to a reasonable level during a *warming up* phase until all particles in the system have gained a reasonable distance from each other.

```
set current_dist [analyze mindist]
for { set cap 20 } { $current_dist < 0.85 } { incr cap 20 } {
    inter ljforcecap $cap
    integrate 200
    set current_dist [analyze mindist]
}
inter ljforcecap 0
```

This loop integrates the system with an increasing force cap until the minimal distance between any pair of particles is larger than 0.85, where the forces become reasonable. The last command switches the force capping off again.

After warming up, we can start integrating the system. The main integration loop could look like this:

```
for {set i 0} { $i < 100 } { incr i} {
    integrate 200

    set t [setmd time]
    set e_tot [analyze energy total]
    set e_kin [analyze energy kinetic]
    set e_ele [analyze energy coulomb]
    puts "\tstep=$i,time=$t,e_tot=$e_tot,e_kin=$e_kin,e_ele=$e_ele"
}
```

This code block is the primary simulation loop and runs 20000 MD steps. Every `integ_steps` time steps, the different energy components are computed using the command **`analyze`** and printed out.

Of course, it would be nice if the observables would not only be printed to the screen, but also saved to a file, so that we can plot it later. Also, one would like to store some of the configurations for later analysis. Finally, it would also be nice to write out the configurations to a file to be able to visualize the system. Therefore, an expanded version of the main integration loop might look like this:

```
set ene_file [open "salt.ene" "w"]
puts $ene_file "\#t\tE_tot\tE_kin\tE_ele"

set vtf_file [open "salt.vtf" "w"]
writevsf $vtf_file
writevcf $vtf_file

for {set i 0} { $i < 100 } { incr i} {
    integrate 200

    set t [setmd time]
    set e_tot [analyze energy total]
    set e_kin [analyze energy kinetic]
    set e_ele [analyze energy coulomb]
    puts "\tstep=$i,time=$t,e_tot=$e_tot,e_kin=$e_kin,e_ele=$e_ele"
    puts $ene_file "[setmd time]\t$e_tot\t$e_kin\t$e_ele"

    writevcf $vtf_file

    analyze push 20
}

close $ene_file
close $vtf_file
```

The first few lines will open two files for output. `salt.ene` stores values of the total, kinetic and coulomb energies in a tabular fashion to be plotted by Gnuplot, XMGrace or any other plotting tool. Plotting the energies can be useful to check whether a system is equilibrated.

`salt.vtf` is a file in the VTF (**V**MD **T**rajectory **F**ormat) format. ESPResSo can write these files via the commands **`writevsf`** and **`writevcf`**, which will output the structure of the system (*i.e.* the number and type of atoms, their bonds, etc.) and the current coordinates of all particles respectively. The file can be easily visualized using VMD[10] (see figure 1).

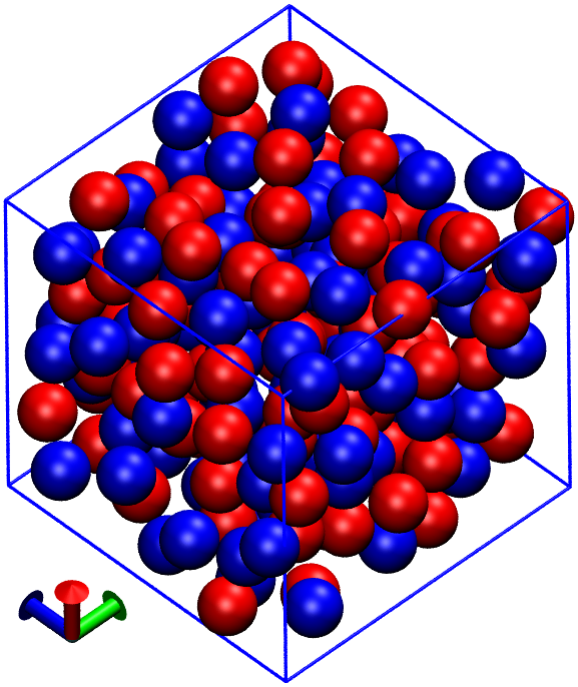


Figure 1: VMD Snapshot of the salt system

The command **`analyze`** `push 20` will store up to 20 configurations in memory, so that they can be analyzed afterwards. With these configurations, we can now investigate the system. As an example, we will calculate the averaged radial distribution function $g_{+-}(r)$, using the **`analyze`** command:

```
set rdf_file [open "salt.rdf" "w"]
puts $rdf_file "#r\tg+-"

set rdf01 [lindex [analyze <rdf> 0 1]
foreach v01 $rdf01 {
    set x [lindex $v01 0]
    set g01 [lindex $v01 1]

    puts $rdf_file "$x\t$g01"
}
close $rdf_file
```

The shown **analyze** <rdf> command returns the radial distribution function of particles of type 1 around particles of type 0 (*i.e.* of opposite charges). Changing the first two parameters to either "0 0" or "1 1" allows to determine the distribution for equal charges. Fig. 2 shows the resulting radial distribution functions, averaged over 100 configurations. In addition, the distribution for a neutral system is given, which can be obtained from our simulation script by simply not turning on P^3M .

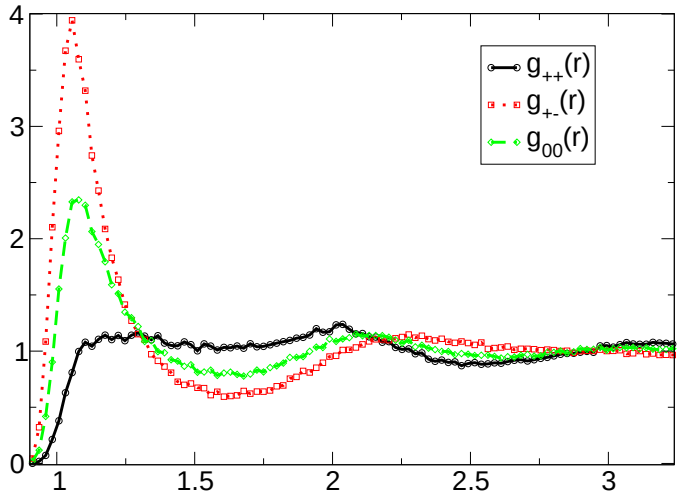


Figure 2: Radial distribution functions $g_{++}(r)$ between equal charges, $g_{+-}(r)$ for opposite charges, and $g_{00}(r)$ for an uncharged system.

The code example given before is still quite simple, and the reader is encouraged to try to extend the example, *e.g.* by using differently sized particle, or changing the interactions. If something does not work, ESPResSo will give comprehensive error messages, which should make it easy to identify mistakes. For real simulations, the simulation scripts can extend over thousands of lines of code and contain automated adaption of parameters or online analysis, up to automatic generation of data plots. Parameters can be changed arbitrarily during the simulation process, as needed for *e.g.* simulated annealing. The possibility to perform non-standard simulations without the need of modifications to the simulation core was one of the main reasons why we decided to use a script language for controlling the simulation core.

4 Concluding Remarks

We have given a short introduction into the simulation package ESPResSo . Of course, our hope is that the some of the readers consider ESPResSo as the simulation code base for their next projects. Informations regarding the latest version, or informations on how to participate in the further development of this package can be found on `http://www.espresso.mpg.de/`.

References

1. ESPResSo, "Homepage", 2004, `http://www.espresso.mpg.de`.
2. A. Arnold, B. A. Mann, H. Limbach, C. Holm, "ESPResSo – An Extensible Simulation Package for Research on Soft Matter Systems", in: Forschung und wissenschaftliches Rechnen 2003, Kurt Kremer, Volker Macho, (Ed.), vol. 63 of *GWDG-Bericht*, pp. 43–59. Gesellschaft für wissenschaftliche Datenverarbeitung mbh, Göttingen, Germany, 2004.
3. H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm, *ESPResSo – An Extensible Simulation Package for Research on Soft Matter Systems*, Comp. Phys. Comm., **174**, no. 9, 704–727, 2006.
4. Tcl/Tk, "Tool Command Language / ToolKit – Homepage", 2003, `http://tcl.activestate.com/`.
5. MPI Consortium, "The message passing interface (mpi) standard", 2004, `http://www.mcs.anl.gov/research/projects/mpi/Homepage`.
6. FFTW, "Fastest Fourier Transform in the West – Homepage", 2003, `http://www.fftw.org/`.
7. CVS, "Concurrent Versions System – Homepage", 2003, `http://www.cvshome.org/`.
8. Doxygen, "Doxygen – A documentation generation system", 2005, `http://www.doxygen.org/`.
9. A. Kolb, B. Duenweg, *Optimized constant pressure stochastic dynamics*, J. Chem. Phys., **111**, no. 10, 4453–59, 1999.
10. W. Humphrey, A. Dalke, K. Schulten, *VMD: Visual Molecular Dynamics*, Journal of Molecular Graphics, **14**, 33–38, 1996.
11. Christoph Junghans and Simn Poblete, *A reference implementation of the adaptive resolution scheme in ESPResSo*, Computer Physics Communications, **181**, no. 8, 1449 – 1454, 2010.
12. Gary S. Grest, Kurt Kremer, *Molecular dynamics simulation for polymers in the presence of a heat bath*, Phys. Rev. A, **33**, no. 5, 3628–31, 1986.

Schriften des Forschungszentrums Jülich
IAS Series

1. **Three-dimensional modelling of soil-plant interactions: Consistent coupling of soil and plant root systems**
   by T. Schröder (2009), VIII, 72 pages
   ISBN: 978-3-89336-576-0
   URN: urn:nbn:de:0001-00505

2. **Large-Scale Simulations of Error-Prone Quantum Computation Devices**
   by D. B. Trieu (2009), VI, 173 pages
   ISBN: 978-3-89336-601-9
   URN: urn:nbn:de:0001-00552

3. **NIC Symposium 2010**
   Proceedings, 24 – 25 February 2010 | Jülich, Germany
   edited by G. Münster, D. Wolf, M. Kremer (2010), V, 395 pages
   ISBN: 978-3-89336-606-4
   URN: urn:nbn:de:0001-2010020108

4. **Timestamp Synchronization of Concurrent Events**
   by D. Becker (2010), XVIII, 116 pages
   ISBN: 978-3-89336-625-5
   URN: urn:nbn:de:0001-2010051916

5. **UNICORE Summit 2010 – Proceedings**
   edited by A. Streit, M. Romberg, D. Mallmann (2010), iv, 123 pages
   ISBN: 978-3-89336-661-3
   URN: urn:nbn:de:0001-2010082304

6. **Fast Methods for Long-Range Interactions in Complex Systems Lecture Notes**
   edited by G. Sutmann, P. Gibbon, T. Lippert (2011), ii, 167 pages
   ISBN: 978-3-89336-714-6
   URN: urn:nbn:de:0001-2011051907

Parallel computing and computer simulations of complex particle systems including charges have an ever increasing impact in a broad range of fields in the physical sciences, e.g. in astrophysics, statistical physics, plasma physics, material sciences, physical chemistry, and biophysics. The present summer school, funded by the German Heraeus-Foundation, took place at the Jülich Supercomputing Centre from 6 – 10 September 2010. The focus was on providing an introduction and overview over different methods, algorithms and new trends for the computational treatment of long-range interactions in particle systems.

The Lecture Notes contain an introduction into particle simulation, as well as five different fast methods, i.e. the Fast Multipole Method, Barnes-Hut Tree Method, Multigrid, FFT based methods, and Fast Summation using the non-equidistant FFT. In addition to introducing the methods, efficient parallelization of the methods is presented in detail.

This publication was edited at the Jülich Supercomputing Centre (JSC) which is an integral part of the Institute for Advanced Simulation (IAS). The IAS combines the Jülich simulation sciences and the supercomputer facility in one organizational unit. It includes those parts of the scientific institutes at Forschungszentrum Jülich which use simulation on supercomputers as their main research methodology.

IAS Series
Volume 6
ISBN 978-3-89336-714-6

JÜLICH
FORSCHUNGSZENTRUM