Gernot Bukovnik

Application of Molecular Dynamics Simulation Methods on Complex Fluid-Dynamical Phenomena

DIPLOMARBEIT

zur Erlangung des akademischen Grades
Diplom-Ingenieur

Diplomstudium Technische Physik

TU Graz
Graz University of Technology

Technische Universität Graz

Betreuer:

Univ.-Prof. Dipl.-Ing. Dr.rer.nat. Wolfgang von der Linden

Institut für Theoretische Physik - Computational Physics

Graz, Mai 2010

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …………………………… ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…………………………… ………………………………………………..
date (signature)

Dedicated to my parents
for their love and support

Abstract

During the last decades computer simulations have become an important tool in many fields of science (e.g. in Astrophysics, Biochemistry, Solid-State Physics, or in Fluid Dynamics) and engineering. Especially molecular dynamics simulations offer a new, alternative approach to study complex fluid dynamical phenomena.
In this master thesis major concepts as well as theoretical foundations of Molecular Dynamics simulation methods have been summarized and advanced simulation methods have been presented. Building on this, a classical molecular dynamics simulation software package (consisting of the simulator *modelMD* and the graphical user interface *simEdit*), which meets the requirements of complex Molecular Dynamics simulations with up to 10^6 particles, has been developed.
Furthermore two representative, complex, fluid dynamical phenomena (namely the Rayleigh-Taylor instability and the Rayleigh-Benard convection) have been introduced and discussed. This representation has been complemented by a series of classical Molecular Dynamics simulations, which have been carried out with the software package modelMD.

Zusammenfassung

Computersimulationen haben sich in den letzten Jahrzehnten als ein wichtiges Werkzeug in vielen Bereichen der modernen Wissenschaft (wie beispielsweise der Astrophysik, Biochemie, Festkörperphysik oder der Fluiddynamik) und der Ingenieurwissenschaft etabliert. Vor allem Moleküldynamik-Simulationen bieten einen neuen, alternativen Ansatz um komplexe, fluiddynamische Phänomene zu untersuchen.
In dieser Diplomarbeit wurden das wesentliche Konzept sowie der theoretische Hintergrund von Moleküldynamik Simulationen zusammengefasst und fortgeschrittene Simulationsmethoden präsentiert. Dies bildete die Grundlage für die Erstellung eines klassischen Moleküldynamik Simulations Software-Packages (bestehend aus einem Simulator *modelMD* und einer grafischen Benutzeroberfläche *simEdit*), welches komplexe Simulationen mit bis zu 10^6 Teilchen ermöglicht.
Im weiterer Folge wurden zwei repräsentative, fluiddynamische Phänomene (die Rayleigh-Taylor Instabilität und die Rayleigh-Benard Konvektion) vorgestellt und diskutiert. Diese Darstellung wurde durch verschiedene Moleküldynamik-Simulationen, welche unter Verwendung des modelMD Software-Packages durchgeführt wurden, ergänzt.

Contents

Abstract v

Table of Contents vii

List of Figures ix

List of Tables xi

1 A Synopsis of Computational Physics 3

2 Molecular Dynamics Methods 7

2.1 Principle . . . 7
2.2 Initialization . . . 8
2.3 Potential and Force Calculation . . . 8
2.3.1 Pair Potentials . . . 9
2.3.2 Complex Many-Body Potentials . . . 12
2.3.3 Boundary Conditions . . . 13
2.4 Integrating the Equations of Motion . . . 14
2.4.1 Störmer-Verlet Method . . . 14
2.4.2 Runge-Kutta Method . . . 16
2.4.3 Predictor-Corrector Method . . . 17
2.5 Temperature and Pressure Control Mechanism . . . 18
2.5.1 Velocity Scaling Method . . . 18
2.5.2 Stochastic Method . . . 19
2.5.3 Feedback Method . . . 20
2.5.4 Constraint Method . . . 22
2.6 Measurements . . . 23
2.6.1 Energy . . . 24
2.6.2 Temperature . . . 24
2.6.3 Pressure . . . 25
2.7 Reduced Units . . . 25
2.8 Advanced Simulation Methods . . . 26
2.8.1 Cutoff Radius . . . 26
2.8.2 Verlet-Neighbor-List-Method and Linked-Cell-Method . . . 28
2.8.3 Parallel-Computing-Methods . . . 29

3 Software Packages modelMD and simEdit **35**
3.1 modelMD 35
3.1.1 Introduction 35
3.1.2 Input 36
3.1.3 Output 40
3.1.4 Further Work 42
3.2 simEdit 42
3.2.1 Introduction 42
3.2.2 Graphical User Interface 43
3.2.3 Output Data 45
3.2.4 Further Work 45

4 Complex Fluid-Dynamical Phenomena **47**
4.1 Rayleigh-Taylor Instability 47
4.1.1 Introduction 47
4.1.2 Modeling and Computational Details 50
4.1.3 Results and Discussion 51
4.2 Rayleigh-Benard Convection 58
4.2.1 Introduction 58
4.2.2 Modeling and Computational Details 60
4.2.3 Results and Discussion 61

Bibliography **71**

Acknowledgments **75**

List of Figures

1.1 Schematic presentation of the typical approach for numerical simulation [30] 4

2.1 Lennard-Jones potential function for spherical-symmetrical Argon 10
2.2 Periodic boundary conditions 13
2.3 Störmer-Verlet methods [30] 16
2.4 Lennard-Jones potential functions with and without smooth cutoff for spherical-symmetrical Argon 27
2.5 Verlet-Neighbor-List-Method 28
2.6 Linked-Cell-Method 29
2.7 Linked-Cell-Method: Benchmark results 30
2.8 Distributed-Memory-Machine 31
2.9 Shared-Memory-Machine 31
2.10 Parallelized MD program: Benchmark results 33

3.1 simulation.inp file 39
3.2 SimEdit GUI 44

4.1 Development of a Rayleigh-Taylor instability with a single wavelength perturbation [40] 48
4.2 Rayleigh-Taylor Instability 49
4.3 Rayleigh-Taylor Instability: Simulation system 51
4.4 Rayleigh-Taylor Instability: Time development of particle positions 53
4.4 Rayleigh-Taylor Instability: Time development of particle positions 54
4.5 Rayleigh-Taylor Instability: Time development of the particle density distribution . . . 55
4.5 Rayleigh-Taylor Instability: Time development of the particle density distribution . . . 56
4.6 Rayleigh-Taylor Instability: Growth of the mixing layer 57
4.7 Rayleigh-Benard Convection 58
4.8 Rayleigh-Benard Convection: Simulation system 60
4.9 Rayleigh-Benard Convection: Time development of streamlines 62
4.9 Rayleigh-Benard Convection: Time development of streamlines 63
4.10 Rayleigh-Benard Convection: Time development of the particle density distribution . . 64
4.10 Rayleigh-Benard Convection: Time development of the particle density distribution . . 65
4.11 Rayleigh-Benard Convection: Temperature distribution 66
4.12 Rayleigh-Benard Convection: Time development of streamlines for simulation system *A*100*b* 66
4.13 Rayleigh-Benard Convection: Horizontal density profile 67

List of Tables

2.1 Dimensionless variables 26
2.2 Linked-Cell-Method: Benchmark results 30
2.3 Benchmark results of the parallelized MD program package modelMD 33

3.1 simulation.inp input file: File format description 36
3.1 simulation.inp input file: File format description 37
3.1 simulation.inp file: File format description 38
3.2 posvel.inp file: File format description 40
3.3 extforce.inp file: File format description 41
3.4 thermostat.inp file: File format description 41

4.1 Rayleigh-Taylor Instability: Simulation parameters 51
4.2 Rayleigh-Benard Convection: Simulation parameters 61

Preface

During the last three decades computer simulations have become an important tool in many fields of science and engineering, since they combine theory (that is normally restricted to simplified models) and experiments (that are restricted by physical parameters). Computer simulations are not limited by those constraints, as they are based on theory and perform (virtual) experiments with any set of parameters.
The progress of computational science was especially made possible by the rapid development of high-performance parallel computer systems. Nowadays it is feasible to mimic and predict physical phenomena and thereby support or even replace complex and expensive experiments increasingly.

Chapter 1 gives a brief introduction to computational physics and numerical simulations, including the typical approach for numerical simulations, particle models as well as ab-initio and classical Molecular Dynamics.

Chapter 2 focuses on the theoretical foundations of classical Molecular Dynamics methods. It looks at the procedure of a typical Molecular Dynamics code and several types of potentials as well as algorithms to integrate the equations of motion and miscellaneous temperature and pressure control mechanism are introduced and discussed. Furthermore, measurement methods are introduced and reduced units are described. Eventually advanced simulation methods and parallelization strategies are delineated.

In chapter 3 a self-developed, high-performance classical Molecular Dynamics simulation software package, including the simulator as well as the graphical user interface, is presented.

Complex fluid dynamical phenomena with focus laid on the Rayleigh-Taylor instability and the Rayleigh-Benard convection are discussed in chapter 4. Important background knowledge as well as various publications concerning those topics are summarized and presented. This is complemented by a series of classical Molecular Dynamics simulations, carried out with the software-packages that are presented in chapter 3. The results are then evaluated and discussed.

Chapter 1

A Synopsis of Computational Physics

The main endeavor of natural science is to describe complex processes in nature as precisely as possible. The major objective is to create a mathematical formulation of those procedures, or in other words, to describe (with a system of differential and integral-equations) how specific quantities depend on other parameters and how they evolve over time. The final aim is to find a solid mathematical model that describes the phenomenon exactly.
With such a model it is possible to carry out detailed observed processes as well as predict them on a certain scale.

Most physical models that describe natural phenomena sufficiently, are so complex that they can't be solved analytically. Therefore, simplified and easier to solve models are developed, although their significance is limited compared to original ones. Unfortunately some phenomena can't be described with simplified mathematical models, as the law of gravitation, which can only be solved for two bodies at most. However, in many cases, one has to deal with more than just two interacting bodies, at microscopic as well as at macroscopic scales.
It is very unlikely to find exact physical laws just by observing processes in nature. A much more effective way is by creating exact conditions artificially in a laboratory. Experiments can then be carried out in this controlled environment and by modifying various parameters and comparing several results, physical laws can be derived.
Typically investigated phenomena range over several length and time scales, as from the quantum-mechanical observation of matter in the nano-scale to studies of the galaxy groups and clusters. Accordingly these phenomena also differ in their time scales (from 10^{-12} to 10^{17} seconds) as well as in their masses (from 10^{-27}kg for a single atom to 10^{40} kg for whole galaxies). Thus it appears that nature phenomena of interests are from a wider range of occurrences, whereas most of them are not reproducible in a laboratory, be it because of too short/long observation periods or too small/huge system sizes. Furthermore, expensive laboratory experiments can be avoided and it is possible to study systems, which can't be reproduced in a

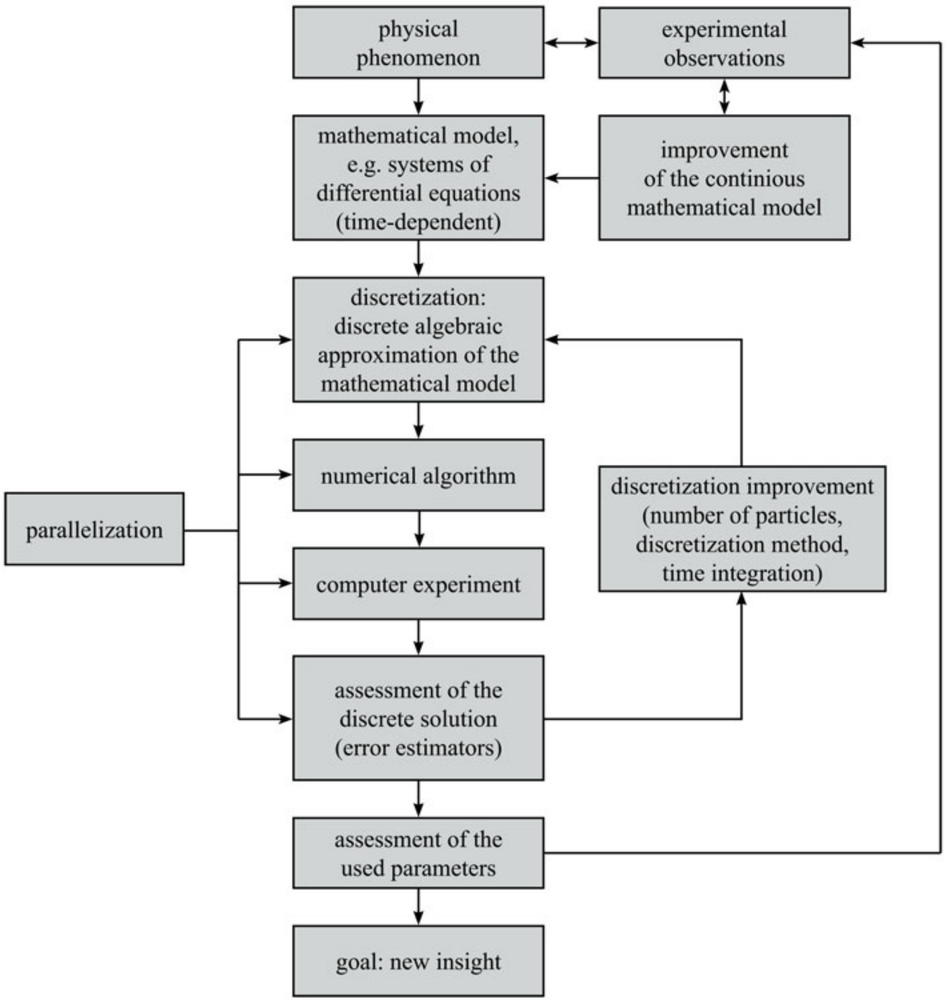

physical phenomenon
experimental observations
mathematical model, e.g. systems of differential equations (time-dependent)
improvement of the continious mathematical model
discretization: discrete algebraic approximation of the mathematical model
numerical algorithm
parallelization
discretization improvement (number of particles, discretization method, time integration)
computer experiment
assessment of the discrete solution (error estimators)
assessment of the used parameters
goal: new insight

Figure 1.1: *Schematic presentation of the typical approach for numerical simulation*

lab. On this account, the computational physics has established itself as a third major branch, apart from experimental and theoretical physics. Here it is possible to precalculate complex technical and physical processes numerically. Furthermore the computation power of modern computer systems makes it possible to determine more complex and realistic models as it would be possible with analytic methods.

The typical procedure of a computer simulation (as shown in figure 1.1) is: After a phenomenon is observed in nature a mathematical model is formulated. This model should describe reality as precisely as possible, although one has to make a compromise on accuracy and computational respectively memory effort that is required to solve this model.

Normally the resulting equations are continuous in time and space. Therefore they have to be discretized in order to be able to handle the problem numerically. Subsequently the system of equations is solved with an appropriate algorithm at some discrete points (in time or/and space). The more dense points are chosen, the more accurate the approximation becomes.

Finally the results are compared with field studies and either they become verified, or the simulation model has to be improved or revised.

Particle models play an important role in numerical simulations. In this approach a physical system is reproduced by a number of discrete particles and their mutual interaction. For this reason a classical system can be described by particle positions, their velocities and their interaction potential functions. In general, particles don't have to be constrained to represent very small bodies as atoms or molecules. They just represent basic modules of a physical model. Therefore single particles could also describe astronomical structures such as galaxies with billions of stars. Adequate physical properties (as position, velocity, mass and charge) are assigned to the particles to mimic those objects.
Many particle models use the laws of classical mechanics like Newton's equations of motion. This system of second order, ordinary differential equations describes the dependence of the acceleration upon a force, that is acting on it. This force results from the particle interaction and depends on the position of the particles. Trajectories of all particles arise as a result of the solution (with proper initial conditions) of this set of differential equations. This approach represents a deterministic method, wherefore all trajectories are predetermined for all times for certain initial conditions.
Particle models play an important role in many fields of research. In *fluid mechanics* it offers a new, alternative approach to study complex fluid dynamical phenomena such as the *Rayleigh-Taylor instability* or the *Rayleigh-Benard convection.* Therefore particle models are used exclusively in this work. In *solid-state physics* it enables the analysis of the already known and the research of new materials. Dynamics of macromolecules at atomic level can be investigated in *biochemistry* as well as theoretical models that can be validated in *astrophysics.*

For very small systems at the atomic level the classical Newton mechanics has to be replaced by the Quantum mechanics. This implies that the Schrödinger equation instead of the Newton one has to be used as the equation of motion. Then trajectories of all atoms can be achieved by solving the Schrödinger equations (with an appropriate Hamiltonian). However, an analytical and numerical solution of the Schrödinger equation is only possible for very simple systems, consisting of a few particles. Therefore approximations have to be used to solve more complex problems.
A famous method is the *Born-Oppenheimer approximation.* This approach involves the equations of motion of the nucleus and the equations of motion of the electrons are treated separately, because of the big mass difference. The Schrödinger equation for the core is then replaced by the Newton equation. Therefore the core moves to the classical model, though with a potential that results from the solution of the electronic Schrödinger equation. For this purpose further approximations have to be used, that are obtained by the *Hartree-Fock method* or the *Density functional theory.* This approach is known as *Ab Initio Molecular Dynamics (AIMD).* Anyway, even with this method the system size is still limited to a few thousand

atoms for complexity reasons.
To handle larger systems further simplifications have to be used, such as parameterized, analytical potential functions that depend only on the nucleon positions. This method is known as *classical Molecular Dynamics*. That approach makes it possible to handle problems with billions of particles, though quantum mechanical effects are not considered anymore.
In the following, basic methods and different approaches of classical Molecular Dynamics are described.

Chapter 2

Molecular Dynamics Methods

This chapter provides the basic concepts of classical Molecular Dynamics (MD) methods. The procedure of a typical MD code is presented and described in detail. Furthermore, important short-range pair-potentials are presented and complex long-range many-body-potentials are briefly discussed. Methods to integrate Newton's equations of motion are pictured and compared. Moreover miscellaneous temperature and pressure control mechanism are introduced and measurement methods of important observables are discussed and reduced units are described. Eventually advanced simulation methods and parallelization strategies, as well as different parallel-computer architectures are delineated and tested for their effectiveness.

2.1 Principle

MD simulations are in many respects, similar to a real, physical experiment. First a simulation system, that consists of N particles, is selected and initialized with certain parameters. Then Newton's equations of motions are solved and several measurements are taken as soon as the system reaches a state of equilibrium.
In summary it can be said, that MD programs are designed to carry out the following steps:

1. Initialization: Initial positions and velocities are set for all particles in the simulation system.

2. Calculation of the forces acting on the particles.

3. Integration of Newton's equation of motion for a small time-step δt.

4. Computation of observables.

Steps 2 and 3 are carried out in a loop and form the main core of MD programs. Subsequently those steps are described more precisely.

2.2 Initialization

To start a simulation, initial positions and velocities have to be set for all particles in the system. There are various ways to do that, as by defining a regular mesh in which every lattice point represents a particle position, or by defining a volume in which particles are positioned randomly for a certain particle density. Since equilibrium properties of a simulation system do not depend on the initial conditions, all reasonable start configurations are possible. However, if such dependencies are observed, the simulation was not run long enough in most cases and therefore the system has not reached an equilibrium state so far.
For the solid state of a certain system the initial conditions are typically chosen to prepare the system in the crystal structure of interest. For a simulation in a liquid phase the simulation can be prepared in any convenient crystal structure. At the temperature and density of a typical liquid state point, the solid state is not thermodynamically stable and therefore the crystal will melt afterwards.
In practice, the final, well equilibrated, state of an earlier simulation is often chosen as initial configuration for a new run. This simulation can then be computed with different parameters as temperature or pressure.
Most important in all positioning strategies is, that the initial particle positions are not too close to each other. Going below a critical distance-limit results in appreciable overlap of the atomic cores, which leads to unpredictable effects.
For the initialization of velocity vectors, their norm should follow a given velocity distribution (representing a distinct temperature) e.g. a Maxwell-Boltzmann velocity distribution, whereas their direction is distributed randomly.

2.3 Potential and Force Calculation

Interaction energies can be separated into *intramolecular* and *intermolecular* contributions. Former are forces that appear among atoms of a molecule in a covalent, ionic or metallic bond. Intramolecular forces are caused by diverse deformations as straining and turning of molecular bonds, or deformation of bond-angles.
Intermolecular forces act between stable molecules or individual atoms and include strong, long-range interactions (as the Coulomb interaction) and relatively weak, short-ranged ones (like the van-de-Waals interaction).
Since this work concentrates particularly on interactions between individual particles, intermolecular interactions are described in more detail in this chapter.

For a system consisting of N particles, the total potential U_{tot} can be separated into several contributions:

$$U_{\text{tot}} = \sum_i U_1(\vec{x}_i) + \sum_{i<j} U_2(\vec{x}_i, \vec{x}_j) + \sum_{i<j<k} U_3(\vec{x}_i, \vec{x}_j, \vec{x}_k) + \dots \tag{2.1}$$

where $\vec{x}_i$ is the position vector of particle i and U_1 represents a single-body potential (resulting from external potential fields), which therefore also includes simulation system walls. U_2, U_3 and all other potential terms of higher order, are interaction potentials between particle pairs (i, j), triples (i, j, k) and so on. In many models only single-, two- and three-body potentials are considered, since the small contribution of higher-order terms can be neglected.
Typically, computing three-body potentials is a very time-consuming task. Therefore those potentials are not considered explicitly in most models. However, they can be taken into account indirectly by combining them with a pair-interaction potential U_2 to an effective potential U_{eff}:

$$U_{\text{eff}} = \sum_{i<j} U_2(r_{ij}) \tag{2.2}$$

with $r_{ij} = |\vec{x}_i - \vec{x}_j|$.

2.3.1 Pair Potentials

Determining thermodynamical material properties with MD simulations requires accurate potential models, that reproduce realistic particle interactions. Usually such models are hard to specify. However, if the focus lays just on the computation of basic structural and dynamical properties, much more simple potential models, that emulate atoms or molecules as spherical-symmetrical balls with the mass m_i, can be used.

Lennard-Jones Potential

One of the most important and widely used potential model is the *Lennard-Jones (LJ) potential.* It consists of an attractive and a repulsive term and reads, in its original form:

$$U(r_{ij}) = \frac{p}{p-q}\left(\frac{p}{q}\right)^{\frac{q}{p-q}} \epsilon \left[\left(\frac{\sigma}{r_{ij}}\right)^p - \left(\frac{\sigma}{r_{ij}}\right)^q\right] \tag{2.3}$$

Typically for the attractive term $q = 6$ and for the repulsive term $p = 12$ is chosen, which leads to the well known Lennard-Jones-(12-6) potential:

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^6\right] \tag{2.4}$$

where the energetic parameter ϵ donates the depth of the potential well. σ accords to the zero crossing of the potential and is interpreted as the particle diameter. The potential is repulsive for small intermolecular distances and has it's minimum $U(r_{\min}) = -\epsilon$ at $r_{\min} = \sqrt[6]{2}\sigma$. The

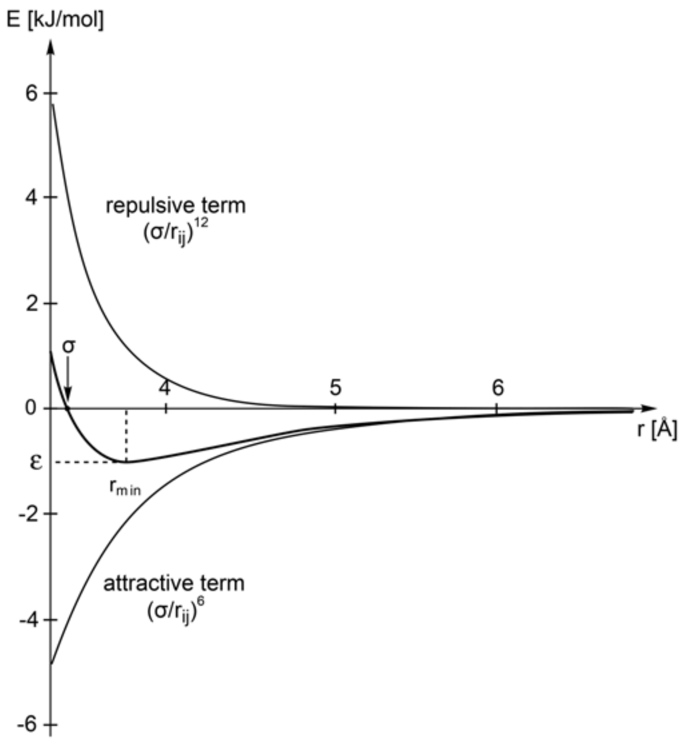


Figure 2.1: *Lennard-Jones potential function for spherical-symmetrical Argon with $\epsilon = 1.00$ kJ/mol and $\sigma = 3.41$ Å. Additionally, the attractive and the repulsive term are plotted as well.*

steep slope of the repulsive term for small distances, can be explained quantum mechanically with the *Pauli exclusion principle*, which states that two electrons, in a single atom, can not have the same four quantum numbers. Therefore, overlapping electron shells are forbidden. On the other hand, the attractive term, the so-called *van der Waals interaction* or *dispersion interaction*, dominates for long intermolecular distances. This interaction is relatively weak compared to normal chemical bonds and are caused by correlations in the fluctuating polarizations of nearby particles. Figure 2.1 shows the Lennard-Jones potential function for spherical-symmetrical Argon.
The resulting force of the Lennard-Jones-(12-6) potential reads:

$$\vec{F}_{ij} = -\nabla U(r_{ij}) = \frac{48\epsilon}{\sigma^2} \left[\left(\frac{\sigma}{r_{ij}} \right)^{14} - \frac{1}{2} \left(\frac{\sigma}{r_{ij}} \right)^{8} \right] \vec{r}_{ij} \tag{2.5}$$

Equations (2.1) and (2.5) act on the simplified assumption, that only one type of particles, which properties are characterized by the parameters ϵ and σ, are used. However, in practice interactions between different particle types have to be modeled as well. To compute the interaction parameters ϵ and σ for disparate particles, empiric combination rules like the *Lorentz-Berthelot* mixing rules are used:

$$\sigma = \frac{1}{2} \cdot (\sigma_{kk} + \sigma_{ll}) \tag{2.6}$$

$$\epsilon_{kl} = \sqrt{\epsilon_{kk} \cdot \epsilon_{ll}} \tag{2.7}$$

where k and l represent different particle types.
The Lorentz-Berthelot mixing rules allow an empirical motivated modeling of simple heterogeneous simulation systems.

Long-Range Pair-Potentials

In computational physics, the most important criterion for the implementation of a potential function, is its effective range. Short-range potentials, as the Lennard-Jones one, that can be approximated with a cutoff radius after that the potential function vanishes can be modeled relative easily, since for force calculations only the nearest neighbors have to be taken into account (see section 2.8.1).
For long-range potentials a cutoff radius can not be defined without taking a massive loss of accuracy into account. Therefore all possible particle pairs have to be considered, which complicates the computation task.
Important examples for slow-decreasing, long-range potentials are the *intermolecular gravitation potential*

$$U_{\mathrm{grav}}(r_{ij}) = -G_{\mathrm{grav}} \frac{m_i m_j}{r_{ij}} \tag{2.8}$$

and the *Coulomb potential*

$$U_{\mathrm{coul}}(r_{ij}) = \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} \tag{2.9}$$

where G_{grav} represents the gravitational constant, m_i and m_j the particle masses, r_{ij} is the distance between two ions, q_i and q_j the electric charges of particle i and j respectively and ϵ_0 is the electrical permittivity of space.
Such long-range potential functions that can be separated in a short-range and a long-range term, which are examined separately:

$$U = U_{\mathrm{short}} + U_{\mathrm{long}} \tag{2.10}$$

U_{short} can be any fast-decreasing potential. The long-range term yields in the case of the Coulomb potential with the electrostatic potential for N point charges at positions $\vec{x}$:

$$U_{\mathrm{elstat}} = \frac{1}{4\pi\epsilon_0} \sum_i \sum_{i<j} q_i q_j \frac{1}{|\vec{x}_j - \vec{x}_i|} \tag{2.11}$$

With a given charge distribution, it is possible to compute the overall potential Φ, using the Poisson equation for electric fields. All other forces can be derived as gradient from this equation. The Poisson equation, or potential equation, for the electric fields is:

$$-\Delta\Phi(\vec{x}) = \frac{1}{\epsilon_0}\rho(\vec{x}) \tag{2.12}$$

which reads in its integral expression:

$$\Phi(\vec{x}) = \frac{1}{4\pi\epsilon_0}\int_{\mathbb{R}^3}\frac{\rho(\vec{x}')}{|\vec{x}-\vec{x}'|}d\vec{x}'. \tag{2.13}$$

where $\rho(\vec{x})$ is the charge distribution of the simulation system and ϵ_0 represents the permittivity.

In simulation systems with long-range potentials and periodic boundary conditions, in addition to the particle interactions in the original box interactions with virtual particles in all other copies of the primary system have to be considered as well. The total electrostatic energy of a periodic system yields:

$$U_{\text{elstat}} = \frac{1}{2}\frac{1}{4\pi\epsilon_0}\sum_{\vec{n}\in\mathbb{Z}^3}\sum_i\sum_{\substack{j\\ i\neq j\ for\ n=0}} q_i q_j \frac{1}{|\vec{x}_j^{\vec{n}}-\vec{x}_i|} \tag{2.14}$$

where $\sum_i$ summarizes only over all particles within the simulation box, whereas $\sum_{\vec{n}} = \sum_{\vec{n}_1}\sum_{\vec{n}}\sum_{\vec{n}_3}\dots$ summarizes over all periodic images of the simulation space. $\vec{x}_j^{\vec{n}} = \vec{x}_j + (n_1 \cdot L_1, n_2 \cdot L_2, n_3 \cdot L_3)$ indicates the positions of all periodic replicas of particle j. Here the interaction of a particle with itself is excluded, though an interaction with its copies is considered.
Equation 2.14 is the origin for most methods to solve the electrostatic problem (see [12,14,30]).

2.3.2 Complex Many-Body Potentials

The modelizing power of the pair-potentials discussed above is limited in their implementation. They are absolute capable to model noble gases as well as fluids, where atoms interact only by Van der Waals forces, though they can't be used to simulate complex interactions, as those which occur in metals or molecules, realistically. For this task, potential functions, that also consider interactions between atoms of the same molecule, have to be used. The idea behind that approach is to use the particle density, respectively the coordination number (which represents the number of nearest neighbors in a molecule or crystal). The bond between the atoms decreases with increasing particle density. This fact leads to potential functions which include a pair-potential term as well as an additional term, that considers the coordination number and therefore the density. Example potential methods are the *Glue-Model*, the *Embedded-Atom-Method*, the *Finnis-Sinclair-Potentials* or the *Effective-Medium-Theory.* They all differ in the way how the coordination number is included in the potential.
Even more complex models are used to model e.g. semiconductors like silicon. Those potentials

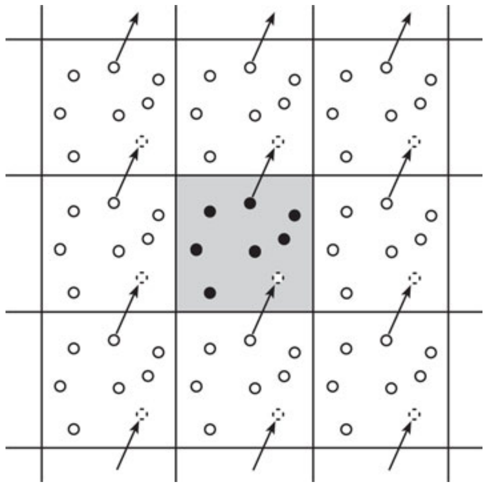


Figure 2.2: *Periodic boundary conditions applied for a two-dimensional simulation space: This approach results in an infinite array of boxes, which are replicas of the original box (shaded in gray). If a particle moves from the central box into another one (indicate by the arrow), it is replaced by it's own image that moves into the central box. This movement is replicated across all the boxes.*

use the *Bond Order* concept, which declares that the strength of a chemical bond depends on its local environment. This approach is closely related to the Glue-Model.

2.3.3 Boundary Conditions

In MD simulations the simulation space has to be truncated to a size, that is computationally ascertainable. Therefore, artificial boundaries, that consider and replace the removed atoms, have to be defined. On this account, an important task of boundary conditions is to limit the effects of an artificially introduced finite simulation space.
Ideal, so-called, *exact boundary conditions* provide an approximated solution, that accord to a system without applied boundaries.

In order to simulate huge particle systems, as they are needed especially for fluid simulations, boundary conditions that mimic the presence of a virtual, infinite particle system, which surrounds the original simulation box, have to be introduced. Typically, *periodic boundary conditions (pbc)* are applied in such cases. In this approach the original system represents a cell of an infinite periodic lattice of identical cells, as shown in figure 2.2. A particle interacts with all other particles in the infinite simulation space, including its own images in duplicated cells. This method makes it possible to perform complex simulations with a relatively small number of particles which experience forces as though they were in a bulk solution.
For certain simulations, periodic boundary conditions are not convenient, namely for long-range potentials. Here all possible particle pairs have to be taken into account and therefore infinite sums rather than finite ones have to be calculated. Solution methods for this kind of problem were already discussed in section 2.3.1. However, in most cases short-range potentials are used, which can be approximated to vanish after a certain cutoff radius (see section 2.8.1).

Therefore periodic boundary conditions don't represent a complex problem for these potential functions.
Additionally there are other important boundary conditions like *perfectly reflecting* and *diffusely reflecting* walls. Here new velocities and directions are assigned to particles that approach the boundary. At perfectly reflecting walls one component of the velocity vector is multiplied by -1, at diffusely reflecting ones the original velocities are changed to new ones with the same magnitudes but randomly chosen inward directions. In this case the determinism is lost.
Boundaries also provide a simple opportunity to connect heat-bathes to the system (see section 2.5), or to remove particles from the simulation space.
Furthermore it is possible to apply any kind of additional force field to the boundaries.

2.4 Integrating the Equations of Motion

The intention of MD simulations is to compute the chronological development of a system in the molecular scale.
Knowing the force acting on each particle in the simulation and using Newton's equation of motion

$$m_i \ddot{\vec{x}}_i = \vec{F}_i, \tag{2.15}$$

where m_i is the mass, $\vec{x}_i$ the position and $\vec{F}_i$ the force acting on the i-th particle, one gets a system of N second-order differential equations, which solutions equals to the trajectories of the particles. It is also possible to convert equation (2.15) to a system of first-order equations:

$$\dot{\vec{x}}_i = \vec{v}_i \qquad \dot{\vec{v}}_i = \frac{1}{m_i}\vec{F}_i. \tag{2.16}$$

Hence a suitable algorithm to integrate Newton's equation of motion is essential for MD programs. There are different numerical approaches that intend to solve such problems. In the following a brief introduction to the most important methods is provided.

The first step to make a mathematical problem suitable for numerical computing, is the process of transferring continuous equations into it's discrete counterparts, also known as *discretization* [30]. These second-order differential equations have to be transferred to a system of equations, which solutions approximate the solutions of the original system at some selected points. This corresponds to the computation of new particle positions and velocities out of old ones via the according forces.

2.4.1 Störmer-Verlet Method

The *Störmer-Verlet* algorithm [30] computes the position at the time-step t_{n+1} from the positions at time t_{n-1} and t_n and the corresponding force at time t_n, whereas the velocity is not needed.

A Taylor expansion of the position vector $\vec{x}$ at $t_n + \delta t$ and $t_n + \delta t$ yields:

$$x(t_n + \delta t) = x(t_n) + \delta t \dot{x}(t_n) + \frac{\delta t^2}{2}\ddot{x}(t_n) + \frac{\delta t^3}{6}\dddot{x}(t_n) + \mathcal{O}(\delta t^4) \tag{2.17}$$

$$x(t_n - \delta t) = x(t_n) - \delta t \dot{x}(t_n) + \frac{\delta t^2}{2}\ddot{x}(t_n) - \frac{\delta t^3}{6}\dddot{x}(t_n) + \mathcal{O}(\delta t^4) \tag{2.18}$$

Adding (2.18) to (2.17) yields for time evolution the *Standard-Form* of the Strömer-Verlet-Algorithm:

$$\vec{x}_i^{\,n+1} = 2\vec{x}_i^{\,n} - \vec{x}_i^{\,n-1} + \delta t^2 \frac{\vec{F}_i^{\,n}}{m_i} + \mathcal{O}(\delta t^4) \tag{2.19}$$

with $\vec{x}_i^{\,n} := \vec{x}_i(t_n)$, $\vec{v}_i^{\,n} := \vec{v}_i(t_n)$ and $\vec{F}_i^{\,n} := \vec{F}_i(t_n)$. It can be seen that only positions at time-steps t_n, t_{n-1} and the force at time-step t_n are needed for the calculation. Velocities are not explicitly calculated in (2.19), but they can be approximated as:

$$\vec{v}_i^{\,n} = \frac{\vec{x}_i^{\,n+1} - \vec{x}_i^{\,n-1}}{2\delta t} \tag{2.20}$$

A major disadvantage of this method are potential numerical rounding errors that arise from the addition of numbers with very different sizes. There are two other formulations of the Störmer-Verlet-Algorithm, namely the *Leap-Frog-Algorithm* and the *Velocity-Störmer-Verlet-Algorithm*, which both reduce the effect of such roundoff errors.

Leap-Frog-Algorithm

The *Leap-Frog-Algorithm* calculates new velocities $\vec{v}_i^{\,n+1/2}$ directly from velocities $\vec{v}_i^{\,n-1/2}$ and the force $\vec{F}_i^{\,n}$:

$$\vec{v}_i^{\,n+1/2} = \vec{v}_i^{\,n-1/2} + \frac{\delta t}{m_i}\vec{F}_i^{\,n} \tag{2.21}$$

The positions of all atoms are one time-step in advance compared to the new velocities:

$$\vec{x}_i^{\,n+1} = \vec{x}_i^{\,n} + \delta t \vec{v}_i^{\,n+1/2} \tag{2.22}$$

Since now the velocities are not calculated for the same time-step as the new positions, $\vec{v}_i^{\,n+1}$ is computed by the average determination:

$$\vec{v}_i^{\,n} = \frac{1}{2}\left(\vec{v}_i^{\,n+1/2} + \vec{v}_i^{\,n-1/2}\right) \tag{2.23}$$

Velocity-Störmer-Verlet-Algorithm

The *Velocity-Störmer-Verlet-Algorithm* calculates new velocities by combining (2.19) and (2.20):

$$\vec{v}_i^{\,n} = \frac{\vec{x}_i^{\,n+1} - \vec{x}_i^{\,n-1}}{2\delta t} = \frac{\vec{x}_i^{\,n}}{\delta t} - \frac{\vec{x}_i^{\,n-1}}{\delta t} + \frac{\vec{F}_i^{\,n}}{2m_i}\delta t \tag{2.24}$$

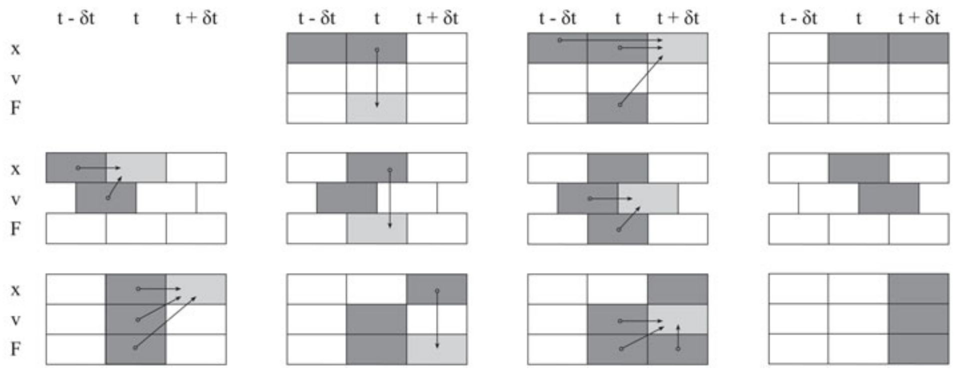


Figure 2.3: *Calculation processes of the three Störmer-Verlet methods: The first line shows the Standard-Form of the Strömer-Verlet-Algorithm (2.19), the second line the Leap-Frog-Algorithm ((2.21) and (2.22)) and the third line the Velocity-Störmer-Verlet-Algorithm ((2.25) and (2.26)). Dark-gray areas symbolize calculated data (evaluated at an earlier time-step), while light-gray ones denote data that is computed at this time-step. Arrows represent calculation processes.*

Adding $\vec{v}_i^{\,n+1}$ and using equation (2.20) leads to:

$$\vec{v}_i^{\,n+1} = \vec{v}_i^{\,n} + \frac{(\vec{F}_i^{\,n} + \vec{F}_i^{\,n+1})\delta t}{2m_i} \tag{2.25}$$

Positions are calculated from (2.19) and (2.20):

$$\vec{x}_i^{\,n+1} = \vec{x}_i^{\,n} + \delta t \vec{v}_i^{\,n} + \frac{\vec{F}_i^{\,n}\delta t^2}{2m_i} \tag{2.26}$$

The calculation processes of the three Störmer-Verlet methods are shown in figure 2.3.

2.4.2 Runge-Kutta Method

The *Runge-Kutta* method [30] is a *single-value method* that is used to approximate solutions of ordinary differential equations. The most commonly used Runge-Kutta method is probably the *classical fourth-order Runge-Kutta* method, also referred as *RK4*:

$$\vec{x}_i^{\,n+1} = \vec{x}_i^{\,n} + \frac{1}{6}\delta t\,(k_{11} + 2k_{12} + 2k_{13} + k_{14}) \tag{2.27}$$

$$\vec{v}_i^{\,n+1} = \vec{v}_i^{\,n} + \frac{1}{6}\delta t\,(k_{11} + 2k_{12} + 2k_{13} + k_{14}) \tag{2.28}$$

with coefficients:

$$\begin{array}{lllllll} k_{11} & = & \vec{v}_i^{\,n} & & k_{21} & = & \vec{a}(t, \vec{r}_i^{\,n},\ \vec{v}^{\,(i)}(t)) \\ k_{12} & = & \vec{v}_i^{\,n} + \frac{\delta t}{2} k_{21} & & k_{22} & = & \vec{a}\left(t + \frac{\delta t}{2},\ \vec{r}_i^{\,n} + \frac{\delta t}{2} k_{11},\ \vec{v}_i^{\,n} + \frac{\delta t}{2} k_{21}\right) \\ k_{13} & = & \vec{v}_i^{\,n} + \frac{\delta t}{2} k_{22} & & k_{23} & = & \vec{a}\left(t + \frac{\delta t}{2},\ \vec{r}_i^{\,n} + \frac{\delta t}{2} k_{12},\ \vec{v}_i^{\,n} + \frac{\delta t}{2} k_{22}\right) \\ k_{14} & = & \vec{v}_i^{\,n} + \delta t\, k_{23} & & k_{24} & = & \vec{a}\left(t + \delta t,\ \vec{r}_i^{\,n} + \delta t\, k_{13},\ \vec{v}_i^{\,n} + \delta t\, k_{23}\right) \end{array} \tag{2.29}$$

This method features a much more accurate approximation than the Störmer-Verlet approach, though such a high precision is not always required in MD simulations.

2.4.3 Predictor-Corrector Method

The *Predictor-Corrector* method [12] belongs to the family of the *multiple-value* methods. In contrast to single-value methods like the Leap-Frog or Runge-Kutta method, this approach uses data of one or more earlier time-steps. There are two major forms of the Predictor-Corrector method, whereas the *Adams-Bashforth-Moulton* approach uses a certain amount of accelerations of earlier time-steps. The *Nordsieck* method uses higher order derivations of the acceleration at the current time-step. Both methods require higher computation amount than the Leap-Frog method, since more calculations have to be executed and additional data has to be saved. However, compared to the Runge-Kutta method the computation amount is still significant lower. In the following we will focus on the Adams-Bashforth-Moulton approach. This algorithm proceeds in three steps. First, the *predictor step* calculates estimated positions and velocities.

$$\vec{x}_i^{n+1} = \vec{x}_i^{n} + \delta t \vec{v}_i^{n} + \delta t^2 \sum_{i=1}^{k-1} \alpha_i \vec{a}_i^{n+(1-i)} \tag{2.30}$$

$$\delta t \vec{v}_i^{n+1} = \vec{x}_i^{n+1} - \vec{x}_i^{n} + \delta t^2 \sum_{i=1}^{k-1} \alpha_i' \vec{a}_i^{n+(1-i)} \tag{2.31}$$

with coefficients α_i and α_i' that satisfy

$$\sum_{i=1}^{k-1}(1-i)^q \alpha_i = \frac{1}{(q+1)(q+2)} \qquad \text{and} \qquad \sum_{i=1}^{k-1}(1-i)^q \alpha_i' = \frac{1}{(q+2)}. \tag{2.32}$$

Using these new positions (2.30) and velocities (2.31), forces and the resulting accelerations can be calculated in the second step. In general they will differ from the predicted accelerations. In the third step, this difference can be used to calculate corrected positions and their derivatives.

$$\vec{x}_i^{n+1} = \vec{x}_i^{n} + \delta t \vec{v}_i^{n} + \delta t^2 \sum_{i=1}^{k-1} \beta_i \vec{a}_i^{n+(2-i)} \tag{2.33}$$

$$\delta t \vec{v}_i^{n+1} = \vec{x}_i^{n+1} - \vec{x}_i^{n} + \delta t^2 \sum_{i=1}^{k-1} \beta_i' \vec{a}_i^{n+(2-i)} \tag{2.34}$$

with $\vec{a}_i^n := \vec{a}_i(t_n)$ and the coefficients α_i and α_i' that satisfy

$$\sum_{i=1}^{k-1}(2-i)^q \beta_i = \frac{1}{(q+1)(q+2)} \qquad \text{and} \qquad \sum_{i=1}^{k-1}(2-i)^q \beta_i' = \frac{1}{(q+2)}. \tag{2.35}$$

2.5 Temperature and Pressure Control Mechanism

An important question of MD simulations is the choice of the ensemble type. In statistical mechanics terms, basic MD simulations work with *microcanonical (NVE) ensembles*, where number of particles (N), volume (V) and energy (E) are constant. Constant volume (V) and temperature (T) experiments accord to *Canonical (NVT) ensembles* and experiments with constant pressure (P) and temperature (T) (like in a laboratory) to *isothermal-isobar (NPT) ensembles*. Among them, NPT ensembles are definitely the most important ones.
Temperature and pressure control mechanisms are powerful tools to provide NPT ensembles as well as control over temperature and pressure during a simulation (in order to study physical and chemical phenomenons like phase transitions). After initializing the system, those technique are used to prepare the ensemble in the so-called *equilibrium-phase*. Subsequently the actual simulation run (production phase) starts.
There are various ways to achieve constant temperature and pressure. A naive approach would be the *velocity scaling method*, where velocities of all particles are repeatedly adjusted to enforce a certain, constant temperature.
There are more serious procedures to achieve constant temperatures such as by coupling the system to a virtual, ideal *heat-bath*. In this approach a stochastic mechanism is required to adjust particle velocities in order to reproduce the effects of a heat-bath, which, unfortunately, infracts the deterministic nature of the dynamics.
Well investigated and widely used approaches are *feedback* and *constraint methods*. In the former, feedback algorithms are used to correct derivations from the preset mean value of the controlled parameters. These corrected values fluctuate, but these fluctuations can be adjusted. In the constraint methods, the controlled parameters are kept strictly constant by extending Newton's equations of motion with additional constraints. Then the temperature can be controlled by introducing constraints that adjust the the system's kinetic energy. In the following these different approaches are described in detail.

2.5.1 Velocity Scaling Method

In this naive, but relatively simple to implement approach, the temperature, which is proportional to the mean square velocity, is repeatedly scaled by multiplying all particle velocities with a time-depended factor β:

$$\vec{v}_i^{\,n} := \beta^n \vec{v}_i^{\,n} \tag{2.36}$$

with

$$\beta^n = \left(\frac{T_{\text{kin}}^D}{T_{\text{kin}}^n}\right)^{1/2} = \left(\frac{T^D}{T^n}\right)^{1/2} \tag{2.37}$$

where T_{kin}^D respectively T^D represent the desired kinetic energy and the corresponding temperature. T_{kin}^n respectively T^n represent the kinetic energy and the temperature at time-step n. Depending on T^D and T^n, β can cover a wide range of values, which can strongly affect the energy distribution of the simulation system. Therefore in most cases the temperature is scaled with a modified factor:

$$\beta_\gamma^n = \left[1 + \gamma \left(\frac{T_{\text{kin}}^D}{T_{\text{kin}}^n}\right)\right]^{1/2} = \left[1 + \gamma \left(\frac{T^D}{T^n}\right)\right]^{1/2} \tag{2.38}$$

where $\gamma \in [0, 1]$ represents a dumping factor. $\gamma = 1$ yields (2.37), while $\gamma = 0$ results in no velocity scaling.

2.5.2 Stochastic Method

Another approach to effect the temperature of a simulation system is by coupling it to a virtual, ideal heat-bath. However, a major disadvantage of this method is that the deterministic nature of the dynamics is destroyed.

The coupling can easily be realized by using the boundary conditions, i.e. by setting the walls to a certain, fixed temperature. To particles that interact with these walls random directions with a magnitude based on the temperature from a Maxwell-Boltzmann velocity distribution are assigned:

$$P(v) = 4\pi \left(\frac{m}{2\pi k_B T}\right)^{3/2} v^2 \exp\left\{-\frac{mv^2}{2k_B T}\right\} \tag{2.39}$$

where m is the mass of the particle, k_B the Maxwell-Boltzmann constant and T the thermodynamical temperature of the wall.

Typically it is easier to determine each component of the velocity separately from a Gaussian distribution (2.40), since it is a complex task to invert $\int_0^V P(v)\, dv$ analytically.

$$P(v_i) = \left(\frac{m}{2\pi k_B T}\right)^{1/2} \exp\left\{-\frac{mv_i^2}{2k_B T}\right\} \tag{2.40}$$

Despite the easy implementation, this method can't be used for simulation systems where boundary conditions are already applied in another way, as in periodic or open systems. Moreover, due to the decreasing wall-volume ratio for increasing simulation space, this approach is little effective to bigger systems.

An alternative approach to avoid those problems is the, *Langevin dynamic.* The idea behind this mathematical model is that all particles suspend in a fictional, viscous medium, where they encounter a fluctuating, random force due to the Brownian motion of the virtual fluid-particles. Therefore Newton's equations of motion (2.16) are extended with additional terms and read:

$$\dot{\vec{x}}_i = \vec{v}_i \qquad \dot{\vec{v}}_i = \frac{1}{m_i}\vec{F}_i - \eta\vec{v}_i + \vec{G}(t, T). \tag{2.41}$$

where η denotes the viscosity of the virtual medium and $\vec{G}(t, T)$ a random force with zero mean, that satisfies:

$$\langle G_i(t)\, G_i(t')\rangle = 2\eta k_B T \delta(t - t') \tag{2.42}$$

in order to ensure that the work done by $\vec{G}(t,T)$ is dissipated by viscous friction (fluctuation-dissipation theorem). An analog approach was implemented in the modelMD software package (see section 3.1). Here the particles that are connected to a heat-bath, are surrounded and repeatedly hit by a virtual medium, consisting of smaller particles with a Maxwell-Boltzmann velocity distribution. Therefore, an additional force $\vec{G}$ is applied to hit particles:

$$\vec{G}_i = 2\eta \left(\vec{v}^{\,\text{rand}} - \vec{v}_i \right) \tag{2.43}$$

where $\vec{v}^{\,\text{rand}}$ is the random velocity of the fictional particles and η represents the density and therefore the viscosity of the virtual fluid [26].

2.5.3 Feedback Method

Controlled Temperature

Since the temperature is proportional to the mean square velocity, the idea is to control the temperature by adjusting the time progression rate. Therefore the virtual, scaled time variable t' has to be introduced:

$$dt' = s(t)\; dt \tag{2.44}$$

where s represents a strictly positive function to rescale time and t represents the real, physical time.

The Lagrangian for this extended system yields:

$$\mathcal{L} = \frac{1}{2} m s^2 \sum_i \left(\dot{\vec{x}}_i \right)^2 - \sum_{i<j} U\left(\vec{r}_{ij}\right) + \frac{1}{2} M_s \dot{s}^2 - n_f T \ln s \tag{2.45}$$

where the dot donates d/dt', T is the desired temperature, $\vec{r}_{ij} = \vec{x}_i - \vec{x}_j$ and $n_f = 3N_a + 1$ is the number of degrees of freedom (which can be reduced due to momentum conservation). M_s represents a "virtual mass", that is required in order to define an equation of motion for the new "coordinate" s.

The lagrangian equations of motion yield:

$$\ddot{\vec{x}}_i = \frac{1}{ms^2}\vec{F}_i - \frac{2\dot{s}}{s}\dot{\vec{x}}_i \tag{2.46}$$

$$M_s \ddot{s} = ms \sum_i \left(\dot{\vec{x}}_i \right)^2 - \frac{n_f T}{s} \tag{2.47}$$

where the dot donates again d/dt'.

The virtual time t' depends on the entire history of the system:

$$t' = \int_0^t s(t)\, dt \tag{2.48}$$

The Lagrangian (2.45) is defined in terms of virtual time. In physical time units (2.45) reads:

$$\begin{aligned} \ddot{\vec{x}}_i &= \frac{1}{m}\vec{F}_i - \frac{2\dot{s}}{s}\dot{\vec{x}}_i && (2.49) \\ \ddot{s} &= \frac{\dot{s}}{s} + \frac{G_1 s}{M_s} && (2.50) \end{aligned}$$

with

$$G_1 = m\sum_i \left(\dot{\vec{x}}_i\right)^2 - n_f T \qquad (2.51)$$

where the dot now donates d/dt.

Here (2.49) represents Newton's equation of motion with an additional friction-like term, proportional to the velocity, while (2.50) defines the feedback mechanism, where s is varied to regulate the temperature.

Controlled Pressure and Temperature

The feedback method also allows to control the pressure by adjusting the volume of the simulation space. Therefore the virtual space coordinate $\vec{x}'$ has to be introduced:

$$\vec{x}' = \frac{1}{V^{1/3}}\vec{x} \qquad (2.52)$$

where V represents a function that controls the volume of the simulation space and therefore the pressure.

The Lagrangian for this extended system (including temperature as well as pressure control) takes the form:

$$\begin{aligned} \mathcal{L} =& \frac{1}{2}mV^{2/3}s^2\sum_i \left(\vec{v}_i\,'\right)^2 - \sum_{i<j} U\left(V^{1/3}\vec{r}_{ij}\,'\right) + \frac{1}{2}M_s\dot{s}^2 + \frac{1}{2}M_V\dot{V}^2 \\ & - n_f T\ln s - pV \end{aligned} \qquad (2.53)$$

where the dot donates d/dt', p is the desired pressure and M_s another "virtual mass". After transforming into physical time units and coordinates equation (2.53) reads:

$$\begin{aligned} \ddot{\vec{x}}_i &= \frac{1}{mV^{1/3}s^2}\vec{F}_i - \left(\frac{2\dot{s}}{s} + \frac{2\dot{V}}{3V}\right)\dot{\vec{x}}_i && (2.54) \\ \ddot{s} &= \frac{\dot{s}}{s} + \frac{G_1 s}{M_s} && (2.55) \\ \ddot{V} &= \frac{\dot{s}\dot{V}}{s} + \frac{G_2 s^2}{3M_v V} && (2.56) \end{aligned}$$

with

$$G_1 = mV^{2/3}\sum_i \left(\dot{\vec{x}}_i\right)^2 - n_f T \tag{2.57}$$

$$G_2 = mV^{2/3}\sum_i \left(\dot{\vec{x}}_i\right)^2 + V^{1/3}\sum_{i<j}\vec{r}_{ij}\cdot\vec{F}_{ij} - 3pV \tag{2.58}$$

where the dot donates d/dt and $\vec{F}_{ij}$ represents the two-body force, that acts on particles i and j.

2.5.4 Constraint Method

Controlled Temperature

An alternative approach to control the temperature is by introducing mechanical constraints into Newton's equations of motion to adjust the kinetic energy and therefore the temperature of the simulation system. Hence the constraint equation reads:

$$\frac{1}{2}m\sum_{i=1}^{N_a}\left(\dot{\vec{x}}_i\right)^2 = N_a E_k \tag{2.59}$$

The constrained equations of motion are:

$$\ddot{\vec{x}}_i = \frac{\vec{F}_i}{m} + \alpha\dot{\vec{x}}_i \tag{2.60}$$

with a friction-like term and the Lagrangian multiplier α. Since $\dot{E}_k = 0$ respectively $\sum_i \dot{\vec{x}}_i\cdot\ddot{\vec{x}}_i = 0$, the value of α yields:

$$\alpha = -\frac{\sum_i \dot{\vec{x}}_i\cdot\vec{F}_i}{m\sum_i\left(\dot{\vec{x}}_i\right)^2} \tag{2.61}$$

Controlled Pressure and Temperature

The constraint method allows to control the pressure of a simulation system as well. The unconstrained Lagrangian, formulated in scaled coordinates is:

$$\mathcal{L} = \frac{1}{2}mV^{2/3}\sum_i\left(\dot{\vec{x}}_i\right)^2 - \sum_{i<j}U(V^{1/3}\vec{r}_{ij}) \tag{2.62}$$

The constraints equations for pressure and temperature are:

$$\frac{1}{2}mV^{2/3}\sum_i\left(\dot{\vec{x}}_i\right)^2 = NE_k \tag{2.63}$$

$$mV^{2/3}\sum_i\left(\dot{\vec{x}}_i\right)^2 + V^{1/3}\sum_{i<j}\vec{r}_{ij}\cdot\vec{F}_{ij} = 3pV \tag{2.64}$$

The equation of motion takes the form

$$\ddot{\vec{x}}_i = \frac{\vec{F}_i}{mV^{1/3}} + (\alpha' - 2\gamma)\dot{\vec{x}}_i \tag{2.65}$$

with the dilation rate $\gamma := \dot{V}/3V$ and the Lagrange multiplier α'.
With the constant-temperature condition $\dot{E}_k = 0$ it follows that

$$\sum_i \dot{\vec{x}}_i \cdot \ddot{\vec{x}}_i + \gamma \sum_i \left(\dot{\vec{x}}_i\right)^2 = 0 \tag{2.66}$$

and

$$\alpha := \alpha' - \gamma = -\frac{\sum_i \dot{\vec{x}}_i \cdot \vec{F}_i}{mV^{1/3}\sum_i \left(\dot{\vec{x}}_i\right)^2} \tag{2.67}$$

With the constant-pressure condition it is possible to compute γ:

$$\frac{d}{dt}(pV) = p\dot{V} = 3\gamma pV = \frac{1}{3}\sum_{i<j}\frac{d}{dt}\left(\vec{r}_{ij}\cdot\vec{F}_{ij}\right) \tag{2.68}$$

For pair potentials, that depend only on the distance $\vec{r}_{ij}$, one has $\vec{r}\cdot\vec{F} = -r\frac{dU}{dr}$ and it follows:

$$\frac{d}{dt}(\vec{r}\cdot\vec{F}) = -\psi\,\vec{r}\cdot\dot{\vec{x}}_i \tag{2.69}$$

with

$$\psi := \frac{d^2U}{dr^2} + \frac{1}{r}\frac{dU}{dr} \tag{2.70}$$

Thus

$$\gamma = -\frac{V^{2/3}\sum_{i<j}\psi_{ij}\vec{r}_{ij}\cdot\dot{\vec{r}}_{ij}}{9pV + V^{2/3}\sum_{i<j}\psi_{ij}(\vec{r}_{ij})^2} \tag{2.71}$$

with $\psi_{ij} = \psi(r_{ij})$. In addition to solving the equations of motion, the (numerical) solution of the dilation equation

$$\dot{V} = 3\gamma V \tag{2.72}$$

has to be found as well.

2.6 Measurements

Typically observables are measured after the simulation system has reached an equilibrium state. They can be measured either periodically for certain time-steps or by averaging over time. For the second case the expectation value for a quantity O reads:

$$\langle O\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^T dt\, O(t) \tag{2.73}$$

Normally observables depend on positions and velocities of the particle in the simulation system as well as on the time:

$$O^n = O(\vec{x}_1^{\,n}, \dots, \vec{x}_N^{\,n},\ \vec{v}_1^{\,n}, \dots, \vec{v}_N^{\,n},\ t) \tag{2.74}$$

2.6.1 Energy

The *kinetic energy* of a simulation system can easily be calculated:

$$T_{\text{kin}}^n = \sum_{i=1}^{N} \frac{(\vec{p}_i^{\,n})^2}{2m_i} = \frac{1}{2} \sum_{i=1}^{N} m_i\, (\vec{v}_i^{\,n})^2 \tag{2.75}$$

To evaluate the *potential energy*, single and many-body forces have to be taken into account. In case of simple pair-potentials this yields:

$$U^n = \sum_{i=1}^{N} U_{\text{single}}(\vec{x}_i^{\,n}, \vec{v}_i^{\,n}) + \sum_{i<j} U_{\text{pair}}(\vec{x}_i^{\,n}, \vec{x}_j^{\,n}) \tag{2.76}$$

The *total energy* of a system

$$E^n = T_{\text{kin}}^n + U^n \tag{2.77}$$

should be constant over time. However, due to numerical and rounding errors it typically fluctuates around the total initialization energy of the simulation system.

2.6.2 Temperature

With the equipartition theorem, the *temperature* of a simulation system in equilibrium state can be derived from:

$$T_{\text{kin}}^n = \frac{1}{2} f N k_B T^n \tag{2.78}$$

where f represents the degrees of freedom, N the number of particles in the system and k_B the Boltzmann constant. For the total temperature in a system this yields:

$$T^n = \sum_{i=1}^{N} \frac{m_i\, (\vec{v}_i^{\,n})^2}{f k_B} \tag{2.79}$$

In the non-equilibrium state, complex thermodynamic quantities like temperature lack signification have to be taken into account as well.

2.6.3 Pressure

The *virial pressure* is commonly used to obtain the pressure of MD simulations.

$$p = \frac{Nk_BT}{V} - \frac{1}{d\,V}\left\langle \sum_{i<j} \vec{x}_{ij} \cdot \vec{F}_{ij} \right\rangle \tag{2.80}$$

where p represents the pressure, N the number of particles in the system, k_B the Boltzmann constant, T the temperature, V the volume and d the dimension of the problem. $\vec{x}_{ij}$ is the position vector from particle i to particle j and $\vec{F}_{ij}$ is the force acting on particle i caused by particle j. The first term in (2.80) refers to an ideal gas contribution, the second to the virial. $\langle \ldots \rangle$ denotes a time average along the system's trajectory.
For an ideal gas ($\vec{F}_{ij} \equiv \vec{0}$), equation (2.80) reduces to:

$$p = \frac{Nk_BT}{V} \tag{2.81}$$

2.7 Reduced Units

Dimensionless variables (or reduced units) are often used in simulations to express quantities like energy, density, pressure or temperature. The idea behind is to choose convenient units for energy, length and mass and express all other quantities with them.
For a system that uses a Lennard-Jones interaction potential typically the unit of energy is expressed in terms of ϵ, the unit of length in terms of σ and the unit of mass in m (the mass of atoms in the system). All other units can be obtained by combinations of them. An overview of important reduced quantities and their transformation rules is shown in table 2.1. Those rules make a transformation from reduced units back to real ones (and vice versa), relatively easy.

The most important reason to introduce reduced units results from the *law of corresponding states*: There are infinite combinations of ρ^*, T^*, ϵ^* and σ^*, that correlate to the same state in dimensionless variables. For instance, a simulation of a system (that uses the Lennard-Jones interaction potential) with $\rho^* = 0.5$ and $T^* = 0.5$ corresponds to a system of Ar at a temperature of $60K$ with a density of $840kg/m^3$ and a system of Xe at a temperature of $112K$ with a density of $1617kg/m^3$ [14].
Another reason to use reduced units is to avoid critical roundoff errors. When using real physical units (e.g. SI), one has to deal with quantities with values of very different sizes. Hence, rounding errors can arise due to several floating-point multiplications. This problem can be avoided by using reduced units, since all quantities of interest are normally of the same order. Therefore dimensionless variables can also be appropriated for a control mechanism, because a huge or a small variable always indicates a problem in the simulation.

name	symbol	definition
dimensionless distance	r^*	r/σ
dimensionless energy	E^*	E/ϵ
dimensionless temperature	T^*	k_BT/ϵ
dimensionless number density	ρ^*	ρ/σ^3
dimensionless internal energy	U^*	U/ϵ
dimensionless time	t^*	$t/\left(\sigma(M/\epsilon)^{1/2}\right)$
dimensionless velocity	v^*	$v/(\epsilon/M)^{1/2}$
dimensionless force	F^*	$F\sigma/\epsilon$
dimensionless pressure	p^*	$p\sigma^3/\epsilon$
dimensionless self diffusion coefficient	D^*	$D/\left(\sigma/\epsilon/M)^{1/2}\right)$

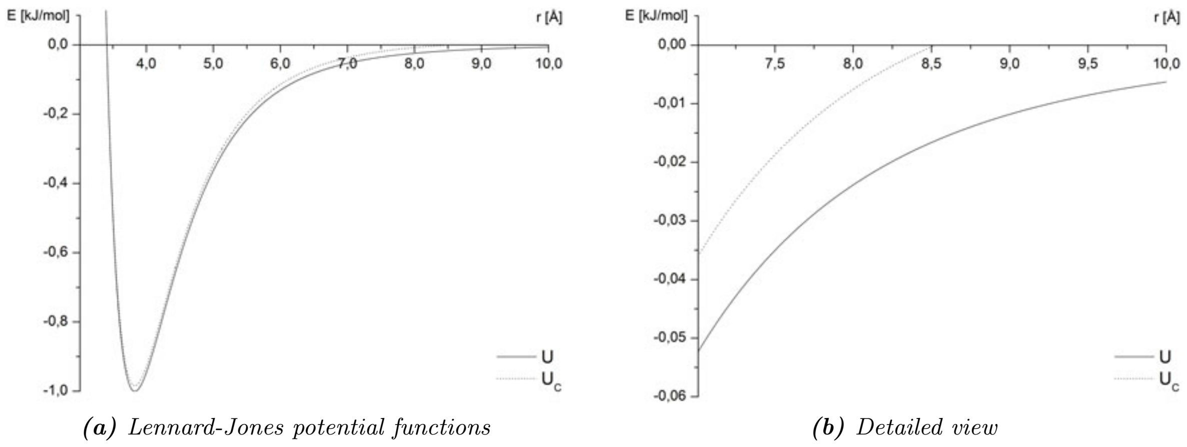
Table 2.1: *Important dimensionless variables: Reduced units are donated with superscript* *.

2.8 Advanced Simulation Methods

The most constraining limitation of MD simulations is the computational effort to study microscopic effects over appropriate time scales. Current studies investigate complex systems with up to 10^7 or even more particles. For such models, reaching meaningful simulation times (typically between 10^{-9} and 10^{-6} seconds) is a serious issue due to the time-step limitation (usually less than 10^{-15} seconds), used in the integration of Newton's equation of motion.
Although the computing power improved exponentially during the last years, a classical MD algorithm is not roughly powerful enough to handle such problems. To deal with these complex simulations, advanced simulations methods like the *Linked-Cell-Method* and *parallelized algorithms* were developed. In this chapter the major methods are described.
In addition to those methods, there are some much more complex methods of performance improvement regarding processor characteristics. Without going into details, some of them should be mentioned: *primary* and *secondary cache*, *address-space-mapping* and *memory interleaving*.

2.8.1 Cutoff Radius

In MD simulations, forces acting on every single particle are evaluated at each time-step. This is the most time consuming part of the simulation process. To speed up this step, it is reasonable to include only particles, that make a contribution to the force-computation. For example, it would be pointless to include all particles in the force calculation of a potential, which acts only on the nearest neighbor particles. Similar considerations apply to fast decreasing potential like the Lennard-Jones potential (see section 2.4). Here, fast decreasing means that the potential drops faster than $\frac{1}{r^d}$, where d describes the dimension of the problem. In such a case a *cutoff radius* r_{cut} can be introduced, after that the potential vanishes. With this approximation, only



(a) *Lennard-Jones potential functions* ***(b)*** *Detailed view*

Figure 2.4: *Lennard-Jones potential functions with and without smooth cutoff for spherical-symmetrical Argon ($\epsilon = 1.00$ kJ/mol and $\sigma = 3.41$ Å): a) shows the complete attractive part of the potentials, while b) shows a detailed view in the range of r_c. The solid line illustrates the original Lennard-Jones potential function, while the dotted line indicates the potential with cutoff at $r_c = 2.5\sigma = 8.525$ Å.*

particles with $r_{ij} \leq r_{cut}$ are considered in the force evaluation. Typically $r_{cut} = 2.5\sigma$ is chosen.

Methods that implement this cutoff radius need to ensure that energies and resulting forces are still continuous functions of the distance r_{ij}. This is the case for:

$$U(\vec{r}) \rightarrow U_c(\vec{r}) = \begin{cases} U(\vec{r}) - U\left(\frac{r_c}{r}\vec{r}\right) - \nabla U\Big|_{\frac{r_c}{r}\vec{r}} \cdot \left(1 - \frac{r_c}{r}\right)\vec{r} & \text{for} \quad r < r_c \\ 0 & \text{for} \quad r \geq r_c \end{cases} \tag{2.82}$$

Figure 2.4 shows the original Lennard-Jones potential energy function as well as the cut one. For spherical symmetric potential energy functions (as for the Lennard-Jones one) the evaluation of this expression is simplified by the fact that $U(\frac{r_c}{r}\vec{r})$ and $\nabla U|_{\frac{r_c}{r}\vec{r}}$ do not depend on $\vec{r}$, but have fixed values for a given r_{cut}. Differentiating (2.82) yields the foce (U beeing the Lennard-Jones potential from equation (2.4)):

$$\vec{F}_{ij} = \begin{cases} \frac{48\epsilon}{\sigma^2}\left[\left(\frac{\sigma}{r_{ij}}\right)^{14} - \left(\frac{\sigma}{r_c}\right)^{14} - \frac{1}{2}\left(\frac{\sigma}{r_{ij}}\right)^{8} + \frac{1}{2}\left(\frac{\sigma}{r_c}\right)^{8}\right]\vec{r}_{ij} & \text{for} \quad r < r_c \\ \vec{0} & \text{for} \quad r \geq r_c \end{cases} \tag{2.83}$$

Due to this simplification forces are not calculated accurately and the total energy of the simulation system is slightly changed. This calculation error can be disregarded, if r_{cut} is chosen to be great enough.
Introducing a cutoff radius decreases the force computation complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, if the particles are uniformly distributed in the simulation space.

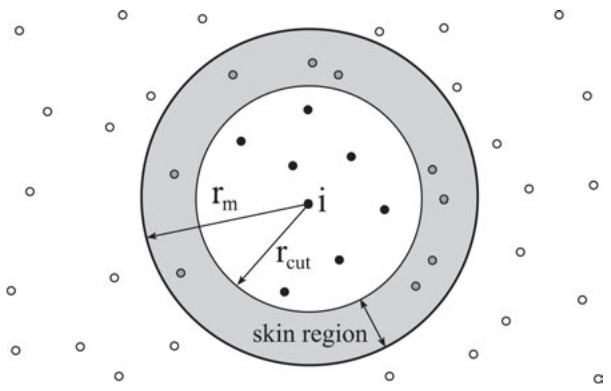


Figure 2.5: *Schema of the Verlet-Neighbor-List-Method: For a particle i, a Verlet-list is created that includes all particles which are within a certain distance r_m. The, so-called, skin region represents a reservoir of particles, which has to be updated repeatedly. If in successive time-steps particles from outside r_{cut} enter the interaction space, they originate from the skin region and are therefore stored in the list as well.*

2.8.2 Verlet-Neighbor-List-Method and Linked-Cell-Method

The first step towards performance increase is to minimize the number of computations that have to be performed. Most methods for non-bonded force calculations can be classified into two categories: the *Verlet-Neighbor-List-Method* and the *Linked-Cell-Method* (or a combination of both). Those methods are common approaches to evaluate approximated, fast decreasing potentials like (2.83).

In the *Verlet-Neighbor-List-Method* [28] for every particle a Verlet-list is generated after a certain number N_m of time-steps. This list includes all possible particles that are close enough to interact with that particle during the time period $N_m \delta t$. These are all atoms within a certain distance r_m. r_m and N_m are chosen such that

$$r_m = r_{cut} + N_m \overline{v} \delta t \tag{2.84}$$

where $\overline{v}$ is a typical atom velocity within this simulation (Fig. 2.5). The updating process of the Verlet-Neighbor-List is still an operation with a complexity of $\mathcal{O}(N^2)$. However, this approach increases the performance of the algorithm, since it is only done every N_m time-steps. The performance can still be enhanced by using the Linked-Cell-Method to update the Verlet-Neighbor-Lists.

The idea behind the *Linked-Cell-Method* [47] is to separate the simulation space into uniform cells with a side length not less than r_{cut}. Because of the cut potential, the interaction between particles is limited to particles in the same cell or a neighbor cell (8 cells in two dimensions, 26 cells in three dimension) as shown in figure 2.6 for a two-dimensional problem.

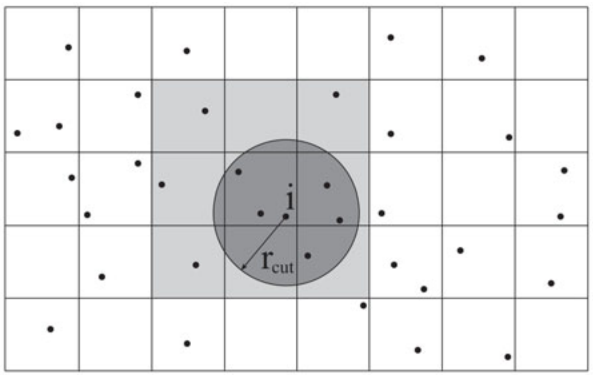


Figure 2.6: *Linked-Cell-Method: The simulation space is separated into quadratic cells with a size of $r_{cut} \cdot r_{cut}$. The dark hatched circle symbolizes the interaction radius r_{cut} of particle i. The light hatched area indicates all effected cells.*

The force, acting on a particle i in cell ic yields then:

$$\vec{F}_i \approx \sum_{\substack{kc \\ kc \,\epsilon\, \mathcal{N}(ic)}} \quad \sum_{\substack{j \\ j \,\epsilon\, \{particle\ of\ cell\ kc\} \\ i \neq j}} \vec{F}_{ij} \tag{2.85}$$

where $\mathcal{N}(ic)$ represents cell ic as well as all its neighbor cells.
This method decreases the force-summation complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. It is even possible to improve this result by using a modified Linked-Cell-Method [49].
In order to implement this algorithm in an effective way, abstract data structures, so-called *single-linked list structures* are used.

To verify the Linked-Cell-Method, a series of fourteen benchmarks, using the modelMD software package (see section 3.1), were performed. The simulations consisted of 50 to 3200 gas particles, which were uniformly distributed in boxes with sizes between 6.30 Å x 6.30 Å x 6.30 Å and 31.75 Å x 31.75 Å x 31.75 Å, in order to keep a constant relative particle density of 0.1. All simulations were performed for 10^4 time-steps with a single time-step length of $\Delta t = 0.0005$. The total runtime was measured for simulations using the basic algorithm and for simulations using the Linked-Cell-Method. The results are shown in table 2.2 and figure 2.7. It can be clearly seen that computation times for classical simulations grow proportional to N^2, while computation times for the advanced algorithm (using the Linked-Cell-Method), grow proportional to N.

2.8.3 Parallel-Computing-Methods

Beyond the use of cells and linked neighbor lists, another method to increase the computational efficiency of a simulation software is parallel computing. Here a large problem is broken into discrete parts and spread over several processors, which carry out the computation simultaneously.

particles	t_{sim} [s]	t_{sim_lc} [s]
50	8	8
100	24	15
200	87	31
400	332	63
800	1308	114
1600	5143	227
3200	21608	492

Table 2.2: *Benchmark results of the Linked-Cell-Method: t_{sim} represents the total simulation runtime of the basic algorithm while t_{sim_lc} is the runtime for the Linked-Cell Method.*

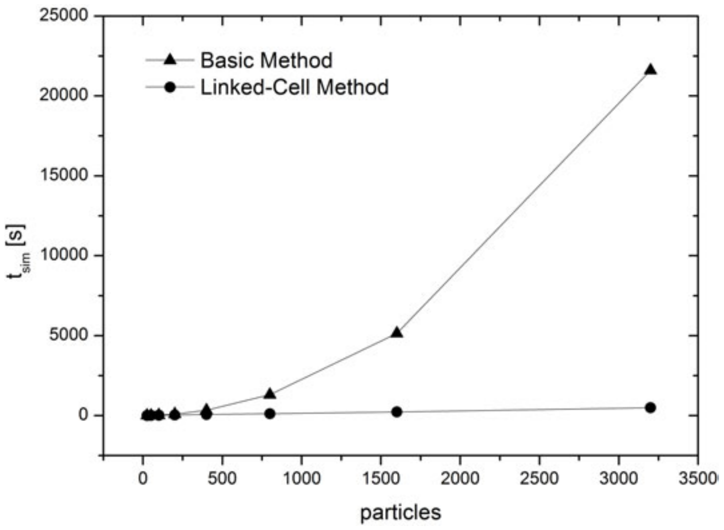


Figure 2.7: *Benchmark results of the Linked-Cell-Method: The total simulation runtime (for 10^4 time-steps) depending on the number of particles in the simulation is illustrated.*

There are different kinds of parallel computer architectures, which are classified by whether they are operating using a single set or multiple sets of instructions (instruction stream), whether or not those instructions are using single or multiple sets of data (data stream), as proposed by Flynn [22] in 1966. The four basic types are: *SISD* (Single Instruction stream, Single Data stream - classical microprocessor), *SIMD* (Single Instruction stream, Multiple Data stream - vector processors), *MISD* (Multiple Instruction stream, Single Data stream) and *MIMD* (Multiple Instruction stream, Multiple Data stream).
Nowadays most parallel computer systems are of the *MIMD* type, wherefore this architecture is described here in more detail.

A MIMD machine uses a number of processors, that execute independently and asynchronously different instructions with different data. They can be categorized as multiple processor machines using a *distributed memory* or a *shared memory* architecture.

In *distributed memory* machines (Fig. 2.8) every processor has its private memory, which can not be accessed by another processor. In order to perform computations, data has to be exchanged between these private memories over a network - the so-called message passing approach. A parallelized programm on a distributed memory machine doesn't consist only of a sequence of computational tasks, like a sequential one, but also has to perform communication tasks at some points. Since this tasks can't be carried out automatically the parallelization of a sequential program for distributed memory machines is much more complicated than on shared memory machines.
A *shared memory* machine (Fig. 2.9) uses a big, global memory, which can be accessed by

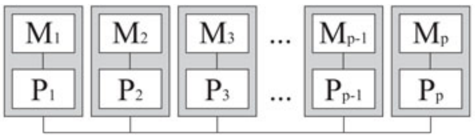


Figure 2.8: *Schema of a Distributed-Memory-Machine*

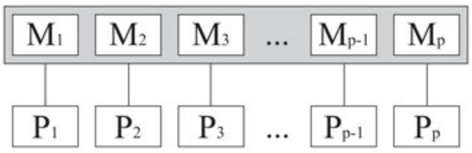


Figure 2.9: *Schema of a Shared-Memory-Machine*

all processors. This global memory can be designed as one big memory block, or smaller distributed memory blocks, that are allocate as a virtual one with a global address space.
However, all data is stored in this shared memory, only computations are distributed among the processors. For an efficient performance it is essential that those tasks are uniformly spread and that they don't interfere with each other (for example by trying to read/write the same data at the same time).
Typically shared memory supercomputers are equipped with 16 to 64 processors of the same type. Greater processor numbers are not suitable, since the performance of the system is limited by the bandwidth of the memory system. Hence a performance decrease is noticeable at a certain hardware-depending number of processors.

Parallelization-Strategies for Molecular Dynamics Programs

Parallelization methods for sequential MD programs heavily depend on the parallel computer architecture the program should work on.

Processors of *distributed memory* machines have to communicate with each other in order to exchange their data. The *Replicated-Data-Method* would be a naive parallelization strategy for this task on these machines. Every processor receives a copy of all data that is needed in the simulation, but works only on a small, allocated data field. After every computation step, the data has to be synchronized among all processors. This yields in a huge communication complexity, since all data has to be exchanged, which also includes data not needed by a processor for its calculations. If the particles are uniformly distributed, this method has a computation complexity of $\mathcal{O}(N/P)$, where N is the number of particles in the simulation and P represents to number of processors used. However, the communication and memory complexity yields $\mathcal{O}(N)$ and dominates the overall performance with an increasing number of processor.
An adequate solution would be the *Data-Decomposition-Method.* Here a processor only receives data needed for its calculations, so N/P particles (that can be, for example, chosen by their particle-number) and their interacting neighbor particles (which can be easily determined by

Verlet-Linked-Lists) have to be exchanged. At every communication step, a processor has to receive and send maximally $\mathcal{O}(N/P)$ particles. This results in a communication and memory complexity of $\mathcal{O}(N/P)$. In contrary to the Replicated-Data-Method, the communication and memory complexity decreases with an increasing number of processors.
Another advanced parallelization-strategy is the *Domain-Decomposition-Method.* The particles are shared among the processors in a way, that only little communication complexity is necessary. This is achieved by splitting the simulation area into small sections, which are assigned to the processors. If a particles leaves such a domain, it is allocated to another processor, which makes only communication between those two processors necessary. At equipartition, every processor receives $\mathcal{O}(N/P)$ particles. Since all these particles belong to the same domain, the processors already possess almost all data they need for their calculations. Missing particles belong to neighbor sections, so only their data has to be exchanged at every communication step. The complexity of data that has to be received decreases to $\mathcal{O}(\sqrt{N/P})$ for a two dimensional and $\mathcal{O}(N/P^{2/3})$ for a three-dimensional problem. If the particles are not uniformly distributed this method can be improved by dynamical splitting of the simulation area.
During the last years, the *Message Passing Interface (MPI)* standard [1] established itself for parallelization on distributed memory machines [8, 36].

The *shared memory architecture* makes a parallelization of a sequential program comparatively easy. As all data is stored globally, there is no need for changing the data structure of the sequential code or data traffic between processors. Parallelization can be done with only a few changes to the original code, which basically is still sequential. Code sections that should be parallelized are encapsulated in special blocks. If special parallelization instructions are necessary in addition to parallelization blocks, they are masked by compiler directives (eg. $OMP directives in Fortran).
Parallelization of MD algorithms on shared memory machines have been extensively studied (e.g. [7, 35]), therefore only the basic idea is represented here.
At MD programs it is reasonable to parallelize the most CPU-time-consuming parts of the algorithm, namely the force calculation, the neighbor-list update and the boundary conditions evaluation. This method has several advantages compared to full parallelization, for example the easy implementation.
Loops are particularly suitable for parallelization on shared memory machines. If the linked-cell-method is used, for instance, the force calculation for all particles is performed by iterating over all particles in the current cell and its neighbor cells. These forces are stored and used to integrate the equations of motion. Such calculations can be easily parallelized by sharing the cells among all processors, which compute the forces acting on the contained particles. Similar techniques are used to parallelize the neighbor-list update and the boundary conditions evaluation.
There are some standards and utilities, that provide parallelization and data synchronization on many different platforms, like the *OpenMP* (Open Multi-Processing) application program-

processors	t_{sim} [s]
1	1945
2	1794
3	1453
4	1218
5	1090
6	1011
7	906
8	826
9	745
10	717
11	712
12	698

Table 2.3: *Benchmark results of the parallelized MD program package modelMD: t_{sim} represents the total simulation runtime of the algorithm for 10^3 time-steps.*

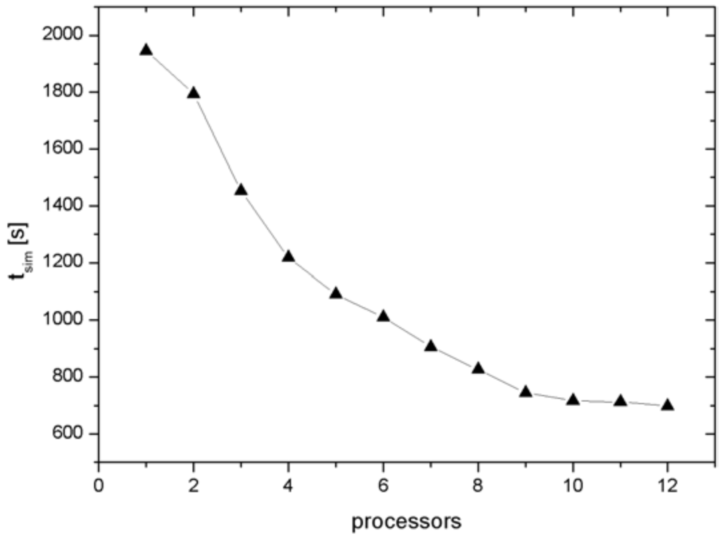


Figure 2.10: *Benchmark results of the parallelized MD program package modelMD: The total simulation runtime (for 10^3 time-steps) depending on the number of processors used for simulation is illustrated.*

ming interface [2], which is also used in the modelMD software package.

To verify this parallelization strategy a series of benchmarks were performed, using modelMD with openMP. The simulations consisted of 85.000 gas particles, which were uniformly distributed in a box with a size of 75 Å x 75 Å x 75 Å. All simulations were computed for 10^3 time-steps with a single time-step size of $\Delta t = 0.0005$. The simulations were performed on a shared memory machine using between 1 (no parallelization) and 12 processors. The total runtime t_{sim} was measured. The results are shown in table 2.3 and figure 2.10.

Chapter 3

Software Packages modelMD and simEdit

A main part of this thesis was the development of a high-performance classical MD simulation software package in order to understand the important processes, routines and algorithms that are used in a MD simulation tool. During an earlier project at the Technical University of Graz, a basic classical MD simulator *modelMD*, including the graphical-user-interface *simEdit* was developed [26]. These software packages were redesigned and further improved to meet the requirements of complex MD simulations with up to 10^6 particles. All simulations in this work were performed with the modelMD software suite. Subsequently a brief, introductive description for both program packages is presented.

3.1 modelMD

3.1.1 Introduction

modelMD is a classical MD simulation software package, designed to perform on parallel as well as on sequential computers. The code was implemented in *FORTRAN95* and parallelized with the *openMP* standard [2] for machines that use a shared-memory architecture.
To integrate Newton's equations of motion the Leap-Frog (section 2.4.1), the Runge-Kutta (section 2.4.2) and the Predictor-Corrector (section 2.4.3) algorithms were implemented.
The Linked-Cell-Method with Verlet-Neighbor-Lists (section 2.8.2) was realized and parallelized for high-performance computations.
Further features of modelMD include the possibility to run simulations in two or three dimensions, reduced (see section 2.7 as well as real physical units, several boundary-conditions, an astrophysical mode and a basic thermostat implementation. The Lennard-Jones potential (2.3.1) was realized as interacting potential.

3.1.2 Input

The required simulation data for modelMD is allocated in a class of input files: *simulation.inp*, *posvel.inp*, *extforce.inp* and *thermostat.inp*. Those files can be manually created or automatically by using the simEdit GUI.
The *simulation.inp* file contains the basic simulation parameters. A detailed format description is listed in table 3.1. A sample input file is shown in figure 3.1.

parameter	description
dimension	Dimension of the problem. Simulations can be run in two or three dimensions.
units	Units that are used for the simulation. Either reduced units (value "*reduced*") or physical units (value "*real*") can be chosen. In case of real physical units, energies are calculated in electron Volt (eV), times in picoseconds (10^{-12} seconds), temperatures in Kelvin and the unit of length is Ångström (10^{-10} meters).
spatrange	Size of the simulation space in each dimension. Coordinates are centered at the origin, therefore *spatrange* $= a$ permits $x_i \in (-\frac{a}{2}, \frac{a}{2}]$.
leftbound rightbound	Boundary conditions of the simulation space, where *leftbound* defines the conditions at $x_i < 0$ and *rightbound* the conditions at $x_i > 0$. Possible values are: "*p*" periodic boundary conditions "*r*" hard reflecting wall "*d*" diffuse reflecting wall "*f*" hard reflecting wall, where the perpendicular velocity-vector-component is multiplied with a constant factor "*w*" hard wall with an applied Lennard-Jones potential "*h*" heat-bath with fixed temperature "*l*" heat-bath, that is linearly increased by time "*e*" heat-bath, that is exponentially increased by time "*c*" delete particles from the simulation

Table 3.1: *File format description for a simulation.inp input file*

name	description
lwallpara rwallpara	Additional parameters for boundary conditions. Depending on the parameters chosen for *leftbound* and *rightbound*, none, one or two supplementary parameters have to be set: "*f*" constant multiplication factor "*w*" Lennard-Jones parameters ϵ and σ "*h*" initial temperature "*l*" initial temperature and linear change per time-step "*e*" initial temperature and exponential change per time-step For all other values *leftbound* and *rightbound* no additional parameters have to be set.
cutoff	Cutoff radius (r_{cut}) of the Lennard-Jones potential, valid for all interactions in the simulation.
simtimetot	Total simulation time (t_{total}).
timestep	Length of a single time-step (Δt).
integrator	Algorithm that should be used to integrate Newton's equations of motion (see section 2.4). Possible parameters are: "*lf*" Leap-Frog-Method "*rk*" Runge-Kutta-Method "*pr*" Predictor-Corrector-Method
statwrite	Time intervals between specific data output. Three parameters have to be set: *observables* (statistics.out), *positions* (trajectories.out), *complete system state* (snapshots.out).
parttypes	Number of particle types that are used in the simulation.
partnames	Names of particle types that are used in the simulation.
masses	Masses of particle types that are used in the simulation.
epsilon	Lennard-Jones potential parameter ϵ for all particle types that are used in the simulation.
sigma	Lennard-Jones potential parameter σ for all particle types that are used in the simulation.
charges	Electric charges of particle types used in the simulation.

Table 3.1: *File format description for a simulation.inp input file*

name	description
mbform mbpart	These two parameters are reserved for the many-body part of modelMD, which has not been implemented yet.
posvel	Position-Velocity mode. Possible values are: "*data*" Positions and velocities are read from data file posvel.inp. "*rand*" Positions and velocities are generated randomly, using an initialization temperature set in *termtemp*.
partdistr	Number of particles for each particle type.
thermtemp	Initialization temperature of particle types that are used in the simulation. This parameter is only required, if "*rand*" is chosen for the parameter *posvel*.
gravaccel	Global, external gravity force.
elecfield	Global, external electric field vector.
magnfield	Global, external magnetic field vector.
forcelist	Additionally to global external forces, it is possible to include special forces, that act only on some particle groups. Possible values are: "*yes*" Additional forces are used and required parameters are read from the input file extforce.inp. "*no*" No external forces are used.
thermostat	Indicates if thermostats are used in the simulation. Possible values are: "*yes*" Thermostats are used. Required parameters are read from the input file thermostat.inp. "*no*" No thermostats are used.
astrograv	Astrophysics gravitation constant. If *astrograv* $\neq 0$, the astrophysical mode (an intermolecular Newton gravitation potential) is activated.
threadnum	Number of threads that should be used for the simulation run at parallelized regions.
dynamic threads	Parameter that indicates if the number of threads in parallel regions can be adjusted automatically during the simulation runtime. Possible values are: "*yes*" Dynamical adjustment is used. "*no*" Dynamical adjustment is not used.

Table 3.1: *Format description for an simulation.inp file*

```
ModMD input file created with simedit.m on 26-Jan-2010 at 18:22
-----
i) Space and boundary conditions:
DIMENSION,      2
UNITS,          real
SPATRANGE,      1000.00, 500.00
LEFTBOUND,      p, r
RIGHTBOUND,     p, r
1LWALLPARA1,    0, 0
2RWALLPARA1,    0, 0
3LWALLPARA2,    0, 0
4RWALLPARA2,    0, 0
CUTOFF,         8.6000
-----
ii) Time and integration:
SIMTIMETOT,     100000.0000
TIMESTEP,       0.2000
INTEGRATOR,     lf
STATWRITE,      5000, 100, 5000
-----
iii) Particle parameters:
PARTTYPES,      2
PARTNAMES,      heavy, light
MASSES,         40, 20
EPSILON,        0.011, 0.011
SIGMA,          2.8, 3.4
CHARGES,        0, 0
MBFORM,         0, 0
MBPART,         0, 0
POSVEL,         data
PARTDISTR,      32130, 21462
THERMTEMP,      150, 150
-----
iv) External forces:
GRAVACCEL,      0, -0.02
ELECFIELD,      0, 0
MAGNFIELD,      0, 0
FORCELIST,      yes
THERMOSTAT,     no
ASTROGRAV,      0
-----
v) Parallel Processing:
THREADNUM,      8
DYNAMICTHREADS, no
```

Figure 3.1: *Sample simulation.inp file*

column	name	description
1	particle type	Indicates the particle type, to which the particle belongs.
2	activity index	The activity index signalizes if the particle is active or will be ignored in the simulation. Possible values are: "*1*" the particle is active "*0*" the particle will be ignored
3	x_i	initial position vector
4	$\dot{x}_i$	initial velocity vector

Table 3.2: *File format description for a posvel.inp file*

The *posvel.inp* file contains initial positions and velocities for all particles in the simulation. The first line corresponds to the number of particles in the simulation. Each following line represents one particle, which parameters are listed in columns that are separated by whitespace. A detailed format description is listed in table 3.2.

It is possible to include special, external forces, that act only on some particle groups. Those forces defined in the *extforce.inp* file. Each line represents one group, whose parameters are listed in columns that are separated by commas. Since all particles are numbered, a group can be defined by specifying a first and last particle. All particles with a number in between are then included in this group. A detailed format description is listed in table 3.3. The start and end particle have to be set to "0" in the last line to signalize the end of the file.

Thermostats can be included in a simulation as well. They are defined in the *thermostat.inp* file, which formation is quite similar to the one of the extforce.inp file: Here each line represents one group, which parameters are listed in columns that are separated by commas. A detailed format description is shown in table 3.4. Again, the start and end particle have to be set to "0" in the last line to signalize the end of the file.

3.1.3 Output

modelMD creates several output files, that contain trajectories and end positions as well as other observables like kinetic and potential energies or temperatures.
Particle trajectories are stored in the *trajectories.xyz* file, using the XYZ-file-format. All positions are saved repeatedly at certain time-steps (which are set in the simulation.inp file). This data can be easily analyzed and visualized with tools like VMD [48].
Position files like *positions_temp.xyz* and *positions_end.xyz* are similar formatted like the trajectories.xyz file, though they only contain particle positions of a single time-step. The positions_temp.xyz file is updated repeatedly at certain time-steps. Therefore it can be used to

column	name	description
1	N_{start}	Number of the first particle of the group
2	N_{end}	Number of the last particle of the group
3	$F_{ext_{x_1}}$	x_1 component of an external force vector
4	$F_{ext_{x_2}}$	x_2 component of an external force vector
5	$F_{ext_{x_3}}$	x_3 component of an external force vector
6	γ	Coefficient of friction
7	η	Density (viscosity) of a virtual fluid (see section 2.5.2)
8	T_0	Temperature of an ideal heat-bath, to which the group is connected to
9	dT/dt	Linear temperature growth coefficient (per time-step) of an ideal heat-bath
10	α_{harm}	Harmonic oscillation force. The center of the harmonic oscillating force conforms to the initial position of every particle it is applied to
11	$A_{0_{x_1}}$	x_1 component of the amplitude of an harmonic oscillation force
12	$A_{0_{x_2}}$	x_2 component of the amplitude of an harmonic oscillation force
13	$A_{0_{x_3}}$	x_3 component of the amplitude of an harmonic oscillation force
14	ω	Angular frequency of the harmonic oscillation force

Table 3.3: *File format description for an extforce.inp file*

column	name	description
1	N_{start}	Number of the first particle of the group
2	N_{end}	Number of the last particle of the group
3	T	Designated temperature
4	γ	Dumping factor of the Velocity Scaling thermostat (see section 2.5.1)
5	δN	Number of computation steps between temperature adjustment

Table 3.4: *File format description for a thermostat.inp file*

control simulation results during runtime. By contrast, the positions_end.xyz file is created at the end of the simulation and contains all particle positions at the last time-step.
When a simulation finishes, all final particle positions and velocities, as well as activity indices and particle types are saved in the *posvel.out* file. This file is formatted in the same way as the posvel.inp file, in order to be able to use the data to continue a previous simulation. Therefore an already completed simulation can be continued just by changing the file extension of posvel.out to posvel.inp.
The *statistics.out* file stores frequently energies and temperatures during the simulation runtime.
The complete system state is repeatedly saved to the *snapshots.out* file.

3.1.4 Further Work

The current version of modelMD is already capable to perform complex MD simulations. However, there is a number of interesting features left to be implemented that would improve computation performance and enhance simulation possibilities:

modelMD was parallelized using the openMP standard to run simulations on machines with a shared-memory architecture. Including an implementation of the MPI standard with the Domain-Decomposition approach for the Linked-Cell-Method would make additional advanced parallelization-strategies for distributed-memory machines possible. This would mean a further improvement of the computation performance.

Long-range potentials (as the Coulomb potential or the intermolecular gravitation potential - see section 2.3.1) as well as many-body potentials (as briefly discussed in section 2.3.2) can not be used in modelMD so far. A treatment of this kind of potentials would represent a major feature.

Furthermore, there is only one basic temperature control mechanism, namely the Velocity Scaling Method (see section 2.5.1) implemented yet. Additional temperature and pressure control mechanisms (as described in section 2.5) would mean a clear improvement to modelMD.

3.2 simEdit

3.2.1 Introduction

simEdit is a graphical-user-interface (GUI), designed to create input files for the modelMD packages quickly and easily. The main program was implemented in *MATLAB*, computation-intensive subroutines in *C++*. These functions are dynamically loaded by the MEX (Matlab

Executable) data interface. Since simEdit creates all required input files for modelMD, it uses the same input parameters, which have been already described in detail in section 3.1. Therefore they wont be listed here again. In the following a brief introduction to the user interface is provided.

3.2.2 Graphical User Interface

The GUI (figure 3.2) is arranged into three main domains: A *parameter section* on the left where all simulation parameters can be set, a *preview window* on the right that provides a dynamic preview of the simulation box and a *menu* at the bottom of the window.

To keep track of all input parameters, the parameter section consists of five basic tabs: *Basics*, *Space*, *Particles*, *External Forces* and *Groups*.
As the name implies, in the *Basics* section, primary settings like the simulation name, unit adjustments, time and integration parameters as well as performance and data output options are set.
The dimension of the problem, boundary condition parameters and the size of the simulation space are set in the *Space* tab.
The *Particles* section allows to adjust all particle parameters like name, mass, charge, Lennard-Jones potential parameters and many-body options.
Parameters for global external forces are set in the *External Forces* tab. This includes gravitation

$$\vec{F}_{g\,i} = -m_i \vec{g} \tag{3.1}$$

as well as electric and magnetic forces, that cause Lorentz forces on charged particles

$$\vec{F}_L = q\left(\vec{E}_{\text{ext}}(\vec{x}_i^{\,n}) + \vec{v}_i^{\,n} \times \vec{B}(\vec{x}_i^{\,n})\right) \tag{3.2}$$

The *Groups* section is probably the most powerful and important part of simEdit. Here particle groups (a cluster of particles of the same type) are assembled and initial positions, initial temperatures as well as additional velocities are assigned. According to a chosen initial temperature, velocities with random directions and a magnitude from a Maxwell-Boltzmann velocity distribution

$$T(x, y, z) = T_0 + T_x \cdot (x - x_0) + T_y \cdot (y - y_0) + T_z \cdot (z - z_0) \tag{3.3}$$

are applied. Here T_0, T_x, T_y, T_z, x_0, y_0 and z_0 can be set.
The position of a group can be specified by a combination of right parallelepiped and spheres in three dimensions, respectively rectangles and circles in two-dimensional problems. These geometric bodies are then filled with particles which positions are determined by using a regular lattice, by randomly positioned particles (for a certain density) or by randomly oriented particle

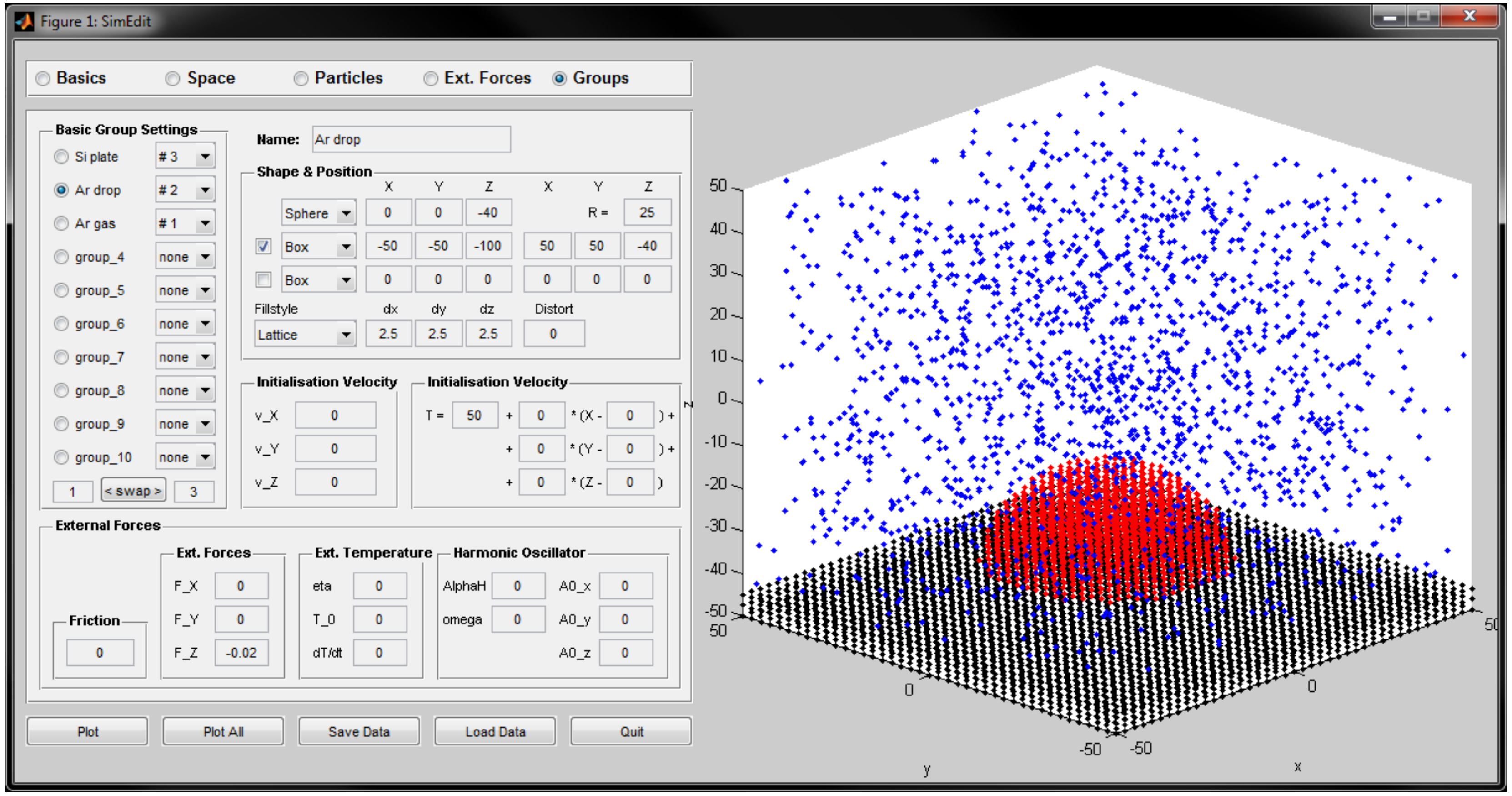


Figure 1: SimEdit

Basics | Space | Particles | Ext. Forces | [✓ Groups]

Basic Group Settings

Si plate	# 3
[✓ Ar drop]	# 2
Ar gas	# 1
group_4	none
group_5	none
group_6	none
group_7	none
group_8	none
group_9	none
group_10	none

1 < swap > 3

Name: Ar drop

Shape & Position

	X	Y	Z	X	Y	Z
Sphere	0	0	-40		R =	25
✓ Box	-50	-50	-100	50	50	-40
Box	0	0	0	0	0	0

Fillstyle	dx	dy	dz	Distort
Lattice	2.5	2.5	2.5	0

Initialisation Velocity

v_X 0
v_Y 0
v_Z 0

Initialisation Velocity

T = 50 + 0 * (X - 0) +
+ 0 * (Y - 0) +
+ 0 * (Z - 0)

External Forces

Friction 0

Ext. Forces

F_X 0
F_Y 0
F_Z -0.02

Ext. Temperature

eta 0
T_0 0
dT/dt 0

Harmonic Oscillator

AlphaH 0 A0_x 0
omega 0 A0_y 0
A0_z 0

Figure 3.2: Screenshot of the Groups section of the SimEdit GUI

chains of a maximum length.
External forces, like harmonic oscillation forces (3.4), friction or connections with ideal heat-bathes, can be allocated to certain particle groups as well.

$$F_{\text{harm}\ i} = -\alpha_i(\vec{x}_i^{\,n} - \vec{x}_i^{\,0})^2 \tag{3.4}$$

If more than one particle group is used in the simulation their order has to be considered as well, since groups that are disposed further down in the group-list have a lower priority (e.g. in figure 3.2 group "Ar drop" has a higher priority as "Ar gas", but a lower one than "Si plate"). Particles of groups with low-priority get deleted, if they are placed too close to particles of groups with higher ones. The minimum distance for particles of different groups can be set with the *minimum distance* parameter at the *particles* tab. On this account, a swap button allows simple exchanging of two groups and accordingly changing the group order.

The menu includes the following functions: *Plot* updates the preview window and creates a two or three-dimensional plot of the current group. All particle groups can be plotted together with the *Plot All* command. *Save Data* creates all input files that is required by modelMD and saves all parameters to a *simulation.mat* file. This data can later be retrieved with the *Load Data* command. *Quit* ends the program.

3.2.3 Output Data

simEdit creates all input files (imulation.inp, posvel.inp, extforce.inp and thermostat.inp), which are required by modelMD. These files and their formation have been already described in detail in section 3.1.2.

3.2.4 Further Work

Although simEdit is already a powerful tool to create all necessary input files for modelMD, there are some important features left to be added:

The particle arrangement is limited to random positioning and positioning on a cubic lattice, in spheres or boxes. Positioning on other grids (hcp, fcc) and shapes would mean a good improvement to the program.

Although all simulation parameters can be adjusted using simEdit, some options were planned and created in the graphical frontend, but not implemented in the program routines yet. These options include parallelization and thermostat parameters.

Chapter 4

Complex Fluid-Dynamical Phenomena

This chapter concentrates on complex fluid dynamical phenomena with focus laid on the Rayleigh-Taylor instability and the Rayleigh-Benard convection. Important background knowledge as well as various publications concerning those topics are summarized and presented. This is complemented by a series of classical MD simulations, carried out with the software package modelMD. The results are evaluated and discussed.
All simulations in this chapter were carried out in two dimensions in order to observe more complex structures. With this restriction various phenomena can be already simulated with approximately 10^4 particles, while in three-dimensional simulations at least 10^7 particles would be necessary. Such complex simulations with approximately 10^6 time-steps would even nowadays require high-end supercomputer power.

4.1 Rayleigh-Taylor Instability

4.1.1 Introduction

The Rayleigh-Taylor (RT) instability is a classical example of a turbulence and was first investigated by Lord Rayleigh [37] in 1882 and later theoretically described by Taylor [20] in 1950. This phenomenon concerns a hydrodynamical instability at an interface between two fluids of different densities and masses, that occurs when the lighter fluid is pushing the heavier one [15]. Related processes develop when shock-waves pass through the interface (see figure 4.1).
There is a great number of certain phenomena associated with the development of the unstable interface, as for instance the formation of bubbles and spikes (including Kelvin-Helmholtz instabilities at their margin), formation of droplets or interaction processes between bubbles (which leads to their fusion and furthermore to turbulent mixing and chaotic behavior of the system).

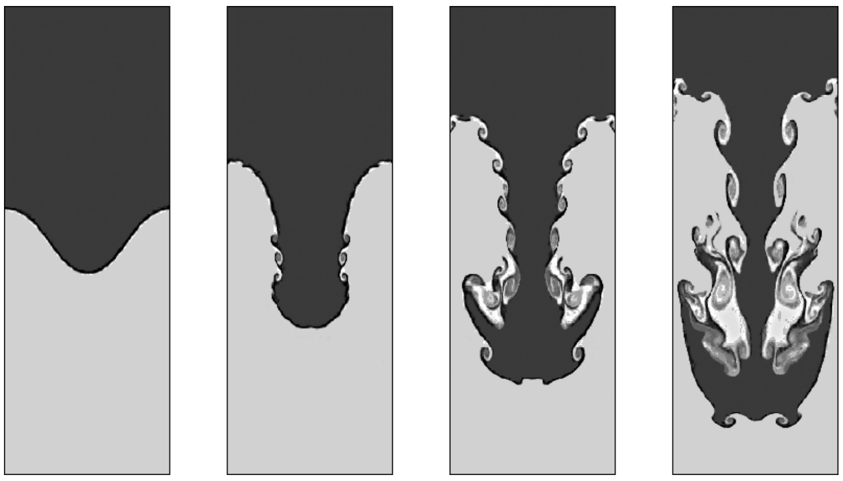


Figure 4.1: *Development of a RT instability with a single wavelength perturbation: Formation of Kelvin-Helmholtz instabilities are shown in the second and later snapshots and the formation of a typical mushroom cap in the third and fourth frame of the sequence.*

The relevance of this instability ranges from various terrestrial phenomena such as weather inversions and geophysical formations like salt domes or volcanic islands to astrophysical phenomena such as supernova explosions.
Most of the previous theoretical work on the RT instability concentrated on single wavelength perturbations [15]. Later, different MD methods were used by Youngs [29], Alda and Dzwinel et al. [45,46] or Kadau et al. [24,25] to describe the phenomenon for many different wavelength modes.
The purpose of this work was to perform MD simulations on the well studied RT instability, as an example of complex hydrodynamic phenomena, to reproduce and combine the results of earlier works on the microscope scale and to verify the modelMD code. The major goal was to perform state-of-the-art MD simulations of the RT instability phenomenon with random initial perturbations (where various different wavelength modes are present), since this occurrence is normally more relevant to practical applications.

To describe the growth process of the instability it is helpful to subdivide it into four distinctive stages [15]. They are discussed in the following for a system, consisting of two infinitely extended inviscid fluids, that meet at an interface (see figure 4.2).

The upper fluid is defined to be the heavier one:

$$m_H > m_L$$

where subscript H hereinafter indicates the heavy fluid at the top and L the light fluid at the bottom. Both fluids are exposed to an effective external force, acting orthogonally to the

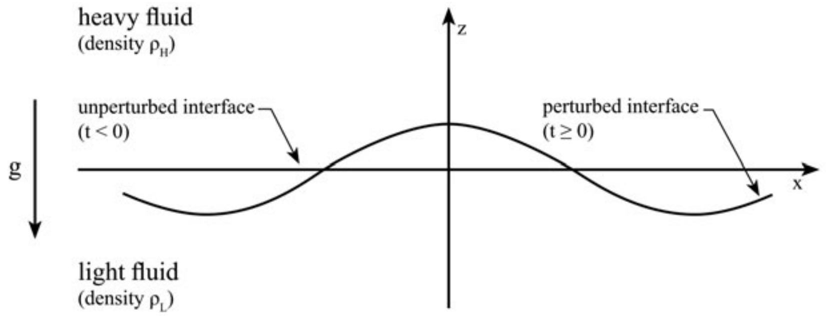


Figure 4.2: *Rayleigh-Taylor Instability: Two infinitely extended, incompressible fluids of different densities meet at an interface. for $t < 0$ the interface is unperturbed an therefore perfectly flat at $z = 0$. For times $t \geqslant 0$, the interface is perturbed.*

interface:

$$\vec{F}_{\text{eff}} = (\vec{a} - \vec{g}) = (a + g)\vec{e}_z = F_{\text{eff}}\vec{e}_z \tag{4.1}$$

where $\vec{a} = a\vec{e}_z$ is an uniform external acceleration applied to the whole system and $\vec{g} = -g\vec{e}_z$ the gravitational acceleration. Here $g > 0$ and $\vec{e}_z$ represents an unit vector orthogonal to the interface, pointing upwards into the heavy fluid.

At times $t < 0$ the fluids are unperturbed, therefore the surface is perfectly flat at $z = 0$ and the system is in a metastable state. At *stage 1* ($t \geqslant 0$) a perturbation P (here a cosine function is chosen for instance) occurs and the system is disturbed.

$$P = \eta(t)\ cos\ kx \tag{4.2}$$

where η is the time depended amplitude, k represents the wave number and x the position.
The time evolution of the perturbation amplitude (4.2) can be determined either by using a potential theory argument via Bernoulli's equation or by an energy analysis. It yields:

$$\ddot{\eta}(t) = \alpha^2(k)\eta(t) \tag{4.3}$$

with

$$\alpha(k) = \left[F_{\text{eff}}\left(\frac{\rho_H - \rho_L}{\rho_H + \rho_L}\right)k - \left(\frac{\sigma}{\rho_H - \rho_L}\right)k^3\right]^{1/2} \tag{4.4}$$

where σ is the coefficient of interfacial tension, ρ_H is the density of the upper fluid and ρ_L the density of the lower one.
The solution of (4.3) for both fluids at rest yields:

$$\eta(t) = \eta(0)\ cosh\ \alpha t \tag{4.5}$$

From this result it follows that the system is stable, as long as α is imaginary. This results in

stable gravity waves at the interface. If α is real, the perturbed interface destabilizes. This is the case if $\sigma = 0$, $F_{\text{eff}} > 0$ and $\rho_H > \rho_L$. In other words, the lighter fluid pushes the heavier fluid and the system is, by definition, not stable anymore.
Furthermore, from (4.5) with α as a real number it follows, that any perturbation grows exponentially with time at this stage.
In *stage 2* the amplitude of the the perturbation continues to grow non-linearly. The behavior of the system is influenced by three-dimensional effects and particularly characterized by the Atwood number, a dimensionless density ratio that is defined as

$$A = \frac{\rho_H - \rho_L}{\rho_H + \rho_L} \tag{4.6}$$

Depending on the Atwood number, two general cases can be recognized: If $A \lesssim 1$, the light fluid penetrates the heavy one and forms round topped bubbles with circular cross sections, while the heavy fluid starts to form spikes and walls between those bubbles.
In case $A \gtrsim 0$, typically both fluids form bubbles while penetrating each other.
In *stage 3* one encounters interactions between the bubbles and the development of characteristic structures on the spikes. Those effects result from several causes. Kelvin-Helmhotz effects occur at the sides of the spikes, what finally can result in the formation of typical mushroom shapes on their top (normally at lower Atwood numbers). This increases the effects of the retarding force on the spikes. At this stage larger bubbles assimilate smaller ones, which makes them grow larger and move faster.
Stage 4 is characterized by the breakup of spikes, the penetration of bubbles through a slab of fluid of finite thickness and other intricate effects through various complex mechanisms. The final stage correspond to a system of chaotic or turbulent mixing of two fluids.

4.1.2 Modeling and Computational Details

The simulations have been carried out in a completely filled, closed, two-dimensional (2D) box configuration with a length of 1000 Å and a height of 250 Å. Periodic boundary conditions were applied in x-dimension, the top and bottom box walls were set to reflect the particles. A layout of the simulation system is shown in figure 4.3.
In this model two different types of particles, P_H (heavy particles) and P_L (light particles), with different masses and interaction potential parameters were used. Argon was chosen as the heavier particle type. All other types and their parameters were derived relatively from this element. The total number of particles in the simulations varied from $2.1 \cdot 10^4$ to $2.7 \cdot 10^4$. They were initially placed on a periodical square-lattice, where P_H were placed in the upper half and P_L in the lower half of the simulation box. Initial velocities for both particle types were chosen to satisfy a Maxwell-Boltzmann velocity distribution for $T = 150K$.
The external gravitation acceleration constant g, was set to act downwards with a magnitude of $g = 0.02$ Å/ps^2. In order to accelerate the mixing process and to make it observable during

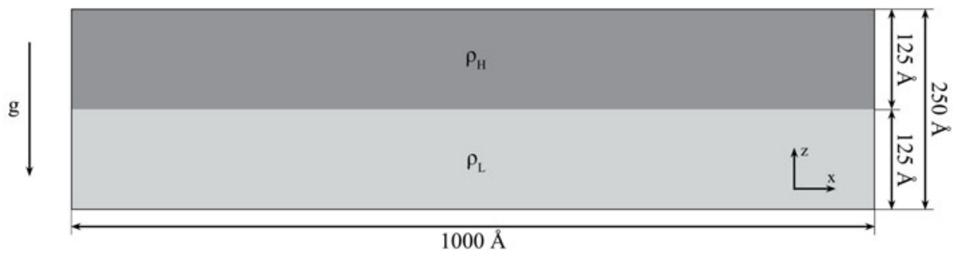


Figure 4.3: *Simulation system of the Rayleigh-Taylor Instability.*

name	N_H	N_L	m_H [au]	m_L [au]	A	σ_H [Å]	σ_L [Å]	ϵ_H [eV]	ϵ_L [eV]
A1	10878	10584	40	40	0.01	3.4	3.4	0.011	0.011
A2	10878	10584	40	20	0.35	3.4	3.4	0.011	0.011
B1	16065	10584	40	20	0.50	2.8	3.4	0.011	0.011
B2	16065	10584	100	20	0.77	2.8	3.4	0.011	0.011
B3	16065	10584	200	20	0.88	2.8	3.4	0.011	0.011
B4	16065	10584	2000	20	0.99	2.8	3.4	0.011	0.011

Table 4.1: *Simulation parameters: name indicates the simulation name, N is the numbers of particles in the upper or the lower half of the system, m represent the particle masses, σ and ϵ are parameters of the LJ potential.*

a reasonable simulation time, this value was chosen to be much stronger than g_{earth}.
The simulations were performed using the leap-frog integrator and a LJ interaction potential. A single time-step length δt of $2.0 \cdot 10^{-13}$ seconds was chosen, which yields for $5.0 \cdot 10^5$ time-steps a total simulation time of $0.1 \cdot 10^{-6}$ seconds.
Different simulations were carried out, varying masses and particle sizes. Table 4.1 shows a complete overview of all simulations with the pertinent particle parameters.

4.1.3 Results and Discussion

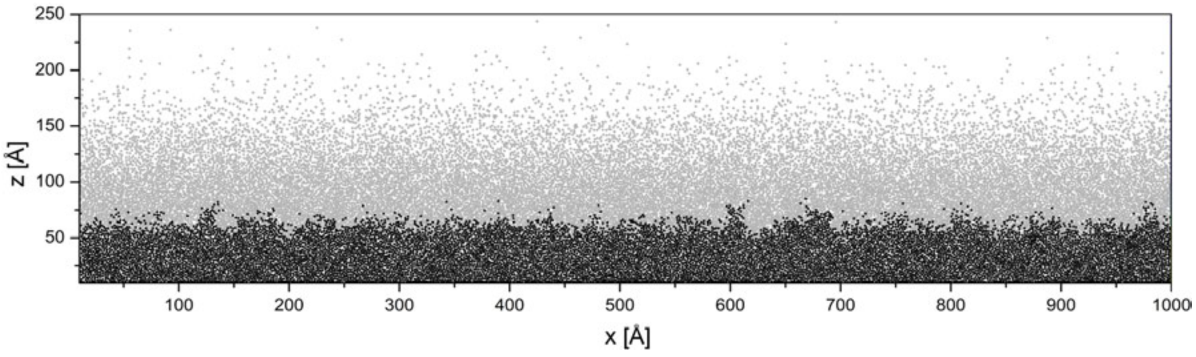
Simulation results are shown in figure 4.6, 4.4 and 4.5, where figure 4.6 shows the growth of the mixing layer, figure 4.4 the particle trajectories and figure 4.5 the density distribution of light particles in the system.
At $t = 0$ the external gravitation field is activated, which is equivalent to an initial shacking of the whole system. The particles start to accelerate towards the bottom wall, where they are compressed and hard reflected. This results in a slightly disturbed interface and an oscillating shockwave in the system. The shockwave compresses and decompresses the entire simulation system periodically from antiparallel, vertical directions (figure 4.5a and 4.5b). Hence the lighter fluid pushes the heavier one and vice versa, consequently the system destabilizes. Therefore all essential conditions for the formation of a RT instability are present and random

initial disturbances at the interface of the heavy and the light fluid start to shape (figure 4.4a and 4.4b).
First, light particles start to form sharp spikes and instability patterns, like mushrooms (figure 4.4c and 4.4d as well as 4.5c and 4.5d). Those patterns expand quickly and decrease the local particle density (from figure 4.4e to 4.4h and figure 4.5e to 4.5h).
The heavy particles start at the same time to penetrate the light particle-layer at the bottom. The chaotic phase begins, when the mushrooms start to interact and merge with each other, destroying regular instability patterns and starting the chaotic mixing of the system (figure 4.4d and 4.5d).

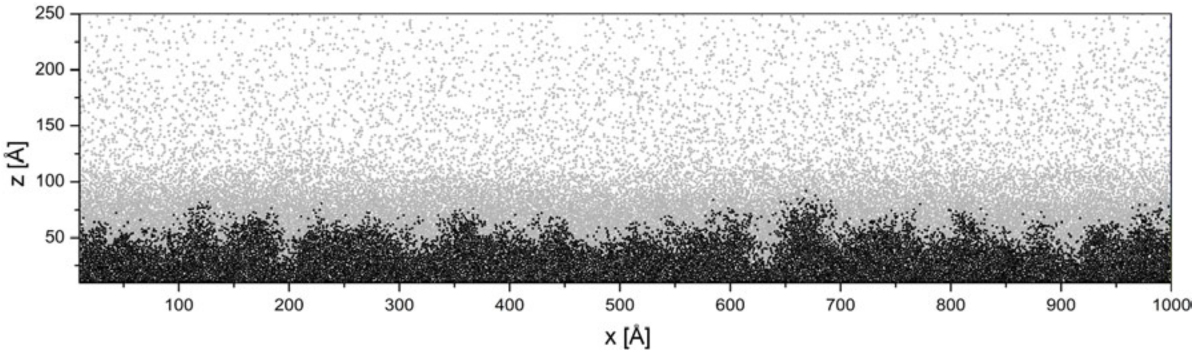
The size of the mixing layer (the penetration distance of spikes and bubbles) can be defined in a wide variety of ways. One possibility [29] is by defining a line average of the density of the light or the heavy fluid:

$$\bar{f}_L(z) = \frac{\int dx f_L(x,z)}{\int dx} \tag{4.7}$$

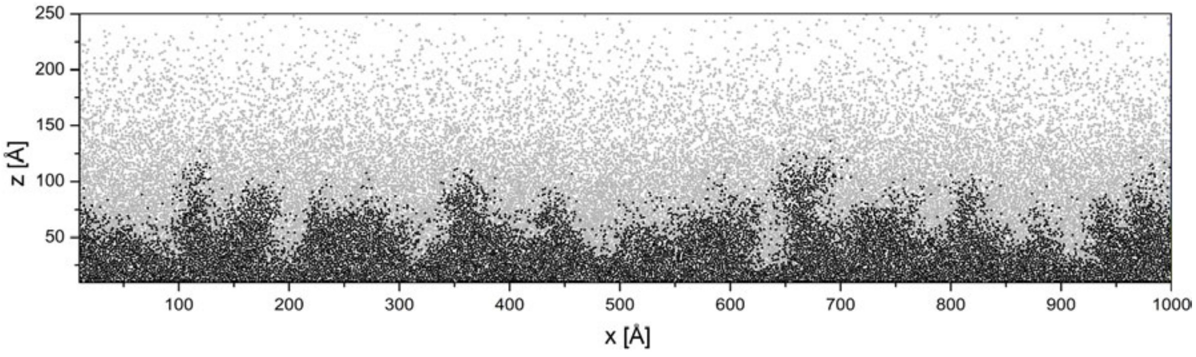
where $f_L(x, z)$ indicates the volume fraction of the light or the heavy fluid at point (x, z).
The size of the mixing region is then defined as the difference in height of the two points where $\bar{f}_L(z) = 0.01$ and $\bar{f}_L(z) = 0.99$.
Figure 4.6 shows the growth of the mixing layers for different mass and size ratios. It is obvious that the development of the instability strongly depends on the mass ratio of the light and the heavy particles. A dependency of the starting time of the mixing process on the size ratio was not observed. The mixing of both fluids started in all simulations approximately at the same moment. Subsequently an exponential growth of the penetration distance until approximately $0.6 \cdot 10^5$ time-steps can be observed (see figure 4.6b). Afterwards the disturbances grow linearly, which is already the change-over to the non-linear (turbulent) phase. This growth behavior conforms with the data of macroscopic experiments. The linear growth of the mixing layer is interfered with an oscillation that represents the repeatedly compression and decompression of the system that results from the shockwave caused by the initial shacking. Among the simulations, the highest growth rate can be observed for a mass ratio of $m_H/m_L = 100$, the lowest when both particle types have the same mass $m_H/m_L = 1$.



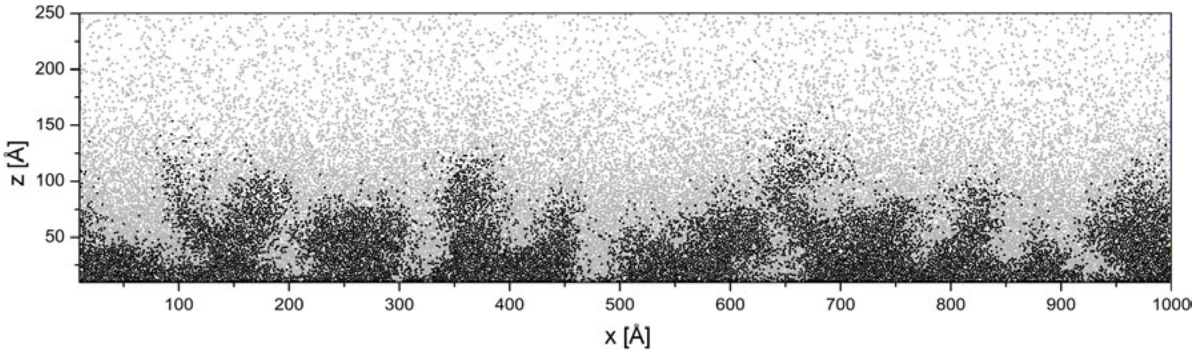
(a) $t = 0.75 \cdot 10^5$ *time-steps*



(b) $t = 1.50 \cdot 10^5$ *time-steps*

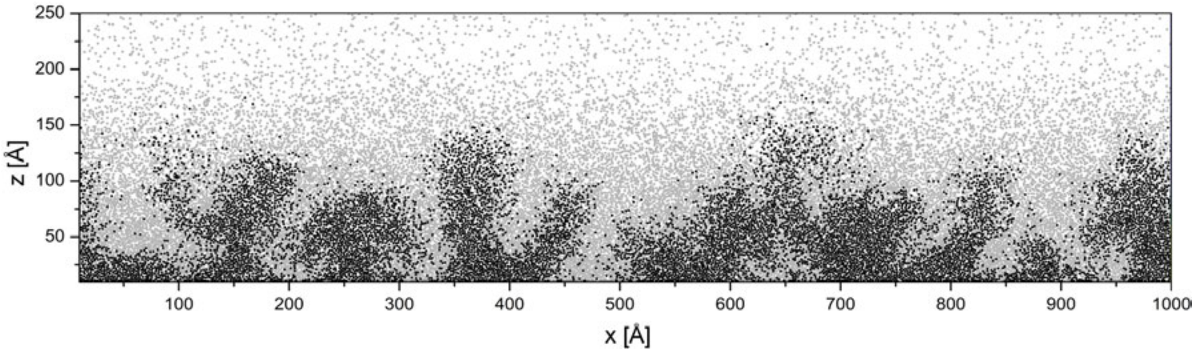


(c) $t = 1.75 \cdot 10^5$ *time-steps*

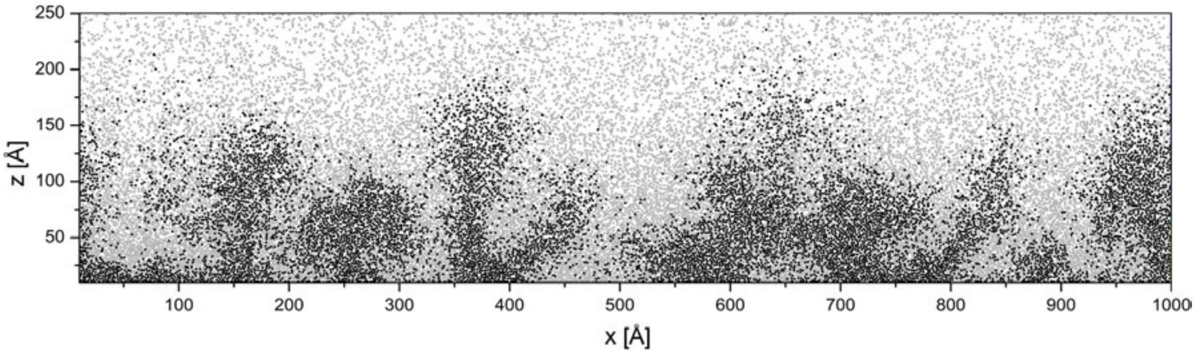


(d) $t = 2.25 \cdot 10^5$ *time-steps*

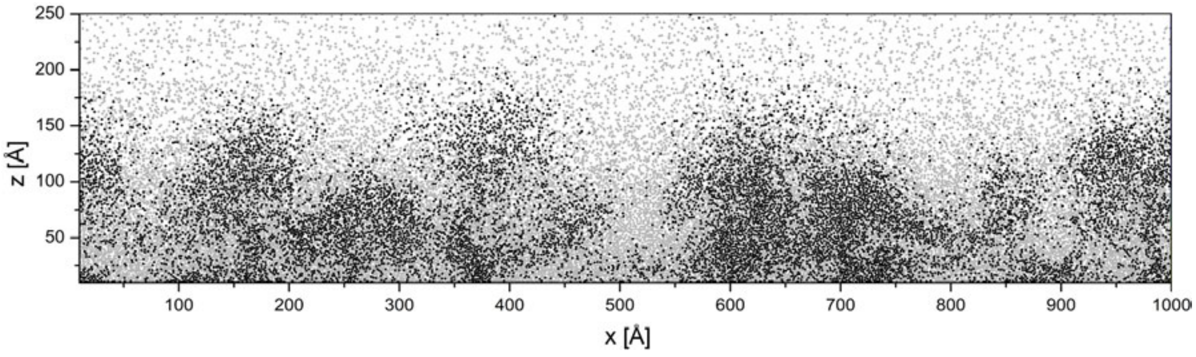
Figure 4.4: *Time development of particle positions: Heavy particles are colored in light gray, light particles in black. This figure continues on the following page.*



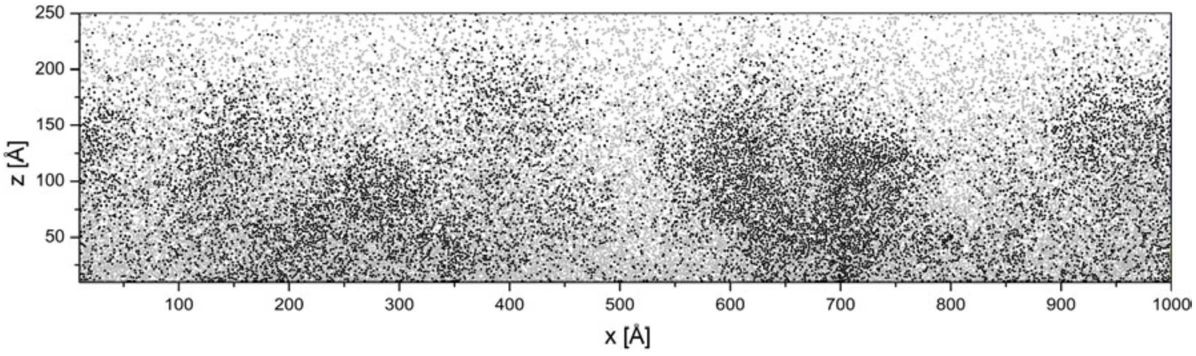
(e) $t = 2.50 \cdot 10^5$ *time-steps*



(f) $t = 2.75 \cdot 10^5$ *time-steps*

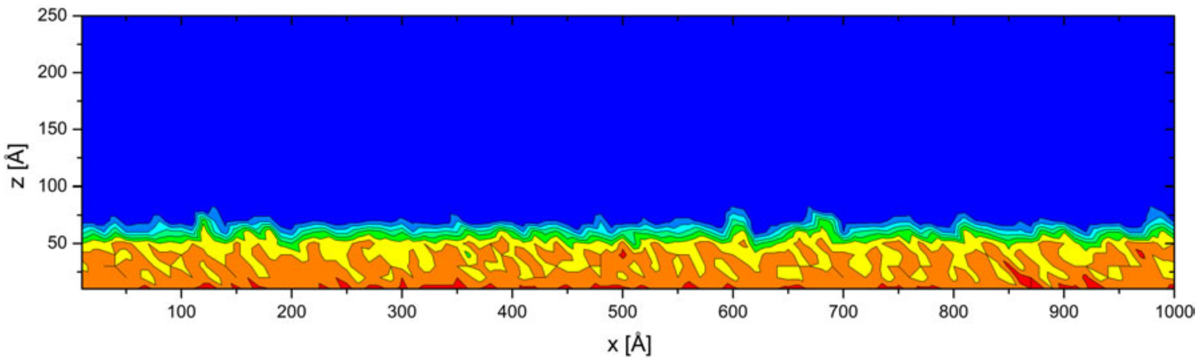


(g) $t = 3.25 \cdot 10^5$ *time-steps*

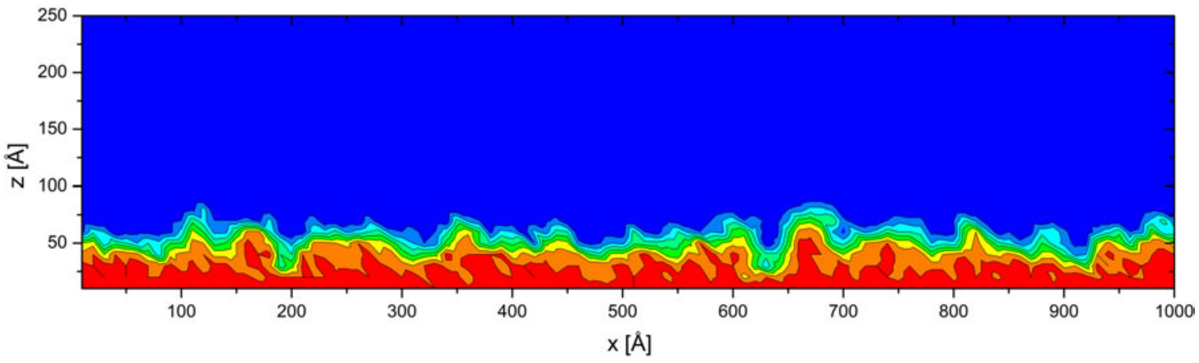


(h) $t = 3.75 \cdot 10^5$ *time-steps*

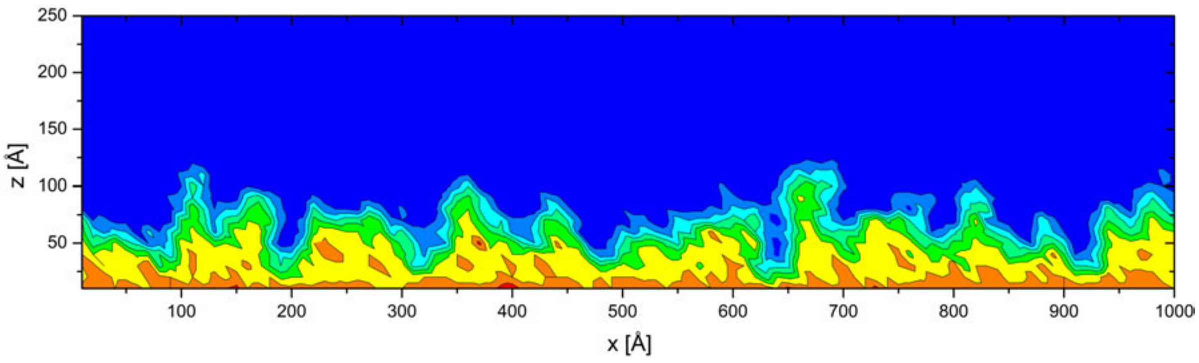
Figure 4.4: *Time development of particle positions: Heavy particles are colored in light gray, light particles in black. This figure is continued from the previous page.*



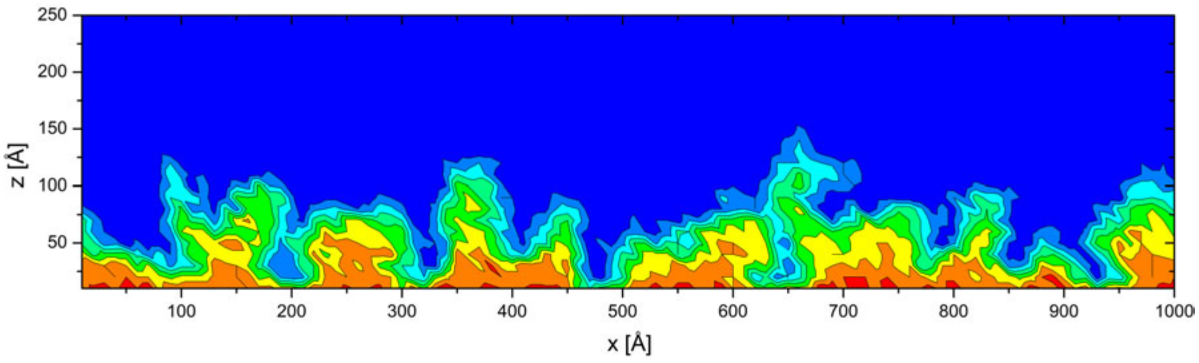
(a) $t = 0.75 \cdot 10^5$ *time-steps*



(b) $t = 1.50 \cdot 10^5$ *time-steps*

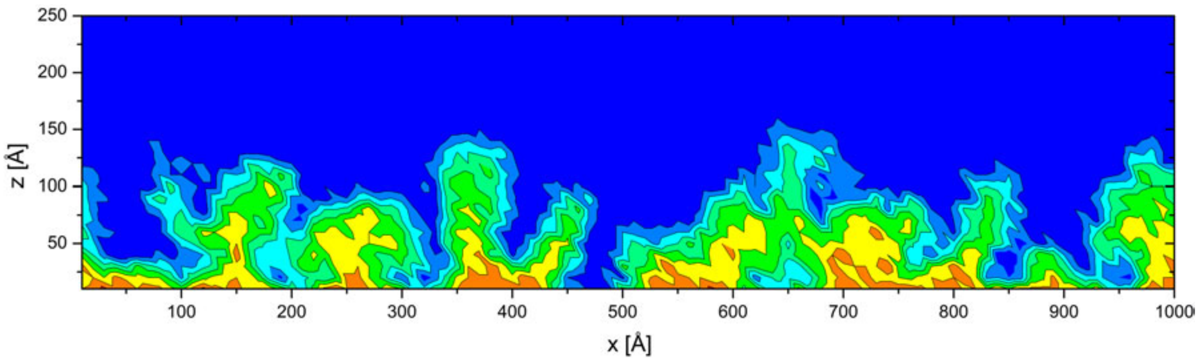


(c) $t = 1.75 \cdot 10^5$ *time-steps*

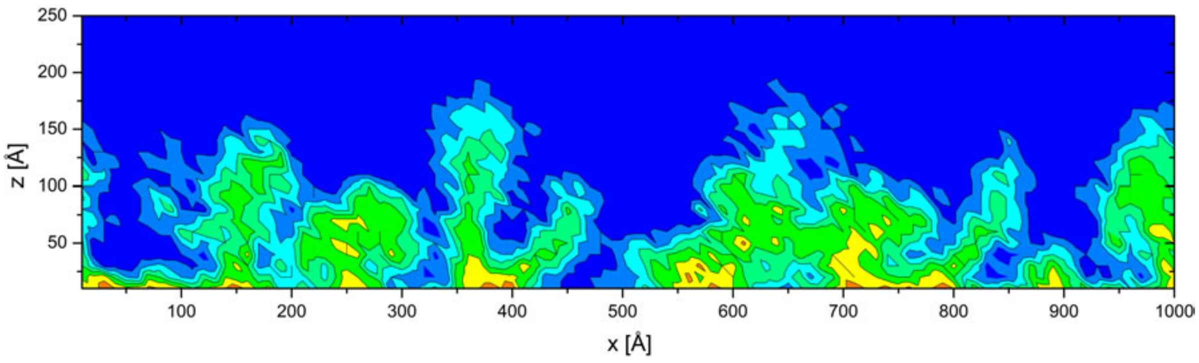


(d) $t = 2.25 \cdot 10^5$ *time-steps*

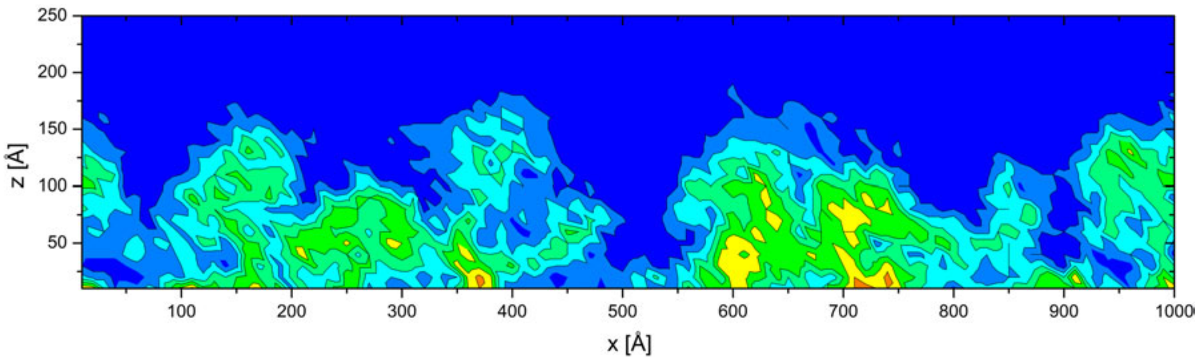
Figure 4.5: *Time development of the particle density distribution for the light particles: The density changes from red (high) to blue (no particles). This figure continues on the following page.*



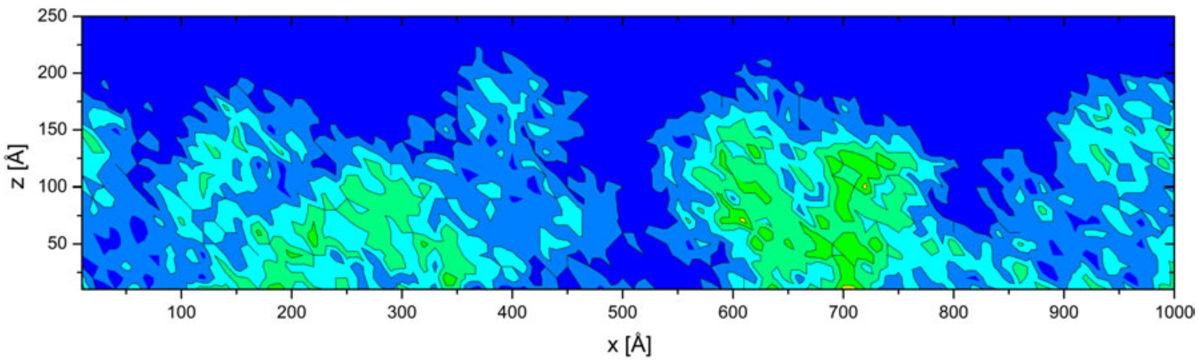
(e) $t = 2.50 \cdot 10^5$ *time-steps*



(f) $t = 2.75 \cdot 10^5$ *time-steps*

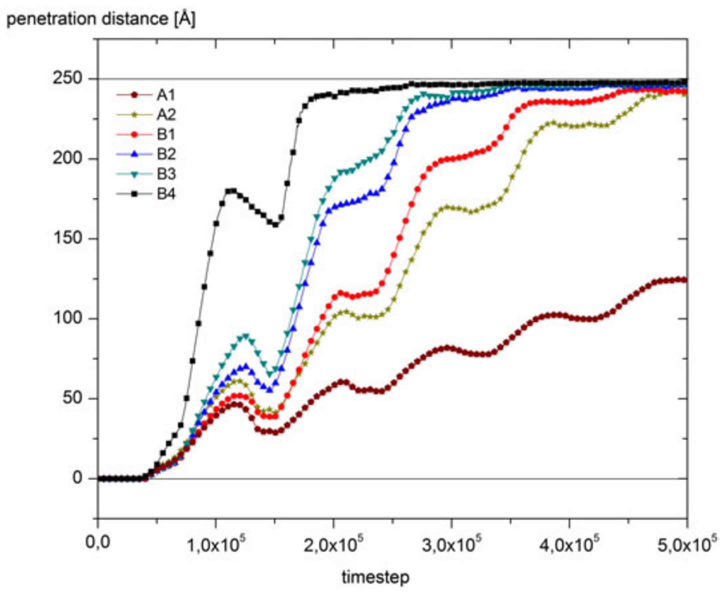


(g) $t = 3.25 \cdot 10^5$ *time-steps*

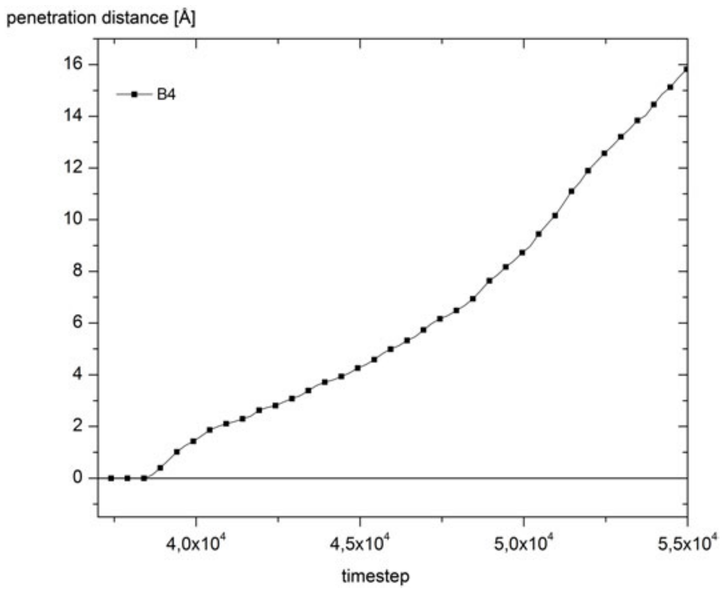


(h) $t = 3.75 \cdot 10^5$ *time-steps*

Figure 4.5: *Time development of the particle density distribution for the light particles: The density changes from red (high) to blue (no particles). This figure is continued from the previous page.*



(a) *Penetration distances*



(b) *Detailed view*

Figure 4.6: *Growth of the mixing layer for different mass and size ratios: Figure a) illustrates the growth of the mixing layers of all simulations (compare table 4.1). Two horizontal lines at 0 Å and 250 Å mark the horizontal boundaries of the system. Figure b) shows a detailed view in the range of the exponential growth of simulation B4.*

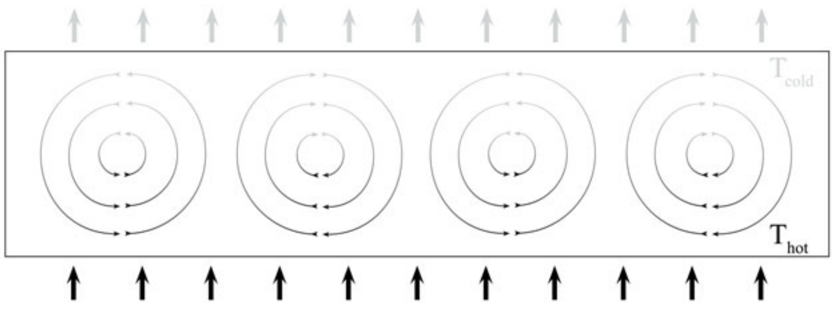


Figure 4.7: *Schema of the Rayleigh-Benard Convection: When the temperature of the bottom plate (T_{hot}) is slightly higher than the temperature of the upper one (T_{cold}) a permanent flow of thermal energy will conduct through the system. Convection patterns arise because of thermal conductivity and convection cells arise due to gravitational forces.*

4.2 Rayleigh-Benard Convection

4.2.1 Introduction

The Rayleigh-Benard (RB) system is one of the most representative nonequilibrium hydrodynamic systems. The phenomenon was first described by Benard in 1900 and later complemented by Baron Rayleigh.
The RB problem deals with a fluid layer confined between two horizontal parallel plates, which are kept at different temperatures. If the temperature- difference (or more precisely, the Rayleigh number) between the lower and the upper plate is smaller than a critical value, a heat conduction state is established. However, if the temperature-difference exceeds this critical value, convection offers a more efficient method of heat transfer and convection flow patterns (as convection rolls due to gravitational forces) emerge (see figure 4.7).

The relevance of the RB convection reaches from solar astrophysics phenomena (like the formation of convection bubbles in hot stars) and meteorological phenomena (as the convective shifting and mixing of huge gas masses that are responsible for the creation of storms) to industrial processes (such as used in cooling systems for example).

RB convection has been studied in detail experimentally and numerically. A review of all relevant studies was published by Ahlers [17] and Cross et al. [9]. During the last two decades the RB phenomenon was also observed at the microscopic level using a wide range of different computer-simulation approaches like the *MD method*, the *Dynamical Non-Equilibrium Molecular Dynamics (D-NEMD)* method [18], the *Direct Simulation Monte Carlo (DSMC)* method [3] or the *Smooth-Particle Applied Mechanics (SPAM)* method [19].
The earliest work to investigate the RB convection phenomenon using the MD method was published by Mareschal et al. [31, 32] in 1987. They showed that RB convection cells can be already observed within a hard-disk fluid system consisting of about $5.0 \cdot 10^3$ particles. The

RB convection was also simulated using the MD method by Rapaport [10, 11], who observed the development of convective rolls in a hard-disk fluid driven by opposed temperature and gravitational fields. In a recent work Rapaport [13] used MD simulations to model pattern formation in three-dimensional RB convection. Mareschal [33], Puhl et al. [6] and Given et al. [4] compared the macroscopic simulations results to the corresponding analytical and numerical solution of the macroscopic hydrodynamic equations. Watanabe et al. [41] concentrated in their work on the chaotic motions of particles in RB convection rolls.
In a recent work Mugnai et al. [27] applied the D-NEMD method to describe the formation of convective rolls in two-dimensional fluid systems of soft-disks.
Garcia [21] and Sefanov et al. [39] studied the convection rolls using the DSMC method. The phenomenon has also been extensively studied using the DSMC method by Watanabe et al. [42–44], who showed the transition between conduction and convection as well as the growth of the spatial correlations of temperature fluctuations in this transition.
Posch, Hoover and Kum [5, 34] used the SPAM method to investigate the RB problem.

The purpose of this work was to perform state-of-the-art MD simulations on the well studied RB convection, as an other example of complex hydrodynamic phenomena, to reproduce and combine the results of earlier works in the microscope scale and to verify the modelMD code.

The numerical and analytical investigation of the RB problem bases on the *Navier-Stokes equations* as well as on energy, mass and momentum conservation. Together with the *Boussinesq approximation* [23] and the assumption that the material properties of the fluid are constant this can be transformed into a simplified system of equations. Here all fluid properties and the driving forces are represented by two dimensionless quantities, namely the *Rayleigh number* Ra and the *Prantl number* Pr:

$$Ra = \frac{\alpha g L_z^3 \Delta T}{\nu \kappa} \tag{4.8}$$

$$Pr = \frac{\nu}{\kappa} \tag{4.9}$$

where α is the thermal expansion coefficient, g the gravitational acceleration and L_z the height of the layer. ΔT represents the temperature difference between the horizontal plates, ν the kinematic viscosity and κ the diffusivity.

Equations (4.8) and (4.9) represent the basic theoretical parameters for studies of the RB convection problem. Although this set of equations is a rough approximation it remains very complex and further simplifications have to be made before the system can be used for numerical studies [38].
The linear stability analysis can be used to calculate the critical value Ra_c of the Rayleigh

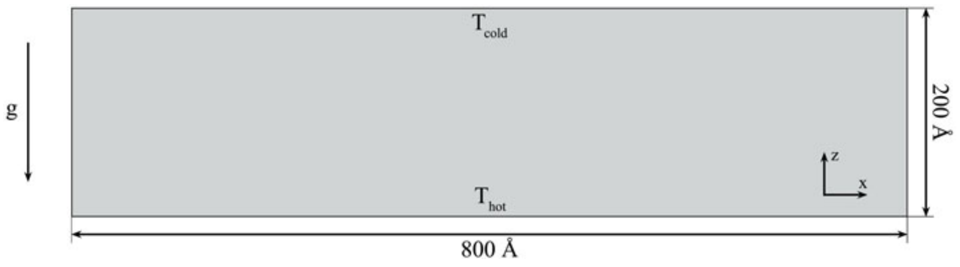


Figure 4.8: *Simulation system of the Rayleigh-Benard Convection.*

number at which convection flow patterns start to appear. For an infinite horizontal layer it yields: $Ra_c = 1708$ [11].
With the same method it is possible to calculate the preferred wavelength of the convective vertices. This method yields that the preferred width of convection rolls is equal to the height of the simulation system and therefore the rolls have typically a square cross-section [11].

4.2.2 Modeling and Computational Details

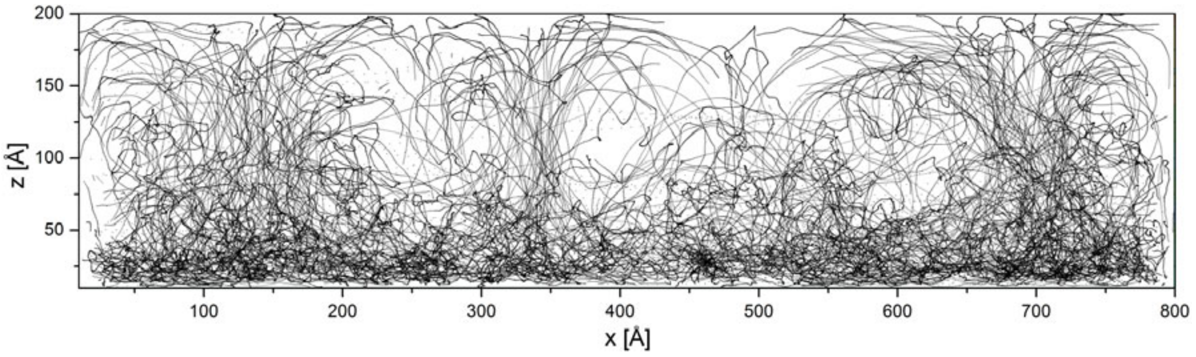
The simulations have been carried out in a completely filled, closed, two-dimensional (2D) box configuration with a length varying between $L_x = 200$ Å and $L_x = 800$ Å and a height of $L_z = 200$ Å. This correspondents to aspect ratios from $L_x : L_z = 1 : 1$ to $L_x : L_z = 4 : 1$.
Periodic boundary conditions were applied in x-dimension, the top and bottom box wall were connected to an ideal heat bath with temperature ratios varying from $T_{hot} : T_{cold} = 1 : 1$ to $T_{hot} : T_{cold} = 1000 : 1$.
The particle type used in this model was Argon with with the mass $m = 40au$ and Lennard-Jones parameters $\sigma = 3.4$ Å and $\epsilon = 0.011$ eV. The total number of particles in the simulations varied from $3.6 \cdot 10^3$ to $1.4 \cdot 10^4$. Initially they were placed on a periodical square-lattice. Initial velocities were chosen to satisfy a Maxwell-Boltzmann distribution for $T = 3.5 \cdot 10^4 K$. Very high initial temperatures as well as very high wall temperatures and a high external gravitation acceleration constant g of 0.02 Å/ps^2, were chosen to expedite the formation process of convection cells and to make it observable during a reasonable simulation time.
A schema of the simulation system is shown in figure 4.8.
The simulations were performed using the leap-frog integrator and a LJ interaction potential. A single time-step length δt of $2.0 \cdot 10^{-13}$ seconds was chosen, which yields for $4.5 \cdot 10^6$ time-steps a total simulation time of $0.9 \cdot 10^{-5}$ seconds.
Different simulations were carried out, varying vertical wall temperature- and aspect-ratios. Table 4.2 shows a complete overview of all simulations with the pertinent particle parameters.

name	N	L_x [Å]	L_z [Å]	$L_x : L_z$	T_{hot} [K]	T_{cold} [K]	$T_{hot} : T_{cold}$	T_{init} [K]
A1	14762	800	200	4:1	$3.50 \cdot 10^6$	$3.50 \cdot 10^6$	1:1	$3.50 \cdot 10^4$
A2	14762	800	200	4:1	$3.50 \cdot 10^6$	$1.75 \cdot 10^6$	2:1	$3.50 \cdot 10^4$
A5	14762	800	200	4:1	$3.50 \cdot 10^6$	$7.00 \cdot 10^5$	5:1	$3.50 \cdot 10^4$
A10	14762	800	200	4:1	$3.50 \cdot 10^6$	$3.50 \cdot 10^5$	10:1	$3.50 \cdot 10^4$
A100	14762	800	200	4:1	$3.50 \cdot 10^6$	$3.50 \cdot 10^4$	100:1	$3.50 \cdot 10^4$
A100b-e	14525	800	200	4:1	$3.50 \cdot 10^6$	$3.50 \cdot 10^4$	100:1	$3.50 \cdot 10^4$
A1000	14762	800	200	4:1	$3.50 \cdot 10^6$	$3.50 \cdot 10^3$	1000:1	$3.50 \cdot 10^4$
B1	7381	400	200	2:1	$3.50 \cdot 10^4$	$3.50 \cdot 10^6$	100:1	$3.50 \cdot 10^4$
B2	3660	200	200	1:1	$3.50 \cdot 10^4$	$3.50 \cdot 10^6$	100:1	$3.50 \cdot 10^4$
B3	14641	400	400	1:1	$7.00 \cdot 10^4$	$7.00 \cdot 10^6$	100:1	$7.00 \cdot 10^4$

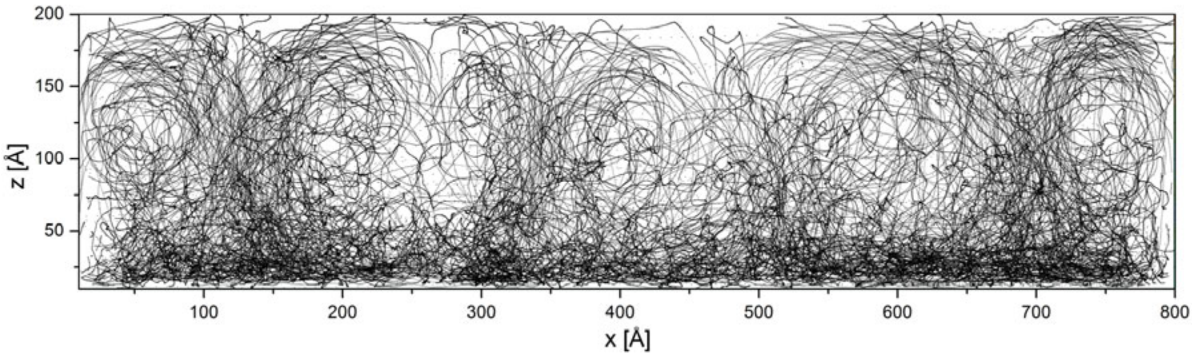
Table 4.2: *Simulation parameters: name indicates the simulation name, N the number of particles in the system, L_x and L_z represent the system size, T_{hot} and T_{cold} are the temperatures at the lower (hot) and upper (cold) boundary and T_{init} is the initialization temperature of the simulation system.*

4.2.3 Results and Discussion

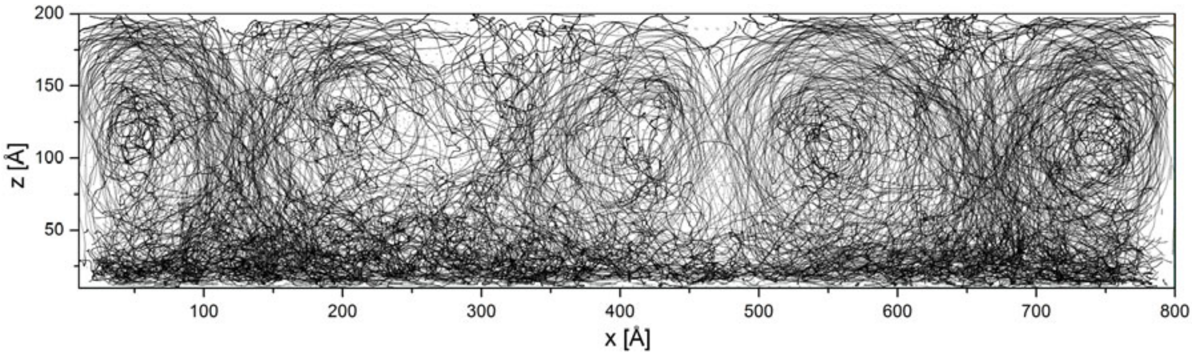
A series of simulations were performed over a wide range of temperature ratios from $T_{hot} : T_{cold} = 1 : 1$ (simulation $A1$) to $T_{hot} : T_{cold} = 1000 : 1$ (simulation $A1000$). As assumed, no indication of convective flow were observed for the lowest temperature gradients $T_{hot} : T_{cold} = 1 : 1$ and $T_{hot} : T_{cold} = 2 : 1$. For simulations with a temperature ratio of $T_{hot} : T_{cold} = 5 : 1$ and higher, convective rolls started to appear after approximately $0.2 \cdot 10^6$ time-steps and were fully developed at $2.0 \cdot 10^6$ time-steps.
The formation process of the convection vortices for simulation $A100$ (with a temperature ratio of $T_{hot} : T_{cold} = 100 : 1$) is described in the following. Figure 4.9 illustrates the particle trajectories and figure 4.10 the density distribution of the simulation system.
At time $t = 0$ the bottom and top box wall were connected with ideal heat bathes with temperatures of $T_{hot} = 3.50 \cdot 10^6$ K and $T_{cold} = 3.50 \cdot 10^4$ K. The system (with an initial temperature of $T_{init} = 3.50 \cdot 10^4$ K started to heat up at the bottom and cool down at the top. Six unevenly spaced convection rolls (where adjacent vortices rotate in opposite directions) began to appear after $0.2 \cdot 10^6$ time-steps (figures 4.9a and 4.10a), grew (figures 4.9b and 4.9c as well as 4.10b and 4.10c) and were fully developed after $0.9 \cdot 10^6$ time-steps (figure 4.9d and 4.10d).
At $1.25 \cdot 10^6$ time-steps a fusion of four adjacent rolls occurred (figure 4.9e and 4.10e). This process was accompanied by a broadening of the four resulting vortices (figure 4.9f and 4.10f). These four rolls remained stable (figures 4.9g and 4.9h as well as 4.10g and 4.10h) until the end of the simulation run, which was terminated after $4.5 \cdot 10^6$ time-steps. The number of resulting vortices conforms with the predicted number of convection rolls from the linear stability analysis for a simulation system with an aspect ratio of $L_x : L_z = 4 : 1$.



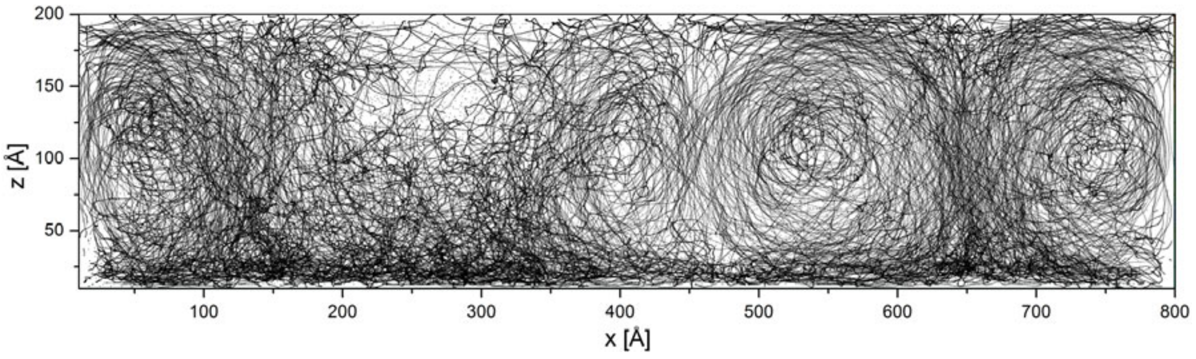
(a) $t = 0.10 \cdot 10^6$ *to* $0.80 \cdot 10^6$ *time-steps*



(b) $t = 0.20 \cdot 10^6$ *to* $0.90 \cdot 10^6$ *time-steps*

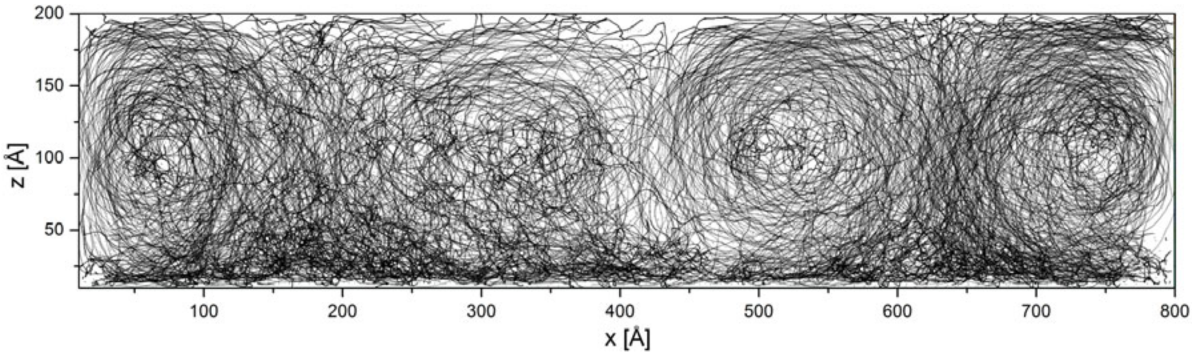


(c) $t = 0.55 \cdot 10^6$ *to* $1.25 \cdot 10^6$ *time-steps*

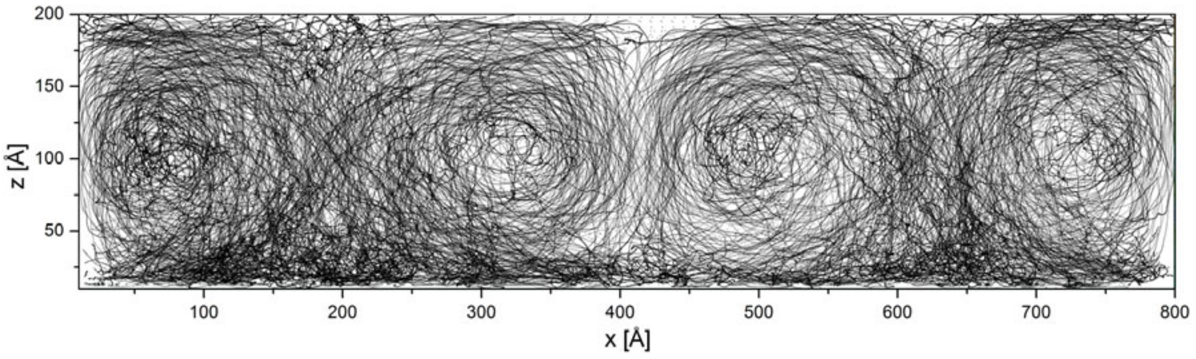


(d) $t = 0.90 \cdot 10^6$ *to* $1.60 \cdot 10^6$ *time-steps*

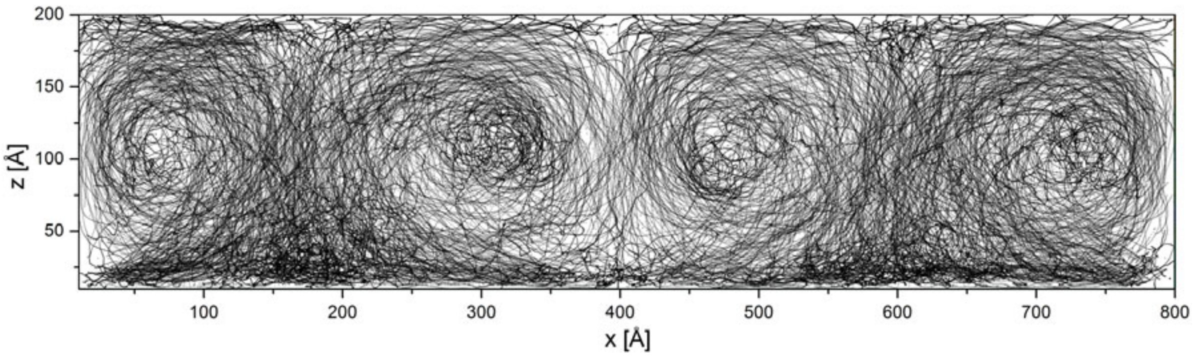
Figure 4.9: *Time development of streamlines: Particle positions are plotted for a period of* $0.7 \cdot 10^6$ *time-steps. Only particles are shown, which are at positions* $z < 10$ *Å at the first time-step of each each sub figure. This figure continues on the following page.*



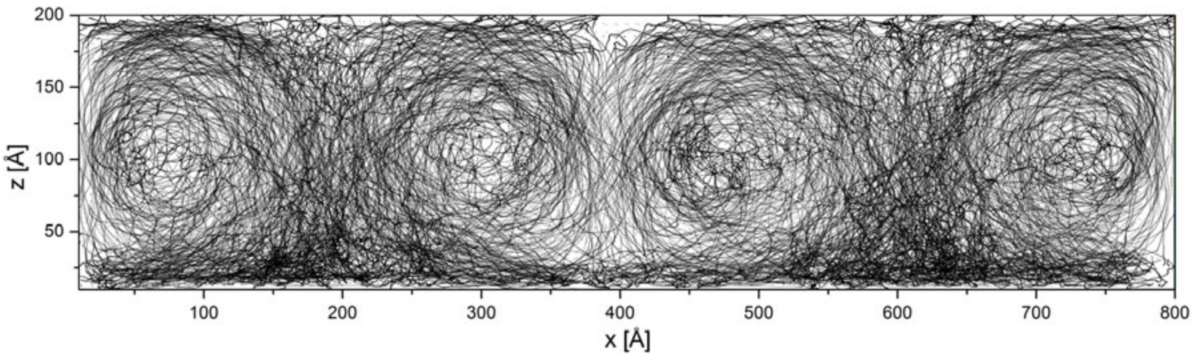
(e) *$t = 1.25 \cdot 10^6$ to $1.95 \cdot 10^6$ time-steps*



(f) *$t = 1.60 \cdot 10^6$ to $2.30 \cdot 10^6$ time-steps*

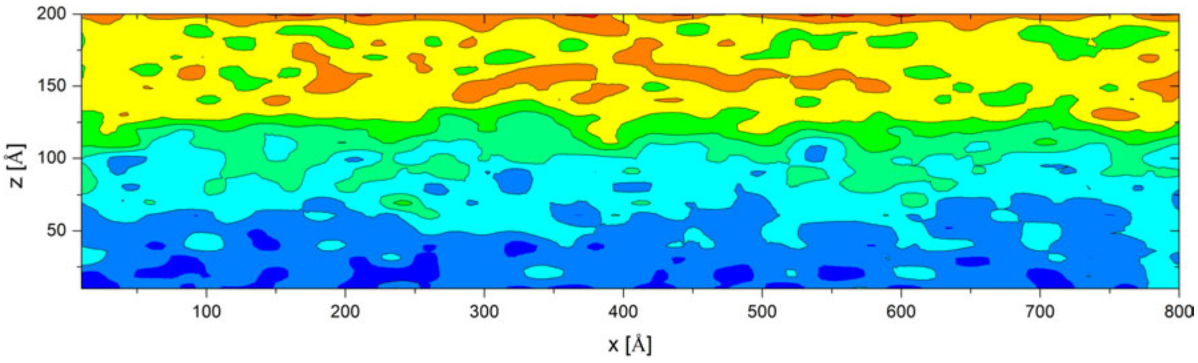


(g) *$t = 2.30 \cdot 10^6$ to $3.00 \cdot 10^6$ time-steps*

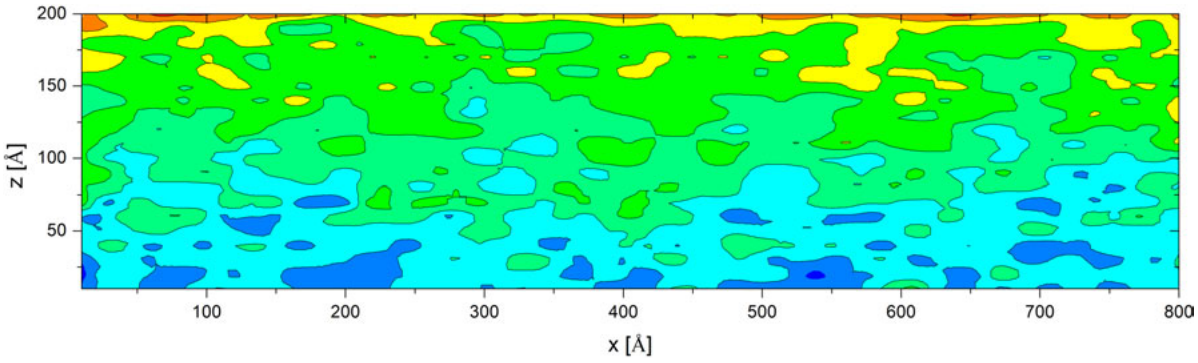


(h) *$t = 3.00 \cdot 10^6$ to $3.70 \cdot 10^6$ time-steps*

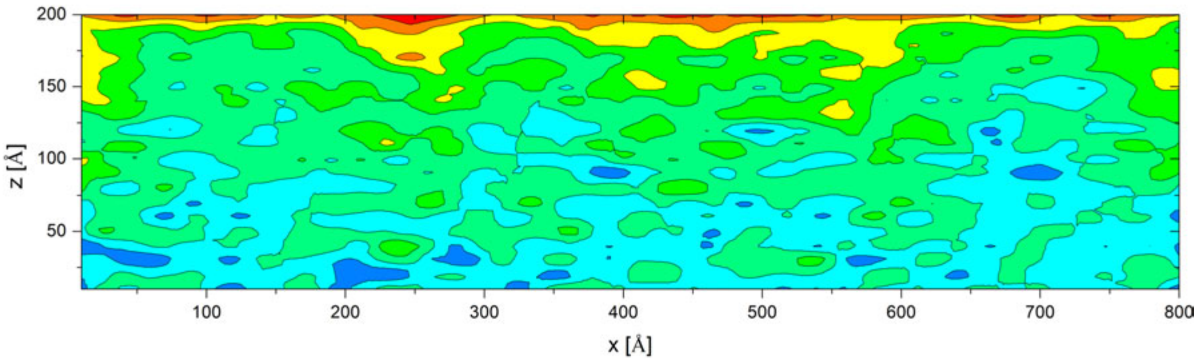
Figure 4.9: *Time development of streamlines: Particle positions are plotted for a period of $0.7 \cdot 10^6$ time-steps. Only particles are shown, which are at positions $z < 10$ Å at the first time-step of each sub figure. This figure is continued from the previous page.*



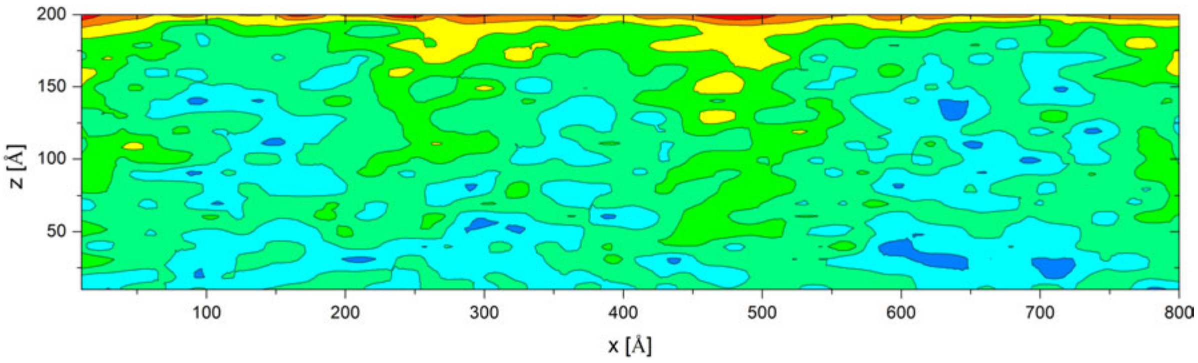
(a) $t = 0.10 \cdot 10^6$ *time-steps*



(b) $t = 0.20 \cdot 10^6$ *time-steps*

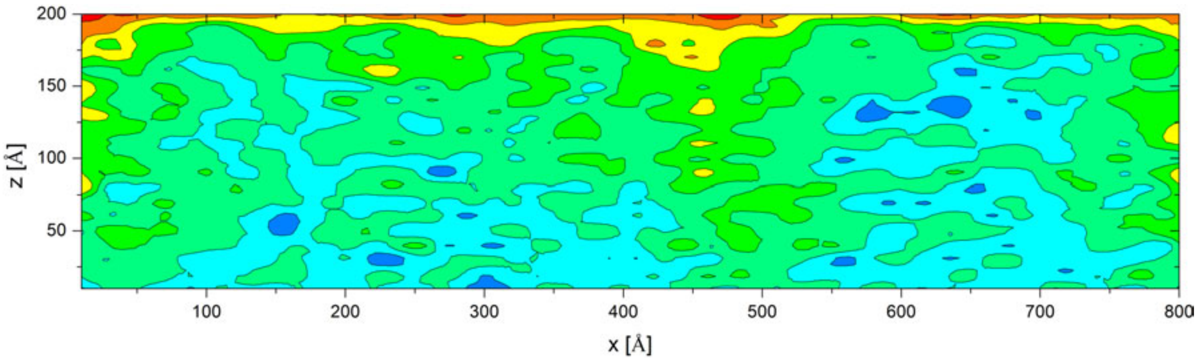


(c) $t = 0.55 \cdot 10^6$ *time-steps*

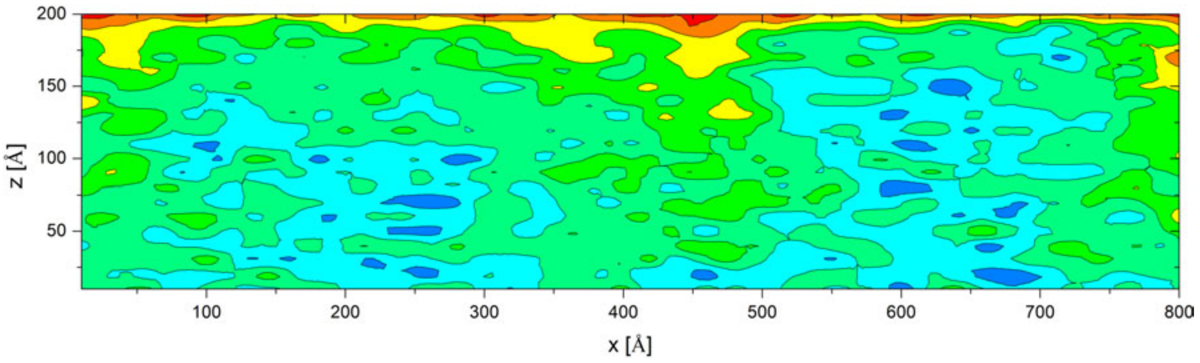


(d) $t = 0.90 \cdot 10^6$ *time-steps*

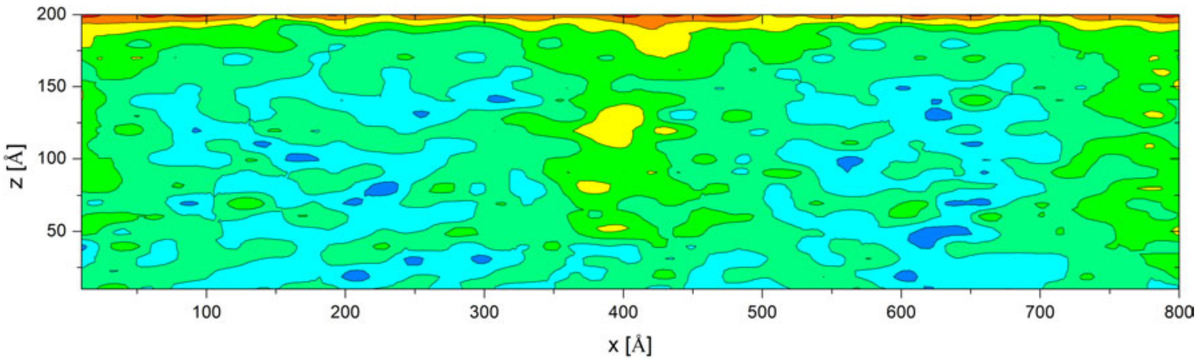
Figure 4.10: *Time development of the particle density distribution for the light particles: The density changes from red (high) to blue (no particles). This figure continues on the following page.*



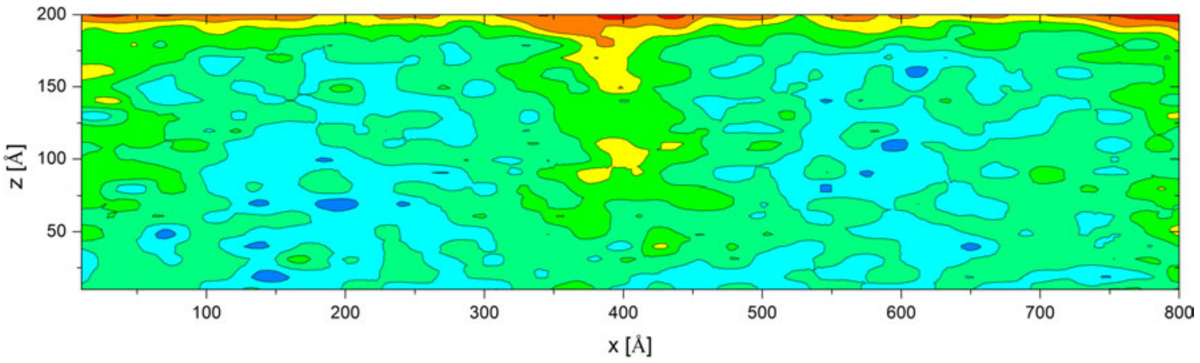
(e) $t = 1.25 \cdot 10^6$ *time-steps*



(f) $t = 1.60 \cdot 10^6$ *time-steps*



(g) $t = 2.30 \cdot 10^6$ *time-steps*



(h) $t = 3.00 \cdot 10^6$ *time-steps*

Figure 4.10: *Time development of the particle density distribution for the light particles: The density changes from red (high) to blue (no particles). This figure is continued from the previous page.*

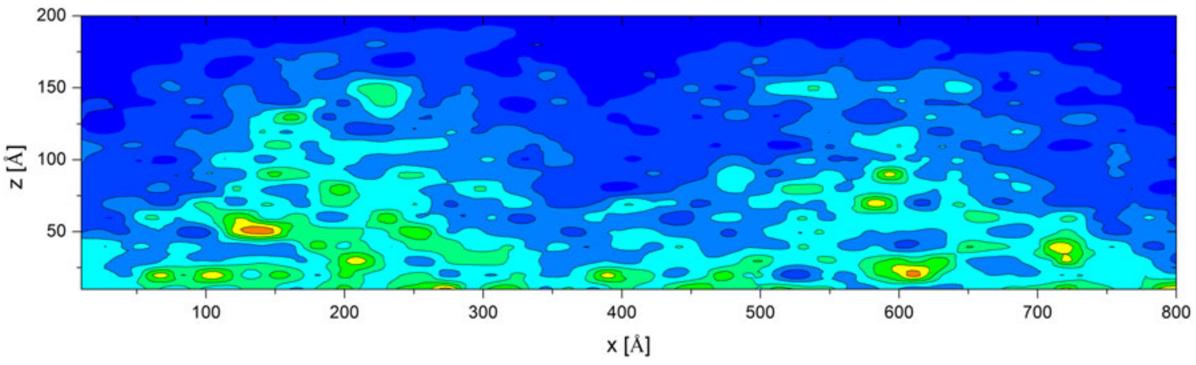


Figure 4.11: *Temperature distribution of simulation* $A100$ *after* $3.0 \cdot 10^6$ *time-steps: The temperature changes from red (hot) to blue (cold).*

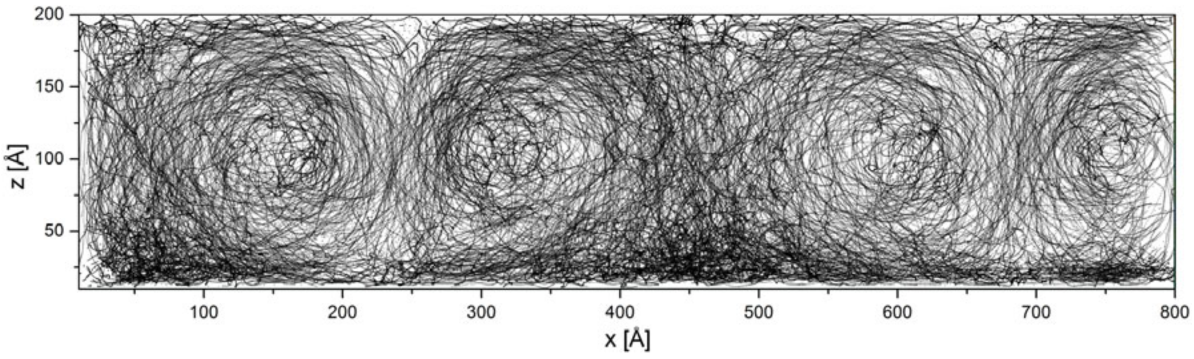
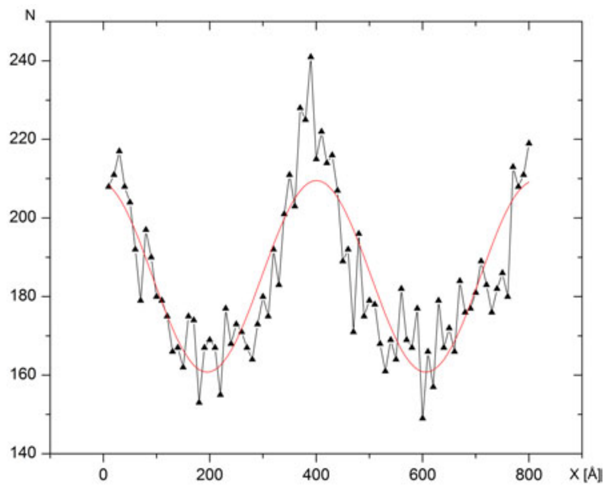


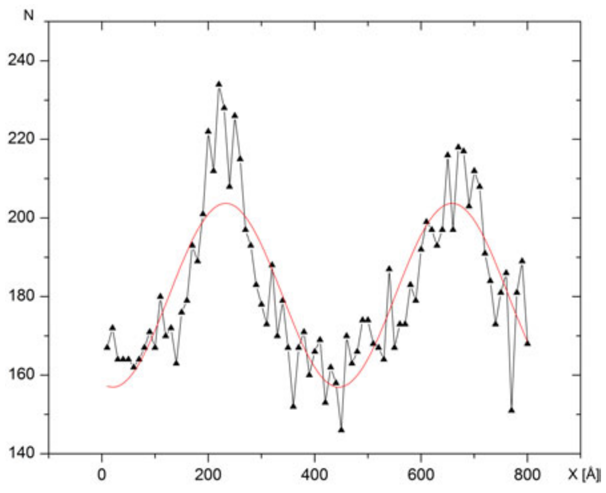
Figure 4.12: *Time development of streamlines for simulation system* $A100b$*: Particle positions are plotted from* $t = 3.00 \cdot 10^6$ *to* $3.70 \cdot 10^6$ *time-steps. Only particles are shown, which are at a positions* $z < 10$ *Å at the beginning of the period.*

Figure 4.11 shows a temperature distribution profile of the simulation system after $3.0 \cdot 10^6$ time-steps. The contour plot indicates two hot convection cells centered at $x = 200$ Å and $x = 600$ Å, respectively. By comparing this result with the particle trajectories (figure 4.9h) and the density distribution (figure 4.10h) at the same time-step, it is obvious that at these positions hot particles are accelerated upwards. Afterwards they are cold down at the upper plate, move downwards (because of gravitational forces) at $x = 0$ Å and $x = 400$ Å, and the process repeats. This procedure results in four vortices located at approximately $x = 50$ Å, $x = 300$ Å, $x = 450$ Å and $x = 750$ Å, where adjacent rolls rotate in opposite directions.

Four further simulations ($A100b$-$A100e$) with the same conditions as simulation $A100$, but with different random initial velocity directions, were performed to verify the appearance of the long-live four-roll state. Unlike similar experiments performed by Rapaport [10], who couldn't reproduce an equal state, the simulations resulted again in the formation of four vortex patterns (as shown in figure 4.12).



(a) *Simulation system A100*



(b) *Simulation system A100b*

Figure 4.13: *Horizontal density profile of the simulation systems a) $A100$ and b) $A100b$ as function of x after $3.0 \cdot 10^6$ time-steps. The center of the convection vortices are located at the inflection points of the fitted sine function.*

Figure 4.13 illustrates the horizontal density profile of the simulation systems $A100$ and $A100b$ as function of x after $3.0 \cdot 10^6$ time-steps. Here higher densities indicate cold sections of the simulation system, while lower ones indicate hot parts. The measured data were fitted with a sine function to illustrate the periodicity of the phenomenon due to the horizontal periodic boundary conditions.

This simulation was repeated for systems with different aspect ratios. Two fully developed and stable vortices (as predicted by the linear stability analysis) were observed for $L_x : L_z = 1 : 2$ (simulation $B1$) after a simulation runtime of $3.0 \cdot 10^6$ time-steps.
However, no stable convection flow patterns appeared for a simulation system with an aspect ratio of $L_x : L_z = 1 : 1$ (simulation $B2$), not even after $6.0 \cdot 10^6$ time-steps. An explanation for this result could be the relatively low number of particles ($N = 3660$) used in the simulation, since convection cells haven't been observed for systems consisting of less than 5000 particles so far [32]. For that reason a box configuration with the same aspect ratio of $L_x : L_z = 1 : 1$ but with an extended length and height of $L_x = L_z = 400$ Å and therefore a higher number of particles ($N = 14641$) was chosen (simulation $B3$). In this simulation system one fully developed vortex was observed after a simulation runtime of $3.0 \cdot 10^6$ time-steps.

Conclusion

In this master thesis, theoretical foundations of classical Molecular Dynamics simulation methods were presented and summarized.
Furthermore a high-performance classical Molecular Dynamics simulation software package, including a simulator as well as a graphical user interface, has been developed. Although the current version of this software is already capable to perform complex Molecular Dynamics simulations, there is still a number of interesting features (such as a parallelized implementation for distributed memory machines, long-range potentials, many-body potentials as well as additional temperature and pressure control mechanism) left to be implemented that would improve computation performance and enhance simulation possibilities.
With the developed software package various simulations were performed on complex fluid-dynamical phenomena, namely on the Rayleigh-Taylor instability and on the Rayleigh-Benard convection.
The Rayleigh-Taylor instability phenomenon was successfully simulated for a completely filled, closed, two-dimensional box configuration with random initial perturbations. Various particle parameter combinations, which were relatively derived from the element Argon, were used in the simulations and the results conformed with the data of macroscopic experiments. All stages of the growth process of the instability as well as characteristically structures, such as bubbles, spikes and mushroom shapes, have been observed.
Moreover it was possible to simulate the Rayleigh-Benard convection phenomenon effectively for a completely filled, closed, two-dimensional Argon system with various aspect ratios and a wide range of temperature ratios between the bottom and the top plate. The formation process of the convection vortices, which number conforms with the predicted number of convection rolls from the linear stability analysis, has been observed.

Bibliography

[1] The message passing interface (mpi) standard. 32

[2] The openmp api specification for parallel programming. 33, 35

[3] Bird G. A. Molecular gas dynamics. *NASA STI/Recon Technical Report A*, 76:40225–+, 1976. 58

[4] Given J. A. and Clementi E. Molecular dynamics and rayleigh–benard convection. *The Journal of Chemical Physics*, 90(12):7376–7383, 1989. 59

[5] Posch H. A., Hoover W. G., and Kum O. Steady-state shear flows via nonequilibrium molecular dynamics and smooth-particle applied mechanics. *Phys. Rev. E*, 52(2):1711–1720, Aug 1995. 59

[6] Puhl A., Malek Mansour M., and Mareschal M. Quantitative comparison of molecular dynamics with hydrodynamics in rayleigh-bénard convection. *Phys. Rev. A*, 40(4):1999–2012, Aug 1989. 59

[7] Tarmyshov K. B. and Muller-Plathe F. Parallelizing a molecular dynamics algorithm on a multiprocessor workstation using openmp. *Journal of Chemical Information and Modeling*, 45(6):1943–1952, 2005. 32

[8] Berendsen H. J. C., van der Spoel D., and van Drunen R. Gromacs: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications*, 91(1-3):43–56, 1995. 32

[9] Cross M. C. and Hohenberg P. C. Pattern formation outside of equilibrium. *Rev. Mod. Phys.*, 65(3):851, Jul 1993. 58

[10] Rapaport D. C. Molecular-dynamics study of rayleigh-bénard convection. *Phys. Rev. Lett.*, 60(24):2480–2483, Jun 1988. 59, 66

[11] Rapaport D. C. Unpredictable convection in a small box: Molecular-dynamics experiments. *Phys. Rev. A*, 46(4):1971–1984, Aug 1992. 59, 60

[12] Rapaport D. C. *The Art of Molecular Dynamics Simulation.* Cambridge University Press, New York, NY, USA, 1995. 12, 17

[13] Rapaport D. C. Hexagonal convection patterns in atomistically simulated fluids. *Phys. Rev. E*, 73(2):025301, Feb 2006. 59

[14] Frenkel D. and Smit B. *Understanding Molecular Simulation, Second Edition: From Algorithms to Applications (Computational Science Series, Vol 1).* Academic Press, 2 edition, 2001. 12, 25

[15] Sharp D.H. An overview of rayleigh-taylor instability. *Physica D: Nonlinear Phenomena*, 12(1-3):3 – 10, IN1–IN10, 11–18, 1984. 47, 48

[16] Müller-Plathe F. Parallelising a molecular dynamics algorithm on a multi-processor workstation. *Computer Physics Communications*, 61(3):285 – 293, 1990.

[17] Ahlers G. Experiments with pattern-forming systems. *Physica D: Nonlinear Phenomena*, 51(1-3):421 – 443, 1991. 58

[18] Ciccotti G., Jacucci G., and McDonald I. R. Thought-experiments by molecular dynamics. *Journal of Statistical Physics*, 21(1):1–22, Jul 1979. 58

[19] Hoover W. G. *Smooth Particle Applied Mechanics: The State of the Art,*. World Scientific, 2006. 58

[20] Taylor G. The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. i. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 201(1065):192–196, 1950. 47

[21] Garcia. Hydrodynamic fluctuations and the direct simulation monte carlo method. *Microscopic Simulations of Complex Flows (NATO Asi Series B, Physics)*, 236:177 – 188, 1990. 59

[22] Flynn M. J. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, September 1972. 30

[23] Tritton D. J. *Physical Fluid Dynamics, 2nd ed.* Oxford University Press, 1988. 59

[24] Kadau K., Rosenblatt C., Barber J. L., Germann T. C., Huang Z., Carlès P., and Alder B. J. The importance of fluctuations in fluid mixing. *Proceedings of the National Academy of Sciences*, 104(19):7741–7745, 2007. 48

[25] Kadau K., Germann C. T., Hadjiconstantinou N. G., Lomdahl P. S., Dimonte G., Holian B. L., and Alder B. J. Nanohydrodynamics simulations: An atomistic view of the rayleigh taylor instability. *Proceedings of the National Academy of Sciences of the United States of America*, 101(16):5851–5855, 2004. 48

[26] Lichtenegger K. Application of molecular dynamics simulation methods in the nanoscale regime. 2004. 20, 35

[27] Mugnai M. L., Caprara S., Ciccotti G., Pierleoni C., and Mareschal M. Transient hydrodynamical behavior by dynamical nonequilibrium molecular dynamics: The formation of convective cells. *The Journal of Chemical Physics*, 131(6):064106, 2009. 59

[28] Verlet L. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159(1):98, Jul 1967. 28

[29] Youngs D. L. Numerical simulation of turbulent mixing by rayleigh-taylor instability. *Physica D: Nonlinear Phenomena*, 12(1-3):32 – 44, 1984. 48, 52

[30] Griebel M., Knapek S., Zumbusch G., and Caglar A. *Numerische Simulation in der Moleküldynamik. Numerik, Algorithmen, Parallelisierung, Anwendungen.* Springer, Berlin, Heidelberg, 2003. ix, 12, 14, 16

[31] Mareschal M. and Kestemont E. Molecular-dynamics study of rayleigh-bénard convection. *Nature*, 329:427–429, 1987. 58

[32] Mareschal M. and Kestemont E. Order and fluctuations in nonequilibrium molecular dynamics simulations of two-dimensional fluids. *Journal of Statistical Physics*, 48:1187–1201, 1987. 58, 67

[33] Mareschal M., Malek Mansour M., Puhl A., and Kestemont E. Molecular dynamics versus hydrodynamics in a two-dimensional rayleigh-bénard system. *Phys. Rev. Lett.*, 61(22):2550–2553, Nov 1988. 59

[34] Kum O., Hoover W. G., and Posch H. A. Viscous conducting flows with smooth-particle applied mechanics. *Phys. Rev. E*, 52(5):4899–4908, Nov 1995. 59

[35] Couturier R. and Chipot C. Parallel molecular dynamics using openmp on a shared memory machine. *Computer Physics Communications*, 124(1):49 – 59, 2000. 32

[36] Trobec R., Scaronterk M., Praprotnik M., and Janei D. Implementation and evaluation of mpi-based parallel md program. *International Journal of Quantum Chemistry*, 84(1):23–31, 2001. 32

[37] Rayleigh. Investigation of the character of the equilibrium of an incompressible heavy fluid of variable density. *Proceedings of the London Mathematical Society*, Volume s1-14(1):170–177, 1882. 47

[38] Greenside H. S. and Coughran W. M. Nonlinear pattern formation near the onset of rayleigh-bénard convection. *Phys. Rev. A*, 30(1):398–428, Jul 1984. 59

[39] Stefanov S. and Cercignani C. Monte carlo simulation of benard's instability in a rarefied gas. *European Journal of Mechanics, B/Fluids*, 5:543 – 554, 1992. 59

[40] Shengtai and Hui. Parallel amr code for compressible mhd or hd equations. 2006. ix

[41] Watanabe T. and Kaburaki H. Increase in chaotic motions of atoms in a large-scale self-organized motion. *Phys. Rev. E*, 54(2):1504–1509, Aug 1996. 59

[42] Watanabe T. and Kaburaki H. Particle simulation of three-dimensional convection patterns in a rayleigh-bénard system. *Phys. Rev. E*, 56(1):1218–1221, Jul 1997. 59

[43] Watanabe T., Kaburaki H., Machida M., and Yokokawa M. Growth of long-range correlations in a transition between heat conduction and convection. *Phys. Rev. E*, 52(2):1601–1605, Aug 1995. 59

[44] Watanabe T., Kaburaki H., and Yokokawa M. Simulation of a two-dimensional rayleigh-bénard system using the direct simulation monte carlo method. *Phys. Rev. E*, 49(5):4060–4064, May 1994. 59

[45] Alda W., Dzwinel W., Kitowski J., Mościński J., Pogoda M., and Yuen D. A. Complex fluid-dynamical phenomena modeled by large-scale molecular-dynamics simulations. *Computers in Physics*, 12(6):595–600, 1998. 48

[46] Dzwinel W., Alda W., Pogoda M., and Yuen D. A. Turbulent mixing in the microscale: a 2d molecular dynamics simulation. *Physica D: Nonlinear Phenomena*, 137(1-2):157 – 171, 2000. 48

[47] Hockney R. W. and Eastwood J. W. *Computer simulation using particles.* Taylor & Francis, Inc., Bristol, PA, USA, 1988. 28

[48] Humphrey W., Dalke A., and Schulten K. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996. 40

[49] Mattson W. and Rice B. M. Near-neighbor calculations using a modified cell-linked list method. *Computer Physics Communications*, 119(2-3):135–148, 1999. 29

Acknowledgments

I want to express my gratitude to my supervisor Univ. Prof. Dipl.-Phys. Dr.rer.nat. Wolfgang von der Linden who was abundantly helpful and offered invaluable assistance, support and guidance.
Deepest gratitude are also due to the members of the supervisory committee Prof. Dipl.-Ing. Dr.techn. Winfried Kernbichler and Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Kindl.
Furthermore I would like to thank particularly Dipl.-Ing. Mag. Klaus Lichtenegger, Dipl.-Ing. Klaus Allmaier, Dipl.-Ing. Alexander Blümel and Dipl.-Ing. Thomas Noisternig since this master thesis profitted a lot from our interesting discussions and the many new impulses I received from them.
I am very grateful for the invaluable assistance of Dr. Udo von Toussaint, Prof. Dipl.-Ing. Dr. Gundolf Haase and Dr.-Ing. Jan Treibig.
Special thanks also to Dipl.-Ing. Christian Walchshofer, Dipl.-Ing. Jörg Killmann, Robert Karner, Christoph Schwab and Dipl.-Ing. Ulrike Fluch for their helpful advice.
Not forgetting Tara Dudley, Benjamin Nemeth and Birgit Bukovnik who helped me correcting parts of this master thesis.
I am forever indebted to Sini Rautio for her understanding, endless patience and encouragement when it was most required.
Apart from my colleagues and friends, I would like to express my love and gratitude to my beloved family, for their support and endless love through the duration of my studies.