FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

Simulation of Floating Objects in Free-Surface Flows

Simon Bogner

Diplomarbeit

Simulation of Floating Objects in Free-Surface Flows

Simon Bogner

Diplomarbeit

Aufgabensteller:	Prof. Dr. U. Rüde
Betreuer:	Stefan Donath, M. Sc.
	Dipl.-Inf. Christian Feichtinger
Bearbeitungszeitraum:	21.7.2008 – 21.01.2009

Erklärung:

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 21. Januar 2009 ....

Abstract

In this thesis a model for the simulation of floating objects on free surfaces is presented. Free surfaces arise at the transition between two fluids with a large density difference, e.g. between water and air. This effect is important for a lot of technical applications, such as metal foaming or molding. Fluid flows around obstacles or flows involving solid bodies is a phenomenon, of equally notable importance and has applications ranging from vehicle construction to particle technologies and material sciences.

For the research in these areas, computational models are applied with increased popularity. In the field of computational fluid dynamics the lattice Boltzmann methods have been established, as a powerful way to deal with various problems, including the simulation of free surface flows. Approaches for the incorporation of rigid bodies into a lattice Boltzmann model has also been successfully realized in the past. This thesis proposes a model, which combines both ideas, to simulate objects exposed to a liquid flow with a free surface.

In order to simulate the behaviour of the objects in the flow, the lattice Boltzmann model is coupled with a rigid body physics engine. An external force is incorporated into the model, to simulate systems under the influence of gravity. For the second phase, a gas of negligible influence on the flow is assumed, such that only the liquid phase needs to be simulated. The presented model has been tested and validated in terms of drag forces and buoyant forces on primitive rigid bodies and was successfully applied to selected problems, e.g. the propulsion of a paddle wheel and the floating of concave objects.

Zusammenfassung

Die vorliegende Arbeit präsentiert ein Modell für die Simulation von schwimmenden Körpern auf freien Oberflächen. Freie Oberflächen treten als Phasenübergang zwischen flüssigen Medien mit hohem Dichteunterschied auf, wie z.B. zwischen Wasser und Luft, und spielen außerdem eine wichtige Rolle bei zahlreichen technischen Anwendungen wie Metallschäumen oder Schmelzen. Strömungsvorgänge, an denen solide Hindernisse oder Körper beteiligt sind, beschäftigen ebenfalls seit längerem die Forschung und finden eine breite Anwendung, vom Fahrzeugbau bis hin zur Partikeltechnologie der Werkstoffwissenschaften.

Der Erkenntnisgewinn in diesen Bereichen wird zunehmend mit Hilfe computergestützter numerischer Modelle betrieben. Dabei fanden auf dem Feld der Strömungssimulation in den letzten Dekaden die Lattice-Boltzmann-Methoden vermehrt Anwendung, auch in der Simulation von mehrphasigen Strömungen mit freier Oberfläche. Modelle für die Beteiligung von Festkörpern an Strömungsvorgängen wurden ebenfalls bereits mit Lattice-Boltzmann-Methoden erfolgreich realisiert. Diese Arbeit stellt einen Vorschlag dar, beide Ansätze miteinander zu vereinen, um einfache Objekte, die der Strömung einer Flüssigkeit mit freier Oberfläche ausgesetzt sind, zu simulieren.

Dabei wurde Wert auf die Phänomene des Auftriebs, der Reibungskraft und des Treibens eines Körpers in einer Strömung gelegt. Um das dynamische Verhalten der Körper zu simulieren, wurde das Lattice-Boltzmann Modell mit einer Festkörperphysikengine gekoppelt, an welche die in der Strömungssimulation errechneten Kräfte weitergegeben werden. Auch der Einfluss einer Gravitationskraft, dem das Fluid sowie die Objekte ausgesetzt sind, wurde in das Modell integriert. Das Modell kommt ohne zusätzliche Berechnungen für die zweite Phase aus, welche als Gas angenommen und vernachlässigt werden soll. Der Lösungsansatz wurde anhand bekannter Gesetzmäßigkeiten validiert, und konnte unter anderem erfolgreich auf das Problem des Rückstoßes eines Schaufelrades und des Schwimmens eines konkaven Körpers angewandt werden.

Table of contents

1 Introduction 1

1.1 Structure of this Thesis 1
1.2 General Properties of Fluids and Flows 2
1.3 Forces on Immersed Bodies 3
Hydrostatic Forces 3
Hydrodynamic Forces 3

2 From the Boltzmann-Equation to the Lattice Boltzmann Method 5

2.1 The Boltzmann Equation 5
2.1.1 Derivation 5
2.1.2 Conservation Equations 7
2.1.3 Equilibrium and Maxwell Distribution 7
2.1.4 H-Theorem 8
2.2 BGK-Model 8
2.3 Derivation of the Lattice Boltzmann Method 9
2.3.1 Time Discretization 9
2.3.2 Phase Space Discretization 10
Low Mach Number Expansion 10
Discretization of Velocities and Hydrodynamic Moments 11
Lattice Models 12
2.3.3 The Lattice Boltzmann Equation 12

3 Simulation of Floating Objects 15

3.1 Computational Model 15
3.1.1 Parametrization of the LBM 16
3.1.2 Lattice Initialization 17
3.2 Single Phase Free Surface Model 18
3.2.1 Fluid Advection 18
3.2.2 Cell Conversion at the Interface Layer 19
3.2.3 Free Surface Boundary Condition 19
3.2.4 Atmosphere and Bubbles 20
3.3 Solid Wall Boundary Conditions 20
3.3.1 No-slip, Free-slip and Part-slip 20
3.3.2 Curved Boundaries 21
Sphere Intersection 22
Box Intersection 23
Capsule Intersection 23
3.4 Moving Obstacles 24
3.4.1 Obstacle Mapping 24
3.4.2 Momentum Exchange Method 26

4 Implementation Outline 29

4.1 WaLBerla Software Framework 29
4.2 Modified Sweeps 31
 4.2.1 PDF Sweep 31
 4.2.2 Object Sweep 31
 4.2.3 Force Sweep 31
4.3 Other Improvements 32
  Gravity Initialization 32
  Free Surface Visualization 33

5 Test Cases and Results 35

5.1 Gravity Model 35
  Density Gradient 35
  Breaking Dam 37
5.2 Force Calculation 38
  Drag Force 38
  Buoyant Force 39
  Falling Sphere 40
5.3 Floating Objects 42
  Advection Test 42
  Moving Boat 43
  Buoyancy with a Concave Object 44
5.4 Mass Fluctuation with Floating Objects 45
  Submerging Cubes 45
  Tangential Movement 48

6 Conclusion 51

Bibliography 53

Chapter 1
Introduction

In this thesis a model for the simulation of rigid bodies exposed to the free surface flow of a liquid will be presented. The model will be able to cope with effects of friction, buoyancy and advection of objects in a fluid in three-dimensional scenarios. These effects and the physical background will be described at the end of this chapter. Thereby the fluid dynamics will be simulated via the lattice Boltzmann method (LBM). In the past this method has successfully been applied to a number of various problems e.g. flows through porous media, flow past obstacles, multi-phase flows of mixtures and particle suspensions (for detailed and exhaustive survey of applications, see [Suc01]). The targeted application of this work is a two-phase gas-liquid flow with floating objects. It will be assumed that the flow is governed by the dynamics of the liquid phase, such that the dynamics of the gas can be neglected. Therefor an approach proposed by [KT04] will be used, that was originally developed to simulate metal foams and is related to an work by [GS03]. With this method the problem of the two-phase free surface flow reduces to a single-phase system with a free surface, where the gas is only taken into account by its pressure. To combine the fluid simulation with moving objects we follow the method of [Lad93, Lad07], which allows the calculation of the forces exerted on bodies by the liquid as well as the fluid flow induced at the surface of a body when the object is in motion.

The methods to be presented in this thesis have been implemented in the *waLBerla* software framework [GDF+07a], a lattice Boltzmann based fluid solver, that is being developed at the chair for system simulation (LSS). The rigid body dynamics, i.e. the behaviour of the objects, was thereby handled by the *P.E.* physics engine, also developed at the LSS, that can be attached to the waLBerla framework [GDF+07b, Igl07].

1.1 Structure of this Thesis

This thesis has been arranged in a classical manner, i.e. the theoretic background precedes the practical approaches and results: Chapter 2 presents the mathematical background of the LBM that was used in this thesis and is aimed at readers that are new to the lattice Boltzmann area. The theory behind the continuous Boltzmann equation is briefly explained, including its derivation. Then the lattice Boltzmann equation is derived from the Bhatnagar, Gross and Krook model. Chapter 3 gives a description of the overall computational model, i.e. the extension of the LBM to free surface flows, the lattice initialization, the boundary treatment for fixed and moving surfaces and the momentum exchange method for force calculation. The incorporation of moving objects into the free surface model is explained. Curved boundary treatment, which has been used for some simulations, is also explained in that chapter.

Details regarding the implementation of the described model into waLBerla are given in Chapter 4. In Chapter 5 some selected simulation scenarios are presented and the results are discussed. The force calculation is evaluated and errors in the approximation of the exact obstacle shapes are analyzed. Mass fluctuation has been a general problem with moving obstacles, that is also discussed. Some ideas to improve the presented method in the future and other conclusions can be found in Chapter 6.

In the following, we briefly recite some basic facts, terms and equations that describe the effects, we want to simulate. A more detailed look on fluid mechanical effects can be found, for example, in [ZB08]. Readers coming from a physical background may want to skip the two sections.

1.2 General Properties of Fluids and Flows

In physics fluids are defined as substances that do not resist deformation and will adopt to the form of the containing vessel. Even though fluids consist of molecules that collide with each other, a more comfortable way to describe fluids is by assuming them to be continuous objects. This means that certain physical properties are defined at any point at any time. The most important observables for fluids are:

- density $\rho \quad [\frac{\mathrm{kg}}{m^3}]$
- pressure $P \quad [\mathrm{Pa} = \frac{N}{m^2}]$
- temperature $T \quad [K]$
- velocity $\boldsymbol{u} \quad [\frac{m}{s^2}]$

Further it is assumed that certain quantities of the fluid are conserved in the system, no matter what the specific circumstances are. The most important law of conservation is the *continuity equation* given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\, \boldsymbol{u}) = 0, \tag{1.1}$$

which assures the conservation of mass.

Throughout this thesis only the case of *Newtonian fluids* (such as water, air, oil,..) is considered. A Newtonian fluid is characterized by its viscosity constant η [Pa $s = \frac{N\,s}{m^2}$] (*dynamic viscosity)*, which is a measure of the resistance of the fluid to deformation. The ratio of dynamic viscosity and density is defined as *kinematic viscosity* ν $[\frac{m^2}{s}]$:

$$\nu = \frac{\eta}{\rho} \tag{1.2}$$

Important equations describing the motion of liquids or gases are the *Navier-Stokes equations*. These second-order partial differential equations (for the special case of an incompressible liquid) read

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0, \\ \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} &= -\frac{\nabla P}{\rho} + \frac{\eta}{\rho} \triangle \boldsymbol{u} + \boldsymbol{g}. \end{aligned} \tag{1.3}$$

Here $\boldsymbol{g}$ is an acceleration due to a body force (e.g. gravity).

If only a single homogeneous fluid is considered, one speaks of a *one-phase* model. If two or more different fluids are present, this is called a *two-phase* or *multi-phase* model, respectively. The focus lies on a two-phase model with a liquid phase and a gas phase in the following.

One property that distinguishes *liquids* from *gases* is the ability to form *free surfaces*: Free surfaces arise as a boundary layer between two homogeneous immiscible fluids of different density, for example between a liquid and a gas. Inside of the liquid attracting forces between molecules are averaged out over all directions, whereas for molecules at the free surface the forces pulling inwards prevail and give rise to the effect of *surface tension*.

Another criterion is the *compressibility*: Gases are assumed to adopt to whatever volume is available, whereas the volume of a liquid is assumed to stay constant and only changes its shape.

For a fluid in motion one principally distinguishes two regimes of flow. A flow is *laminar* if no turbulences occur, else it is called a *turbulent flow*. The Reynolds number of a flow is defined as

$$\mathrm{Re} = \frac{\rho\, u\, L}{\eta} = \frac{u\, L}{\nu}, \tag{1.4}$$

where u specifies the characteristic velocity of the flow, and L is the characteristic length (e.g. the diameter of a pipe). This dimensionless quantity is used to characterize the type of a flow. Laminar flow occurs at low Reynolds number. If the Reynolds number is increased over a critical level, the flow will become unstable and turbulent. At high Reynolds numbers turbulent flows are typically observed.

1.3 Forces on Immersed Bodies

Hydrostatic Forces

Whenever a liquid contained in a vessel is exposed to a constant uniform force field, e.g. the gravitational field of earth, it will build up a pressure gradient compensating at every point exactly the weight of fluid mass inside a column from that point upwards to the free surface. Provided that the fluid is at rest, there will be a time-constant pressure and density gradient in the fluid in line with the force field. This state is called the *hydrostatic equilibrium.*

If the fluid can be assumed as nearly *incompressible* ($\rho = \mathrm{const}$), the *hydrostatic pressure* can be expressed as a linear function

$$P(z) = \rho\, g\, z + P_0, \tag{1.5}$$

where g $[\frac{m}{s^2}]$ is the gravitational constant, z $[m]$ is the depth relative to the free surface and P_0 $[\mathrm{Pa} = \frac{N}{m^2}]$ is the atmospheric pressure at the free surface.

If a body is submerged into the liquid, the surrounding pressure will exert forces on the body surface. Since pressure is defined as force per area, the total force due to hydrostatic pressure is given by the surface integral

$$\boldsymbol{F} = -\int_S P(\boldsymbol{x})\, \boldsymbol{n}(\boldsymbol{x}) dS, \tag{1.6}$$

where S defines the surface of the body and $\boldsymbol{n}(\boldsymbol{x})$ is surface normal in the point $\boldsymbol{x}$. By Gauss' Theorem the *Archimedean law* can be derived from (1.6), which states

$$\boldsymbol{F} = -\,\boldsymbol{g}\, \rho\, V, \tag{1.7}$$

where $\boldsymbol{g}$ is the directed gravitational strength. Thus the buoyant force on a (partially) immersed object equals the weight of the fluid displaced by the object, and is directed opposite to gravity. If the density of the object is greater than the density of the liquid, it will sink. If its density is lower, it will start to rise and float at the surface.

Hydrodynamic Forces

If fluid motion occurs relative to the object hydrodynamic forces and stresses arise. The forces exerted on the obstacle are usually very hard to determine. For that reason, only the special case of *Stokes flow* around a sphere is considered here. For small Reynolds numbers ($\mathrm{Re} \ll 1$), the *drag force* on a spherical particle of radius r moving with velocity U is given by *Stokes' law:*

$$F_D = -\,6\, \pi\, \eta\, r\, U \tag{1.8}$$

The negative sign indicates, that F_D is directed opposite to U.

So, if a sphere of density ρ_s is falling through a fluid with density ρ accelerated by gravity its drag force will increase accordingly until finally its buoyant force combined with the drag force balances the gravitational force and the velocity stays constant. This terminal velocity is given by

$$U_{\text{terminal}} = \frac{2}{9} \frac{(\rho_s - \rho)\, r^2}{\eta}\, g\,. \tag{1.9}$$

Note that this principle also works for spheres of lower density $\rho_s < \rho$. Then the buoyant force is equalled by the sum of drag force and gravitational force, since the velocity is directed upwards in this case.

Chapter 2

From the Boltzmann-Equation to the Lattice Boltzmann Method

In this chapter the *Lattice Boltzmann Method* (LBM) is derived from the Boltzmann equation, which is a classical result from statistical mechanics and forms the theoretical ground for the LBM. The Boltzmann equation originates as a description of the distribution of molecules (particles) of a gas. Its solutions describe the approach of a closed system of particles towards thermodynamic equilibrium. This background of the LBM is different from other computational fluid dynamics approaches, which are based directly on the continuous flow equations (Sec. 2.1).

By replacing the collision integral of the Boltzmann equation with a simplified collision operator, the *BGK equation* is obtained (Sec. 2.2).

From this equation the lattice Boltzmann algorithm can be deduced by a discretization of phase space. For a system near equilibrium the Navier-Stokes equations of fluid flow can be derived from either the continuous Boltzmann equation, as well as from the lattice Boltzmann equation, which is the core of the LBM - thus proving its validity for the description of fluid flow (Sec. 2.3).

2.1 The Boltzmann Equation

2.1.1 Derivation

In the field of statistical mechanics, macroscopic processes such as the flow of a gas or liquid, are derived from microscopic models by methods of probability theory. Let our model be a gas consisting of a population of $N \gg 1$ uniform particles of mass m. We introduce a 6-dimensional space $X \times V$ called *phase space*, such that every particle of the system is given by its position $\boldsymbol{x} \in X$, and Velocity $\boldsymbol{v} \in V$ in the phase space. We now define f to be the distribution function of the particle population, such that $f(\boldsymbol{x}, \boldsymbol{v}, t)\, d\boldsymbol{x}d\boldsymbol{v}$ equals the number of particles at position $\boldsymbol{x} \in X, \boldsymbol{v} \in V$ at time t in a volume element $d\boldsymbol{x}d\boldsymbol{v}$ around $(\boldsymbol{x}, \boldsymbol{v})$ in phase space.

Macroscopic quantities can now be defined as moments of the particle distribution function f. Obviously

$$\int f(\boldsymbol{x}, \boldsymbol{v}, t)\, d\boldsymbol{x}d\boldsymbol{v} = \mathrm{N}. \tag{2.1}$$

Other important macroscopic observables are

$$\begin{aligned}
\text{particle density} \quad n(\boldsymbol{x}, t) &= \int f(\boldsymbol{x}, \boldsymbol{v}, t)\, d\boldsymbol{v}, && (2.2)\\
\text{mass density} \quad \rho(\boldsymbol{x}, t) &= m\, n(\boldsymbol{x}, t) = \int m\, f(\boldsymbol{x}, \boldsymbol{v}, t)\, d\boldsymbol{v} && (2.3)\\
\text{macroscopic velocity} \quad u(\boldsymbol{x}, t) &= \frac{1}{n(\boldsymbol{x}, t)} \int \boldsymbol{v}\, f(\boldsymbol{x}, \boldsymbol{v}, t)\, d\boldsymbol{v}, && (2.4)\\
\text{momentum} \quad \rho u(\boldsymbol{x}, t) &= \int m\, \boldsymbol{v}\, f(\boldsymbol{x}, \boldsymbol{v}, t)\, d\boldsymbol{v}, && (2.5)\\
\text{energy density} \quad E(\boldsymbol{x}, t) &= \int \frac{1}{2} m\, \boldsymbol{v}^2\, f(\boldsymbol{x}, \boldsymbol{v}, t)\, d\boldsymbol{v}. && (2.6)
\end{aligned}$$

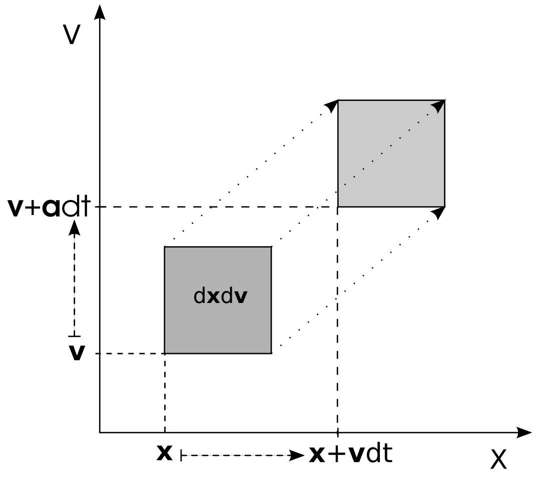


Figure 2.1. Evolution of a volume element $\boldsymbol{dxdv}$ around $(\boldsymbol{x},\boldsymbol{v})$ from time t to $t+\mathrm{dt}$.

We are now going to derive the Boltzmann equation by inspecting how the number of particles at time t in a volume element $\boldsymbol{dxdv}$ around $\boldsymbol{x}$ and $\boldsymbol{v}$ in phase space evolves during a time step dt (see Fig. 2.1). This is the standard derivation of the equation and can be found for example in [Sch99] or [Hän04]. In presence of an *external force* $\boldsymbol{F}$ (e.g. gravity), the whole population of molecules is accelerated by $\boldsymbol{a}=\frac{\boldsymbol{F}}{m}$. At time $t+\mathrm{dt}$ the number of particles in the volume element $\boldsymbol{dxdv}$ around $(\boldsymbol{x}+\boldsymbol{v}\,\mathrm{dt}, \boldsymbol{v}+\boldsymbol{a}\,\mathrm{dt})$ must equal the number of particles from the original volume up to the amount of particles driven in or out of the volume element because of collisions between particles.

$$f(\boldsymbol{x}+\boldsymbol{v}\,\mathrm{dt},\boldsymbol{v}+\boldsymbol{a}\,\mathrm{dt},t+\mathrm{dt})\,\boldsymbol{dxdv}=f(\boldsymbol{x},\boldsymbol{v},t)\,\boldsymbol{dxdv}+\left.\frac{\partial f}{\partial t}\right|_{\text{Collision}}\boldsymbol{dxdv}. \tag{2.7}$$

We divide the equation by $\boldsymbol{dxdv}$ and expand the left hand side as $f(\boldsymbol{x}+\boldsymbol{v}\,\mathrm{dt},\boldsymbol{v}+\boldsymbol{a}\,\mathrm{dt},t+\mathrm{dt})=f(\boldsymbol{x},\boldsymbol{v},t)+\mathrm{dt}\,\boldsymbol{v}\frac{\partial f}{\partial \boldsymbol{x}}+\mathrm{dt}\,\boldsymbol{a}\frac{\partial f}{\partial \boldsymbol{v}}+\mathrm{dt}\frac{\partial f}{\partial t}$ and obtain after a division by dt

$$\frac{\partial}{\partial t}f(\boldsymbol{x},\boldsymbol{v},t)+\boldsymbol{v}\frac{\partial}{\partial \boldsymbol{x}}f(\boldsymbol{x},\boldsymbol{v},t)+\boldsymbol{a}\frac{\partial}{\partial \boldsymbol{v}}f(\boldsymbol{x},\boldsymbol{v},t)=\left.\frac{\partial f}{\partial t}\right|_{\text{Collision}}, \tag{2.8}$$

which is the Boltzmann equation with undetermined collision term.

For the original Boltzmann equation the collision term $\left.\frac{\partial f}{\partial t}\right|_{\text{Collision}}$ is substituted by

$$\left.\frac{\partial f}{\partial t}\right|_{\text{Collision}}=\int W(\boldsymbol{v},\boldsymbol{v}_0;\boldsymbol{v}',\boldsymbol{v}_0')\Big[f(\boldsymbol{x},\boldsymbol{v}',t)\,f(\boldsymbol{x},\boldsymbol{v}_0',t)-f(\boldsymbol{x},\boldsymbol{v},t)\,f(\boldsymbol{x},\boldsymbol{v}_0,t)\Big]\,\boldsymbol{dv}_0\boldsymbol{dv}'\boldsymbol{dv}_0'. \tag{2.9}$$

This is known as Boltzmann's Stoßzahlansatz, where change in the particle balance due to collisions is modelled by elastic two-particle collisions. $W(\boldsymbol{v}_0,\,\boldsymbol{v}_1;\,\boldsymbol{v}_0',\,\boldsymbol{v}_1')=W(\boldsymbol{v}_0',\,\boldsymbol{v}_1';\,\boldsymbol{v}_0,\,\boldsymbol{v}_1)$ is the probability that two particles with velocities $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ end up with velocities $\boldsymbol{v}_0'$ and $\boldsymbol{v}_1'$ after a collision. Hence the number of particles streamed out of or into the volume is proportional to $W(\boldsymbol{v},\,\boldsymbol{v}_0;\,\boldsymbol{v}',\,\boldsymbol{v}_0')\;f(\boldsymbol{x},\,\boldsymbol{v},\,t)\;f(\boldsymbol{x},\,\boldsymbol{v}_0,\,t)$ and $W(\boldsymbol{v},\,\boldsymbol{v}_0;\,\boldsymbol{v}',\,\boldsymbol{v}_0')\;f(\boldsymbol{x},\,\boldsymbol{v}',\,t)\;f(\boldsymbol{x},\,\boldsymbol{v}_0',\,t)$, respectively. See [Sch99] for more details.

Some assumptions not explicitly stated have been made in this derivation [Hän04]: A *rarefied gas*, i.e. a *mean free path*[2.1] a lot larger than the molecule dimensions, was assumed with negligible small intermolecular forces, such that the path of a particle does change only at collisions or because of external forces. The collisions are assumed to be relatively short, such that the distribution function stays constant during collisions. The probability for collisions with more than two particles involved is assumed to be negligibly small.

2.1. The mean free path *is the average distance covered by particles between collisions.*

2.1.2 Conservation Equations

As outlined above, while originating from a lower physical level, the Boltzmann equation can be used to describe macroscopic processes, with macroscopic quantities obtained as moments of the particle distribution function as given by Eq. (2.2) - Eq. (2.6). Important conservation equations for mass, momentum and energy, can be obtained directly as moments of the Boltzmann equation. Since mass, momentum and energy are invariant to elastic collisions, the right hand side vanishes [Sch99].

One obtains as first moment the *continuity equation*

$$\begin{aligned}\int m\frac{\partial}{\partial t}f\,d\boldsymbol{v}+\int m\,\boldsymbol{v}\,\frac{\partial}{\partial \boldsymbol{x}}f d\boldsymbol{v}+\int m\,\boldsymbol{a}\,\frac{\partial}{\partial \boldsymbol{v}}f\,d\boldsymbol{v} &= 0,\\ \frac{\partial}{\partial t}\rho+\nabla\cdot(\rho\,\boldsymbol{u}) &= 0.\end{aligned}$$

The conservation equation for momentum and energy follow analogously from the general identity

$$\int \Phi\,[\frac{\partial}{\partial t}+\boldsymbol{v}\,\frac{\partial}{\partial \boldsymbol{x}}+\boldsymbol{a}\,\frac{\partial}{\partial \boldsymbol{v}}]\,f(\boldsymbol{x},\boldsymbol{v},t)=0,$$

with $\Phi=m\boldsymbol{v}$ and $\Phi=\frac{m}{2}\boldsymbol{v}^2$, respectively.

2.1.3 Equilibrium and Maxwell Distribution

Analytical solutions for the Boltzmann equation are very complex, but for the special case of thermodynamic equilibrium the *Maxwell distribution* can be shown to satisfy the equation. According to [Hän04] this corresponds to a system in the continuous limit with a *Knudsen number*[2.2]

$$\mathrm{Kn}=\frac{l_f}{L}\rightarrow 0,$$

where l_f is the mean free path of the molecules and L is the characteristic length scale.

Equilibrium is given if the collision term vanishes, i.e. if the loss of particles due to collisions exactly equals the gain of particles.

$$f(\boldsymbol{x},\boldsymbol{v}',t)\,f(\boldsymbol{x},\boldsymbol{v}_0',t)-f(\boldsymbol{x},\boldsymbol{v},t)\,f(\boldsymbol{x},\boldsymbol{v}_0,t)=0 \tag{2.10}$$

$$\ln f(\boldsymbol{x},\boldsymbol{v}',t)+\ln f(\boldsymbol{x},\boldsymbol{v}_0',t)=\ln f(\boldsymbol{x},\boldsymbol{v},t)+\ln f(\boldsymbol{x},\boldsymbol{v}_0,t)$$

The logarithmized form above is similar to the following statements of conserved mass, momentum and energy for elastic collisions,

$$\begin{aligned} m+m_0 &= m'+m_0'\\ m\,\boldsymbol{c}+m_0\,\boldsymbol{c}_0 &= m'\,\boldsymbol{c}'+m_0'\,\boldsymbol{c}_0'\\ \frac{m}{2}\boldsymbol{c}^2+\frac{m_0}{2}\boldsymbol{c}_0^2 &= \frac{m'}{2}\boldsymbol{c}'^2+\frac{m_0'}{2}\boldsymbol{c}_0'^2\end{aligned}$$

where $\boldsymbol{c}=\boldsymbol{v}-\boldsymbol{u}$ is the *peculiar speed* of a particle. This gives an ansatz for the solution $f=f_{\mathrm{eq}}$ of Eq. (2.8) in the case of equilibrium as a combination of the latter quantities.

$$\ln f_{\mathrm{eq}}=A\,m+B\,m\,\boldsymbol{c}+C\,\frac{m}{2}\boldsymbol{c}^2,$$

2.2. The Knudsen number is useful for the characterization of a flow of molecules. If it is close to or grater than one, then the continuum assumption does no longer apply.

or

$$f_{\text{eq}} = e^{Am} + e^{Bm\boldsymbol{c}} + e^{C\frac{m}{2}\boldsymbol{c}^2}.$$

Using the conserved quantities density, velocity and energy, A, B and C in the above equation can be determined. The final result is

$$f_{\text{eq}}(\boldsymbol{v}) = n(\boldsymbol{x},t)\sqrt{\frac{m}{2\pi kT(\boldsymbol{x},t)}}^{3} e^{-\frac{m}{2kT(\boldsymbol{x},t)}(\boldsymbol{v}-\boldsymbol{u}(\boldsymbol{x},t))^2} \tag{2.11}$$

which is the Maxwell distribution function. Here $T(\boldsymbol{x},t)$ is the local temperature and $k = 1.38 \cdot 10^{-23}\frac{\text{Nm}}{K}$ is the Boltzmann constant.

2.1.4 H-Theorem

The Maxwell distribution is of particular importance, as it was proven by Boltzmann, that for a homogeneous closed system, every arbitrary non-equilibrium distribution will tend towards such an equilibrium for $t \rightarrow \infty$ (Boltzmann's *H-Theorem*). We will see later that the lattice Boltzmann method is basically a finite difference scheme converging towards equilibrium. The function

$$H(t) = -\int f \ln f d\boldsymbol{x}d\boldsymbol{v}$$

satisfies $H \leqslant 0$ and $\frac{\partial H}{\partial t} \leqslant 0$ and can be used to describe the entropy of the system in accordance with the second law of thermodynamics. The system will stop evolving only if the entropy has reached its maximum, which is possible if and only if $f(\boldsymbol{x}, \boldsymbol{v}', t)\, f(\boldsymbol{x}, \boldsymbol{v}_0', t) - f(\boldsymbol{x}, \boldsymbol{v}, t)\, f(\boldsymbol{x}, \boldsymbol{v}_0, t) = 0$ (Equilibrium). The H-Theorem is one of the most important statements around the Boltzmann equation as it shows the equation's validity for so-called *irreversible processes*, e.g. diffusion of suspensions or flow due to a pressure gradient until an equilibrium state is reached.

2.2 BGK-Model

The Bhatnagar, Gross and Krook (BGK) equation is basically the Boltzmann equation with a simplified collision term $\frac{\partial f}{\partial t}\Big|_{\text{Collision}} = -\frac{1}{\lambda}(f - f_{\text{eq}})$, referred to as the *BGK operator*.

$$\frac{\partial}{\partial t}f + \boldsymbol{v}\frac{\partial}{\partial \boldsymbol{x}}f + \boldsymbol{a}\frac{\partial}{\partial \boldsymbol{v}}f = -\frac{1}{\lambda}(f - f_{\text{eq}}) \tag{2.12}$$

The relaxation parameter $-\frac{1}{\lambda}$ is the *collision frequency* ([Hän04]). Motivated by the H-Theorem collisions are expressed as a linear relaxation towards the local equilibrium distribution, i.e. f_{eq} is a Maxwell distribution function as defined above with parameters set to the local density, velocity and temperature.

We will derive the lattice Boltzmann method from this model in the next section. Therefore it is important to mention, that all crucial properties of the Boltzmann equation stated in Sec. 2.1 are inherited: The H-Theorem can be formulated equivalently for the BGK equation. And obviously mass, momentum and energy are invariants for the BGK operator, because of the nature of f_{eq} as described above.

2.3 Derivation of the Lattice Boltzmann Method

In this section we are going to discretize the BGK model to obtain the lattice Boltzmann equation. The first step will be time discretization (Sec. 2.3.1), followed by the discretization of phase space, which depends on the desired lattice model and needs a suitable discretized approximation (*low mach number expansion*) of the equilibrium function and the force term (Sec. 2.3.2). Finally, the most important properties of the lattice Boltzmann equation will be stated (Sec. 2.3.3).

2.3.1 Time Discretization

The idea illustrated in the following was taken from [HL97] and [Luo98]. To obtain a time discrete Boltzmann equation the BGK equation is integrated over a time interval δ_t. This is done by applying the method of characteristics for partial differential equations (see for example [GL88]).

For the BGK equation (Sec. 2.2) the characteristic system reads

$$\begin{aligned}
\frac{dt}{ds} &= 1, && (2.13)\\
\frac{d\boldsymbol{x}}{ds} &= \boldsymbol{v}, && (2.14)\\
\frac{d\boldsymbol{v}}{ds} &= \boldsymbol{a}, && (2.15)\\
\frac{df}{ds} &= \frac{\partial f}{\partial t}(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s)) + \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s))\,\boldsymbol{v} + \frac{\partial f}{\partial \boldsymbol{v}}(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s))\,\boldsymbol{a} && (2.16)\\
&= -\frac{1}{\lambda}\Big(f(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s)) - f_{\mathrm{eq}}(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s)) \Big).
\end{aligned}$$

Here a new symbolic variable s was introduced, which will be convenient for the integration of Eq. (2.16). From Eq.s (2.13) - (2.15) follows at once

$$\begin{aligned}
t(s) &= t+s,\\
\boldsymbol{x}(s) &= \boldsymbol{x}+\boldsymbol{v}\,s,\\
\boldsymbol{v}(s) &= \boldsymbol{v}+\boldsymbol{a}\,s.
\end{aligned}$$

Note that here t, $\boldsymbol{x}$ and $\boldsymbol{v}$ on the right hand sides are treated as constants. Now Eq. (2.16) can be integrated over the interval $[0,\,\delta_t]$, by using $e^{\frac{1}{\lambda}s}$ as integrating factor, which is multiplied to the equation first:

$$\begin{aligned}
e^{\frac{1}{\lambda}s}\cdot\frac{d}{ds}f(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s)) + \frac{1}{\lambda}e^{\frac{1}{\lambda}s}\cdot f(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s)) &= \frac{1}{\lambda}e^{\frac{1}{\lambda}s}\cdot f_{\mathrm{eq}}(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s)) \qquad \Big|\int_0^{\delta_t} ds\\
\Big[e^{\frac{1}{\lambda}s} f(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s)) \Big]_0^{\delta_t} &= \frac{1}{\lambda}\int_0^{\delta_t} e^{\frac{1}{\lambda}s}\cdot f_{\mathrm{eq}}(\boldsymbol{x}(s),\boldsymbol{v}(s),t(s))\,ds
\end{aligned}$$

When solved for $f(\boldsymbol{x}+\boldsymbol{v}\,\delta_t,\boldsymbol{v}+\boldsymbol{a}\,\delta_t,t+\delta_t)$, one has

$$f(\boldsymbol{x}+\boldsymbol{v}\,\delta_t,\boldsymbol{v}+\boldsymbol{a}\,\delta_t,t+\delta_t) = e^{-\frac{1}{\lambda}\delta_t}f(\boldsymbol{x},\boldsymbol{v},t) + \frac{1}{\lambda}e^{-\frac{1}{\lambda}\delta_t}\int_0^{\delta_t} e^{\frac{1}{\lambda}s} f_{\mathrm{eq}}(\boldsymbol{x}+\boldsymbol{v}\,s,\boldsymbol{v}+\boldsymbol{a}\,s,t+s)\,ds.$$

The following steps can be found more detailed in [Thü02]. To eliminate the integral on the right hand side, $f_{\mathrm{eq}}(t+s) = f_{\mathrm{eq}}(\boldsymbol{x}+\boldsymbol{v}\,s,\boldsymbol{v}+\boldsymbol{a}\,s,t+s)$ [2.3] is approximated as

$$f_{\mathrm{eq}}(t+s) = (1-\frac{s}{\delta_t})\,f_{\mathrm{eq}}(t) + \frac{s}{\delta_t}\,f_{\mathrm{eq}}\,(t+\delta_t) + \mathcal{O}(\delta_t^2),$$

2.3. $\boldsymbol{x}$ - and $\boldsymbol{v}$ - parameters omitted for briefer notation with $0 \leqslant s \leqslant \delta_t$.

This gives an integral-free equation for $f(t+\delta_t) = f(\boldsymbol{x} + \boldsymbol{v}\,\delta_t, \boldsymbol{v} + \boldsymbol{a}\,\delta_t, t + \delta_t)$:

$$f(t+\delta_t) = f(t) + (e^{-\frac{1}{\lambda}\delta_t} - 1)\,[f(t) - f_{\mathrm{eq}}(t)] + \left(1 + \frac{\lambda}{\delta_t}(e^{-\frac{1}{\lambda}\delta_t} - 1)\right)[f_{\mathrm{eq}}(t+\delta_t) - f_{\mathrm{eq}}(t)]$$

We now expand the exponential function in δ_t as

$$e^{-\frac{1}{\lambda}\delta_t} = 1 - \frac{\delta_t}{\lambda} + \mathcal{O}(\delta_t^2)$$

and assume f to be smooth enough to finally obtain

$$f(\boldsymbol{x} + \boldsymbol{v}\,\delta_t, \boldsymbol{v}, t + \delta_t) + \boldsymbol{a}\frac{\partial f}{\partial \boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{v}, t) \cdot \delta_t = f(\boldsymbol{x}, \boldsymbol{v}, t) - \frac{1}{\tau}[f(\boldsymbol{x}, \boldsymbol{v}, t) - f_{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{v}, t)], \tag{2.17}$$

where $\tau := \frac{\lambda}{\delta_t}$ is the *dimensionless relaxation time*.

Note that the equilibrium function f_{eq} is not directly a function of time, but implicitly depends on it by the time-depending macroscopic quantities density, velocity and temperature.

2.3.2 Phase Space Discretization

Low Mach Number Expansion

Before we can formulate the time- and space-discrete Lattice Boltzmann equation, we need an appropriate approximation of the equilibrium distribution, Eq. (2.11), as it still shows in the time-discrete equation, Eq. (2.17).

For convenience, in Lattice Boltzmann models, one usually defines f to be a distribution of mass rather then the number of particles, which yields multiplication of our previous equations by particle mass m, but we will stick to the current notation and will simply reformulate the moments later on.

We define the *normalized temperature* as $\theta := \frac{k\,T}{m}$. Then the equilibrium distribution function (of mass) is

$$f_{\mathrm{eq}} = \rho(\boldsymbol{x}, t)\,\sqrt{\frac{1}{2\,\pi\,\theta}}^{3}\,e^{-\frac{(\boldsymbol{v}-\boldsymbol{u})^2}{2\theta}}. \tag{2.18}$$

A Taylor expansion in $\boldsymbol{u}$ leads to (see [Thü02], original in [HL97])

$$f_{\mathrm{eq}} = \rho\,\sqrt{\frac{1}{2\,\pi\,\theta}}^{3}\,e^{-\frac{1}{2\theta}\boldsymbol{v}^2} \cdot \left(1 + \frac{\boldsymbol{v}\,\boldsymbol{u}}{\theta} + \frac{(\boldsymbol{v}\,\boldsymbol{u})^2}{2\,\theta^2} - \frac{\boldsymbol{u}^2}{2\,\theta}\right) + \mathcal{O}(\boldsymbol{u}^3)$$

and we define our approximated function $\widetilde{f_{\mathrm{eq}}}$ as

$$\widetilde{f_{\mathrm{eq}}} := \rho\,\sqrt{\frac{1}{2\,\pi\,\theta}}^{3}\,e^{-\frac{1}{2\theta}\boldsymbol{v}^2} \cdot \left(1 + \frac{\boldsymbol{v}\,\boldsymbol{u}}{\theta} + \frac{(\boldsymbol{v}\,\boldsymbol{u})^2}{2\,\theta^2} - \frac{\boldsymbol{u}^2}{2\,\theta}\right). \tag{2.19}$$

This approximation requires $\boldsymbol{u}$ to be small. This will have to be considered in the final algorithm: $c_s = \sqrt{\frac{k}{m}T} = \sqrt{\theta}$ is called the *speed of sound* for the model and therefor $\widetilde{f_{\mathrm{eq}}}$ is also called the *low mach number expansion* of f_{eq}, since small mach numbers

$$\mathrm{Ma} = \frac{|\boldsymbol{u}|}{c_s} \ll 1 \tag{2.20}$$

are a criterion for the accuracy of the method.

Another term needed for the Lattice Boltzmann equation is the force term $\boldsymbol{a}\,\frac{\partial f}{\partial \boldsymbol{v}}$ from Eq. (2.17). According to [Luo98] this term must be discretized such that the following constraints hold:

$$\begin{aligned}
\int \boldsymbol{a}\,\frac{\partial f}{\partial \boldsymbol{v}}\,d\boldsymbol{v} &= 0 &(2.21)\\
\int \boldsymbol{v}\,\boldsymbol{a}\,\frac{\partial f}{\partial \boldsymbol{v}}\,d\boldsymbol{v} &= -\rho\,\boldsymbol{a} &(2.22)\\
\int v_i v_j\,\boldsymbol{a}\,\frac{\partial f}{\partial \boldsymbol{v}}\,d\boldsymbol{v} &= -\rho(a_i\,u_j + a_j\,v_i) &(2.23)
\end{aligned}$$

This is fulfilled by

$$\boldsymbol{a}\,\frac{\partial f}{\partial \boldsymbol{v}} = -\rho\sqrt{\frac{1}{2\pi\theta}}^{\,3}\,e^{-\frac{v^2}{2\theta}}\cdot\left(\frac{\boldsymbol{v}-\boldsymbol{u}}{\theta}+\frac{\boldsymbol{v}\,\boldsymbol{u}}{\theta^2}\,\boldsymbol{v}\right)\boldsymbol{a}+\mathcal{O}(\boldsymbol{u}^2)+\mathcal{O}(\boldsymbol{v}^3). \tag{2.24}$$

Note that there are various other ways to include a constant body force in the lattice Boltzmann model. One possibility is to modify the macroscopic velocity $\boldsymbol{u}$ in the equilibrium function by the acceleration $\boldsymbol{a} = \frac{\boldsymbol{F}}{m}$, i.e. the system is relaxed towards a modified equilibrium. A survey of various models can be found in [BG00].

The approximated equilibrium function and force term will now be used to set up a lattice model with discrete positions and velocities for particles.

Discretization of Velocities and Hydrodynamic Moments

When the $\boldsymbol{v}$-space is discretized to a finite set of N velocity vectors $\boldsymbol{v}_i$, $i = 0, \ldots, N-1$, the distribution function is split up accordingly into a set of N directed particle distribution functions $f_i := \tilde{w_i}\, f(\boldsymbol{x}, \boldsymbol{v}, t)$ in a way that preserves the *hydrodynamic moments* of the system (density, momentum and inner energy), resulting in the following constraints

$$\begin{aligned}
\rho(\boldsymbol{x},t) &= \int f(\boldsymbol{x},\boldsymbol{v},t)\,d\boldsymbol{v} &&= \sum_{i=0}^{N}\tilde{w_i}\,f(\boldsymbol{x},\boldsymbol{v}_i,t) &&= \sum_{i=0}^{N} f_i(\boldsymbol{x},t) &(2.25)\\
\rho\,\boldsymbol{u}(\boldsymbol{x},t) &= \int \boldsymbol{v}\,f(\boldsymbol{x},\boldsymbol{v},t)\,d\boldsymbol{v} &&= \sum_{i=0}^{N}\tilde{w_i}\,\boldsymbol{v}_i\,f(\boldsymbol{x},\boldsymbol{v}_i,t) &&= \sum_{i=0}^{N}\boldsymbol{v}_i\,f_i(\boldsymbol{x},t) &(2.26)\\
\rho\frac{D}{2}\theta &= \frac{1}{2}\int(\boldsymbol{v}-\boldsymbol{u})^2 f(\boldsymbol{x},\boldsymbol{v},t)\,d\boldsymbol{v} &&= \frac{1}{2}\sum_{i=0}^{N}\tilde{w_i}(\boldsymbol{v}_i-\boldsymbol{u})^2 f(\boldsymbol{x},\boldsymbol{v}_i,t) &&= \frac{1}{2}\sum_{i=0}^{N}(\boldsymbol{v}_i-\boldsymbol{u})^2 f_i(\boldsymbol{x},t), &(2.27)
\end{aligned}$$

where D is the number of dimensions of the system.

The weights $\tilde{w_i}$ have to be determined, such that the above equations are satisfied. This is achieved by quadrature: The weights $\tilde{w_i}$ can then be used to define the distribution functions $f_i(\boldsymbol{x},t)$.

The set of velocities $\boldsymbol{v}_i$ has to span a lattice of discrete positions in space. A concrete example is given by the D2Q9 model in Fig. 2.2, and the D3Q19 model presented in Sec. 3.1, Fig. 3.1. For an isothermal model, where effects of temperature are excluded, energy does not necessarily need to be included as a conserved quantity. Also note that in presence of an external force $\boldsymbol{F} = \boldsymbol{a}\,m$, one might want to reformulate the third moment in order to include the location dependent potential accordingly [Luo00].

We will not carry out the calculation of the weights here, but will briefly state the results in the following paragraph. The calculations can be found in literature: See for example [Luo00] for the D2Q9 model, or [Hän04] for the D3Q15.

Lattice Models

$$\boldsymbol{v}_i = c\,\boldsymbol{e}_i,\ \text{with } \boldsymbol{e}_i = \begin{cases} (0,0) \text{ for } i=0 \\ (\pm 1, 0) \text{ for } i=1...4 \\ (\pm 1, \pm 1) \text{ for } i=5...8 \end{cases}$$

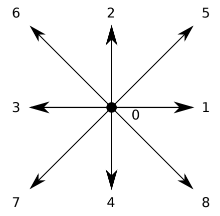


Figure 2.2. The D2Q9 lattice model

The D2Q9 model consists of a two-dimensional (D2) lattice with a set of 9 velocities (Q9), as depicted in Fig. 2.2. The equilibrium distribution discretized as described in the last paragraph reads

$$f_{\text{eq},i} = w_i\,\rho\left(1 + \frac{\boldsymbol{v}_i\boldsymbol{u}}{\theta} + \frac{(\boldsymbol{v}_i\,\boldsymbol{u})^2}{2\,\theta^2} - \frac{\boldsymbol{u}^2}{2\,\theta}\right),\quad w_i = \begin{cases} \frac{4}{9} \text{ for } i=0 \\ \frac{1}{9} \text{ for } i=1...4 \\ \frac{1}{36} \text{ for } i=5...8 \end{cases} \tag{2.28}$$

while the discretized body force term is

$$F_i = w_i\,\rho\left(\frac{\boldsymbol{v}_i - \boldsymbol{u}}{\theta} + \frac{\boldsymbol{v}_i\,\boldsymbol{u}}{2\,\theta^2}\right)\boldsymbol{a}. \tag{2.29}$$

For three dimensions there exist the models D3Q15, D3Q19 and D3Q27. For the second part of this thesis the 3-dimensional $D3Q19$ model with 19 velocities will be used, which uses the same discretized formulae for equilibrium function and force term given by Eq. (2.28) and Eq. (2.29). Only the weights have to be adjusted accordingly. The lattice Boltzmann equation for these models and its properties are given in the following section.

If an isothermal model is assumed, and we will do so for our further considerations, θ is a constant

$$\theta = \frac{1}{3}\,c^2 = c_s^2 \tag{2.30}$$

(See [Luo00]). Also, the parameter c will be normalized to $c=1$, $c_s = \frac{1}{\sqrt{3}}$ in the final model.

For a time step δ_t one then has $\delta_x = c \cdot \delta_t$ as distance along coordinate axes between two neighbouring nodes in the lattice. This distance can be regarded as the mean free path for the lattice model, since collisions will only be treated in the cell centers.

2.3.3 The Lattice Boltzmann Equation

We are now ready to formulate the lattice Boltzmann equation from the time-discrete equation, Eq. (2.17), including the discretized body force term F_i and equilibrium function $f_{\text{eq},i}$ the lattice Boltzmann equation reads

$$f_i(\boldsymbol{x} + \boldsymbol{v}_i\,\delta_t, t + \delta_t) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}\Big(f_i(\boldsymbol{x}, t) - f_{\text{eq},i}(\boldsymbol{x}, t)\Big) - \delta_t\,F_i. \tag{2.31}$$

Usually the time step is normalized to $\delta_t = 1$ and the equation is split up into two steps for advection and collision, respectively:

$$f_i'(\boldsymbol{x} + \boldsymbol{v}_i, t+1) = f_i(\boldsymbol{x}, t) \qquad \textit{(stream step)} \tag{2.32}$$

$$f_i(\boldsymbol{x} + \boldsymbol{v}_i, t+1) = f_i'(\boldsymbol{x} + \boldsymbol{v}_i, t+1) - \frac{1}{\tau}\Big(f_i'(\boldsymbol{x} + \boldsymbol{v}_i, t+1) - f_{\text{eq},i}(\boldsymbol{x} + \boldsymbol{v}_i, t+1)\Big) - F_i \qquad \textit{(collide step)} \tag{2.33}$$

This calculation scheme is known as lattice Boltzmann method. The distribution functions $f_i(\boldsymbol{x},\, t + \delta_t)$ of the next time step for a cell $\boldsymbol{x}$ are obtained from the local neighbours in the stream step, and then relaxed towards the local equilibrium in the collide step:

```
for all cells do
    initialize;

for all timesteps do
    for all cells do stream step;
    for all cells do collide step;
;
```

The initialization consists in providing a valid set of initial distribution functions for every cell, and depends on the simulation scenario. By setting the equilibrium with a macroscopic flow speed of $\boldsymbol{u} = \boldsymbol{0}$, a system at rest is achieved, for example.

We are now going to present some important properties of the lattice Boltzmann method. As already mentioned, the Navier-Stokes equations can be derived from the lattice Boltzmann equation, Eq (2.31), via Chapman-Enskog analysis. This is an important tool in order to analyze the exact properties of a given Lattice Boltzmann model. Again, only the core results are stated. For details the reader is referred to [Thi05], and [Hän04].

For the lattice Boltzmann model as derived above, the pressure P is related to the density by the equation of state

$$P = c_s^2\, \rho \tag{2.34}$$

In presence of an external force the following continuity equation counterpart can be shown to hold for a time step δ_t

$$\frac{\partial \rho}{\partial t} + \nabla_x \cdot \rho \boldsymbol{u} + \frac{\delta_t}{2}\, \rho\, \boldsymbol{a} = 0. \tag{2.35}$$

The macroscopic momentum of the fluid is then calculated as

$$\rho\, \boldsymbol{U} = \rho \boldsymbol{u} + \frac{\delta_t}{2}\, \rho\, \boldsymbol{a}, \tag{2.36}$$

with corresponding macroscopic velocity $\boldsymbol{U}$.

In terms of viscosity one has

$$\tau = \frac{\lambda}{\delta_t} = \frac{\nu + \frac{1}{2} c_s^2\, \delta_t}{c_s^2\, \delta_t}. \tag{2.37}$$

For $\delta_t \to 0$ this yields

$$\lambda = \frac{\nu}{c_s^2}, \tag{2.38}$$

which is exactly the viscosity relation for the BGK equation, Eq. (2.12). These relations can be used to choose proper relaxation parameters for a given viscosity $\nu = \frac{\eta}{\rho}$.

In [Thi05] and [Hän04] also the Navier-Stokes equations are derived from the lattice equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho\, \boldsymbol{U} \;=\; 0 \tag{2.39}$$

$$\frac{\partial}{\partial t} \boldsymbol{U} + \boldsymbol{U}\, (\nabla \cdot \boldsymbol{U}) \;=\; -\frac{1}{\rho} \nabla P + \nu\, \nabla^2 \boldsymbol{U} + \boldsymbol{a} \tag{2.40}$$

Those are the compressible Navier-Stokes equations, that will resemble the incompressible equations for small Mach numbers $\mathrm{Ma} = \frac{|\boldsymbol{U}|}{c_s^2} \ll 1$.

Another important constraint concerning compressibility in presence of a body force ($\boldsymbol{a} = \frac{\boldsymbol{F}}{m} \neq \boldsymbol{0}$) is

$$|\boldsymbol{a}|\, L \ll c_s^2, \tag{2.41}$$

[BG00], where L is the extent of the domain in direction of the acceleration vector $\boldsymbol{a}$.

Chapter 3
Simulation of Floating Objects

The main task of this thesis is the integration *floating objects* into a lattice Boltzmann model, in order to simulate a liquid (i.e. Boltzmann model in the *incompressible limit*) with a free surface in interaction with various rigid bodies. Thus the concept aims at the simulation of a domain partially filled with fluid, while the cells in the remaining regions form an *atmospheric* gas phase. The external force in the lattice Boltzmann model comes into the picture to create the effect of a *gravitational field.* Provided that the liquid is allowed to come to rest, it will build up a planar surface and a pressure gradient in the direction of the gravity vector. The pressure gradient will cause buoyant forces exerted on any immersed body. To simulate the rigid bodies exposed to fluid forces an additional physics engine is used.

The simulation will use the 3-dimensional D3Q19 lattice model presented first place including an overview of the various extensions built into the model (Sec. 3.1). This outline is followed by a description of the *single-phase free surface* extension to the lattice Boltzmann model (Sec. 3.2). The different boundary conditions used for the treatment of solid walls are explained and how to modify them for moving obstacles. (Sec. 3.3).

For the handling of moving obstacles, the Boltzmann solver was coupled with a rigid body physics engine, to allow true interaction of obstacles with fluid (Sec. 3.4).

3.1 Computational Model

In Chapter 1 we have seen that the discretization of phase space naturally yields a lattice of cells communicating with each other by local advection of particles along lattice links corresponding to the set of discrete lattice velocities. The D3Q19 model used in the following part of the thesis is illustrated in Fig. 3.1.

	i	$\boldsymbol{v}_i$
C	0	$c\,(0,0,0)$
W, E	1,2	$c\,(\pm 1,0,0)$
N, S	3,4	$c\,(0,\pm 1,0)$
T, B	5,6	$c\,(0,0,\pm 1)$
NW, NE, SW, SE	7,8,9,10	$c\,(\pm 1,\pm 1,0)$
TW, TE, BW, BE	11,12,13,14	$c\,(\pm 1,0,\pm 1)$
TN, TS, BN, BS	15,16,17,18	$c\,(0,\pm 1,\pm 1)$

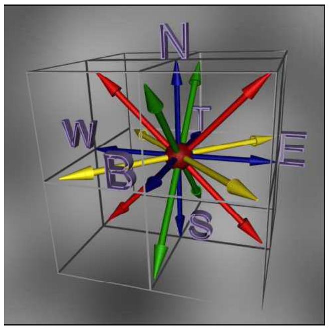


Figure 3.1. The 3-dimensional D3Q19 lattice model with 19 velocity vectors.

For the discrete equilibrium function $f_{\mathrm{eq},i}$ and the discrete force term F_i from the lattice Boltzmann equation, Eq. (2.31), the lattice weights w_i of the D3Q19 model are given by

$$w_i = \begin{cases} \frac{1}{3} \text{ for } i = 0 \\ \frac{1}{18} \text{ for } i = 1...6 \\ \frac{1}{36} \text{ for } i = 7...18 \end{cases} \tag{3.1}$$

The parameter c is normalized to 1.

The simulation shall include obstacles as well as a gas phase and neither the obstacles nor the gas will be handled by the lattice Boltzmann algorithm. Hence cells filled with liquid have to be distinguished from non-liquid cells. Therefor additional cell states are introduced and a flag value is stored in every cell. Possible cell states are *liquid*, *gas*, *interface* and *obstacle*. Interface cells are special liquid cells that neighbour gas cells and are needed to resemble the free surface between the two phases. As described in detail in Sec. 3.2, the interface cells always form a closed layer separating the liquid from the gas phase.

Obstacle cells are placed as solid walls around the simulation domain as well as they mark cells covered by floating objects. An obstacle cell blocks all fluid flow.

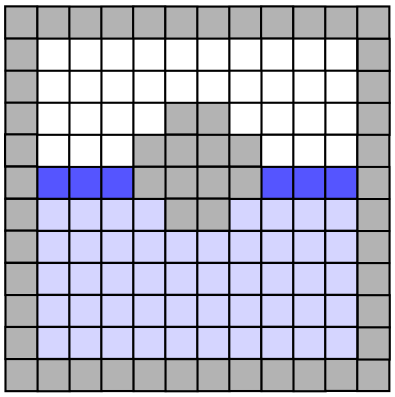


Figure 3.2. 2D slice through a typical lattice configuration. Liquid cells (light blue) are separated from the gas (white) by a layer of interface cells (dark blue). Obstacle cells (grey) block the liquid at the borders of the domain. The obstacle cells in the center represent a floating body.

Fig. 3.2 shows a slice through a possible simulation domain. Cells at the borders of the liquid phase, take a cell placed next to an obstacle or a gas cell as an example, have to be treated specially. For those cells the next sections will present the appropriate boundary conditions, which were incorporated into the lattice Boltzmann model. With the LBM boundary treatment usually consists in the local reconstruction of unknown particle distribution functions, incoming from non-liquid regions during the stream step.

3.1.1 Parametrization of the LBM

The parametrization of a LBM based simulation is a crucial task. Real world values have to be scaled down to dimensionless *lattice values.*

The Parameters of interest for a free surface flow scenario are:

- the fluid viscosity $\nu\,[\frac{m^2}{s}]$
- the surface tension $\sigma\,[\frac{N}{m}]$
- the external force strength(e.g. gravity) $\boldsymbol{a} = \frac{\boldsymbol{F}}{m}\,[\frac{m}{s^2}]$, $\boldsymbol{F}$ $[N]$
- the fluid density $\rho\,[\frac{\mathrm{kg}}{m^2}]$

The given fluid density is usually normalized to the lattice reference density ρ^*

$$\rho^* = \frac{\rho}{\rho} = 1.$$

For a given spacing δ_x and time step δ_t the dimensionless values (indicated by *) are obtained through the following equations:

$$\begin{aligned}\nu^* &= \nu\frac{\delta_t}{\delta_x^2}\\ \sigma^* &= \sigma\frac{\delta_t^2}{\rho\,\delta_x^3}\\ \boldsymbol{a}^* &= \boldsymbol{a}\frac{\delta_t^2}{\delta_x}\end{aligned}$$

Thus, δ_x and δ_t define the internal length scale and timescale of the model. One will also want to consider the Reynolds number of the scenario, which is given by

$$\mathrm{Re} = \frac{|\boldsymbol{u}^*_{\mathrm{ref}}|\,L^*}{\nu^*},$$

with *characteristic length* $L^*(=L\frac{1}{\delta_x})$, and reference lattice velocity $\boldsymbol{u}^*_{\mathrm{ref}} = \boldsymbol{u}_{\mathrm{ref}}\frac{\delta_t}{\delta_x}$.

Note that the reference velocity $\boldsymbol{u}^*_{\mathrm{ref}}$ is limited by Eq. (2.20). In practice it is usually sufficient to assure $\boldsymbol{u} < 0.1$.

3.1.2 Lattice Initialization

Most simulation scenarios in this thesis assume a liquid at rest initially (i.e. zero macroscopic velocity throughout the domain), that is perturbed by the motion of immersed objects. Still, an initialization of all cells with a reference density $\rho_0 = 1$, and macroscopic velocity $\boldsymbol{u} = \boldsymbol{0}$ is not adequate in many cases. In presence of an external force the liquid must therefore be initialized to be in state of hydrostatic equilibrium, such that the force field is balanced by a pressure gradient. For ideal (in this case incompressible) liquids this is given if the pressure at any point of depth z is

$$P(z) = P_0 + \rho\, g\, z, \tag{3.2}$$

where P_0 is the atmospheric pressure at $z = 0$, i.e. at the free surface or the highest position in the container of the liquid, respectively. g is the acceleration due to the force field (e.g gravity), and $\rho = \mathrm{const}$ is the liquid's density.

However, the equation of state, Eq. (2.34), in the lattice Boltzmann model is that of an ideal gas, and hence, an initialization with Eq. (3.2) does not result in a state of equilibrium: Since density $\rho(z) = \frac{1}{c_s^2}\,P(z)$ increases with depth z and is not constant, there will be fluctuations, if this formula is used. Obviously the slope of $P(z)$ is given by $g\ \rho(z)$ in consistency with the second moment of the force term, Eq. (2.22), which gives rise to

$$\frac{dP}{dz} = g\,\rho(z) = \frac{g}{c_s^2}\,P(z). \tag{3.3}$$

This ordinary differential equation can be solved with $P(0) = P_0$ as initial condition, which yields

$$P(z) = P_0 \cdot e^{\frac{g}{c_s^2}\cdot z}. \tag{3.4}$$

Note that this is basically the *barometric formula*. Expressed in terms of density ρ, and with $c_s = \frac{1}{\sqrt{3}}$, $\rho_0 = 3P_0$,

$$\rho(z) = \rho_0 \cdot e^{3gz} \tag{3.5}$$

can be used to initialize the lattice with hydrostatic equilibrium.

For the simulations in this thesis a reference density of $\rho_0 = 1$ was used. For scenarios including an atmospheric gas phase, this yields a constant gas pressure $P_G = P_0 = \frac{1}{3}\ \rho_0 = \frac{1}{3}$ throughout the *atmosphere,* and a nearly linear pressure gradient $P(z)$ in the fluid.

3.2 Single Phase Free Surface Model

A free surface, as already mentioned, is the boundary layer between two immiscible fluids, for example the surface between air and water. Although this may seem a banal phenomenon experienced virtually every day, simulating free surfaces is a rather challenging task, and a research area of its own. For this thesis the model from [KT04] and [Poh07] was used, which was specially designed to work with the lattice Boltzmann model presented above. The way pursued is called *single phase* model, which means that the second phase is assumed to be a gas of negligible influence on the flow, such that only the liquid phase has to be simulated. Thus, while the liquid is simulated by the lattice Boltzmann model, the gas phase comes into play only in terms of boundary conditions imposed at the free surface.

One key ingredient is the interface cell type, that is used to keep track of the free surface movement, whereupon the interface cells form a closed layer between liquid and gas phase.

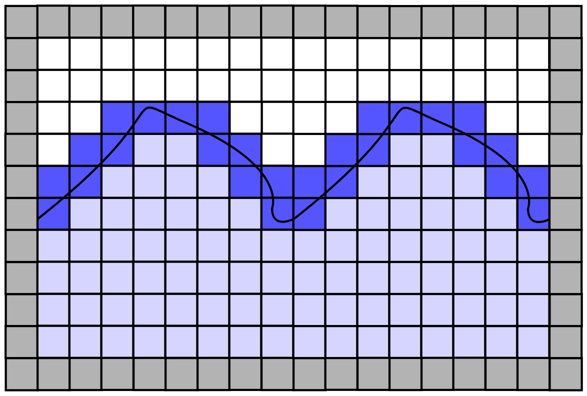


Figure 3.3. Interface cells form a closed layer separating liquid and gas phase.

Depending on the actual position of the surface, the portion of an interface cell that is filled with liquid may change. Hence a fill level $\varphi(\boldsymbol{x}, t)$ is stored in each interface cell. Mass content is then given by

$$M(\boldsymbol{x}, t) = \rho(\boldsymbol{x}, t)\ \varphi(\boldsymbol{x}, t)\,,$$

provided that the cell volume is normalized to $\delta_x^3 = 1$. For $\varphi = 1$ an interface cell contains as much liquid as a liquid cell. For $\varphi = 0$ it contains no liquid at all like a gas cell.

3.2.1 Fluid Advection

During the stream step an interface cell $\boldsymbol{x}$ may loose or gain liquid mass from surrounding liquid and interface cells. This mass exchange $\Delta M(\boldsymbol{x}, t)$ is calculated from the amount of liquid streamed into or out of the cell along each lattice link. The mass exchange $\Delta M_i(\boldsymbol{x}, t)$ with the adjacent cell in direction i is given by the following:

$$\Delta M_i(\boldsymbol{x}, t) = \begin{cases} 0 \\ f_{\bar{i}}(\boldsymbol{x} + \boldsymbol{v}_i\,\delta_t, t) - f_i(\boldsymbol{x}, t) \\ \frac{1}{2}(\varphi(\boldsymbol{x}, t) + \varphi(\boldsymbol{x} + \boldsymbol{v}_i\,\delta_t, t))\,(f_{\bar{i}}(\boldsymbol{x} + \boldsymbol{v}_i\,\delta_t) - f_i(\boldsymbol{x}, t)) \end{cases} \text{for } \boldsymbol{x} + \boldsymbol{v}_i\delta_t \in \begin{cases} G \\ L \\ I \end{cases} \tag{3.6}$$

Here G, L and I denote the set of gas, liquid and interface respectively.

Now $\Delta M(\boldsymbol{x}, t)$ is obtained as the sum over all $\Delta M_i(\boldsymbol{x}, t)$. The overall mass conservation is assured by $\Delta M_{\bar{i}}(\boldsymbol{x} + \boldsymbol{v}_i\, \delta_t, t) = -\, \Delta M_i(\boldsymbol{x}, t)$.

3.2.2 Cell Conversion at the Interface Layer

The free surface is not a fixed boundary, hence cells have to be converted either between gas and liquid or between interface and gas, as shown in Fig. 3.4. Direct changes, e.g. from liquid to gas, are not allowed in order to ensure a closed interface layer.

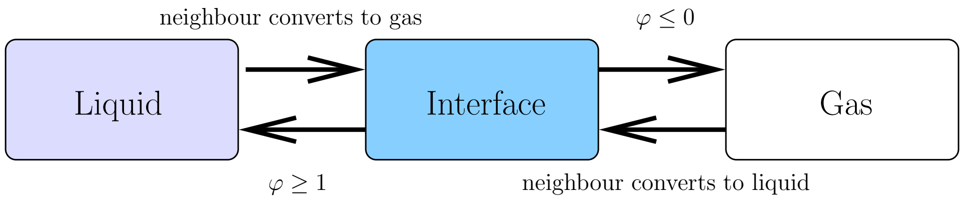


Figure 3.4. Possible state changes for the interface layer.

Whenever a gas cell is transformed into an interface cell its initial mass and fill values are set to zero, while for a liquid-to-interface conversion, the initial fill value is set to 1.

3.2.3 Free Surface Boundary Condition

Cells at the transition from liquid to gas phase are marked as interface. Since the gas phase is not simulated in this single phase model, one has to come up with a way to treat this kind of boundary and construct the missing distribution functions during the stream step. This is accomplished by using the pressure of the gas phase P_G, which is exerting a certain force on the free surface and equate this force to the one exerted on the free surface by the liquid side [KT04].

No macroscopic calculations need to be done, since missing distribution functions can be constructed for each lattice direction individually. Let $\boldsymbol{x}$ be an interface cell with a gas cell at $\boldsymbol{x} + \boldsymbol{v}_i\, \delta_t$, then

$$f_{\bar{i}}'(\boldsymbol{x}, t + \delta_t) = f_{\text{eq},i}(\rho_G, \boldsymbol{u}(\boldsymbol{x})) + f_{\text{eq},\bar{i}}(\rho_G, \boldsymbol{u}(\boldsymbol{x})) - f_i(\boldsymbol{x}, t). \tag{3.7}$$

The equilibrium function is used to involve the gas phase, with density set to $\rho_G = \frac{1}{c_s^2}\, P_G = 3\, P_G$ according to the pressure relation, Eq. (2.34). The velocity parameter is set to the velocity of the interface cell $\boldsymbol{u}(\boldsymbol{x})$. Thus the free surface freely follows the fluid flow.

To take into account the effect of surface tension, the density ρ_G substituted into the equilibrium function is modified to

$$\rho_G = 3\, P_G + 6\, \sigma\, \kappa(\boldsymbol{x}, t), \tag{3.8}$$

where σ is the surface tension of the liquid, and $\kappa(\boldsymbol{x}, t)$ is the local curvature of the free surface.

For the calculation of κ surface normals are constructed first, one for each interface cell. The normals are approximated from the gradient of the fill levels in the neighbourhood of the cell. Assuming the surface as planar within the range of a single cell, a point interpolated by the approximated free surface is calculated in every interface cell. The local curvature $\kappa(\boldsymbol{x})$ can then be extracted from this information. For details, see [Poh07].
The obtained surface points and normals can also be used for visualizations (see Sec. 4.3).

3.2.4 Atmosphere and Bubbles

The gas phase might not always form a connected region. Separated volumes of gas (*bubbles*) might also have different values for pressure and contain different amounts of gas. This can be realized by calculating the actual volume $V_b(t)$ of each gas region and therefrom, in turn, the actual gas pressure of the bubble by

$$P_G(t) = \frac{V_b(0)}{V_b(t)}, \tag{3.9}$$

where $V_b(0)$ is the initial volume of the bubble (here an initial gas pressure of 1.0 has been assumed).

Each bubble has an index as identifier which is stored at the associated gas and interface cells. There is usually one bubble defined as *atmosphere*. For this bubble an atmospheric reference pressure P_0 defined as constant, independent from the actual bubble volume. Hence the gas pressure in the atmosphere is not affected by movements of the interface layer.

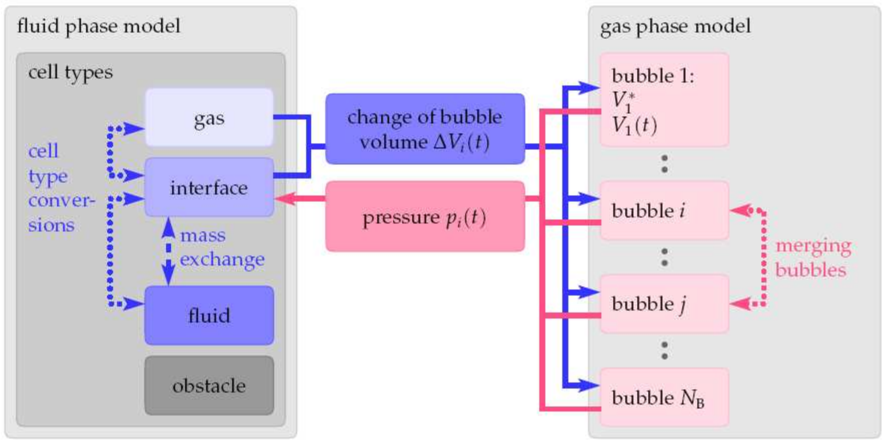


Figure 3.5. Free surface flow model from [Poh07]. The organization of the gas phase in separated volumes is illustrated on the right.

3.3 Solid Wall Boundary Conditions

This section deals with boundary conditions at obstacle cells - for example the walls limiting the simulation domain or other solid obstacles, that block the fluid flow (Fig. 3.2). For the integration of moving objects a moving wall boundary condition is needed.

3.3.1 No-slip, Free-slip and Part-slip

Assume that $\boldsymbol{x}$ is a liquid cell with a neighbouring obstacle cell at $\boldsymbol{x} + \delta_t \boldsymbol{v}_{\bar{i}}$. One of the standard boundary conditions is the *no-slip* bounce-back scheme given by

$$f_i'(\boldsymbol{x}, t+\delta_t) = f_{\bar{i}}(\boldsymbol{x}, t). \tag{3.10}$$

Here the wall is assumed to be placed exactly halfway between liquid and obstacle cell, such that particles are simply reflected back into the cells they are coming from. The result is a zero normal and zero tangential velocity, and thus resembling rough and sticky surfaces, which slow down the fluid flow (Fig. 3.6).

If the wall is moving with velocity $\boldsymbol{u}_o$, then according to [Lad93], the no-slip boundary condition becomes

$$f_i'(\boldsymbol{x}, t + \delta_t) = f_{\bar{i}}(\boldsymbol{x}, t) + \frac{6}{c^2} w_i \, \rho \, \boldsymbol{v}_i \boldsymbol{u}_o, \tag{3.11}$$

where ρ is the fluid density at the boundary, giving a macroscopic fluid velocity $\boldsymbol{u} = \boldsymbol{u}_o$ at the obstacle surface. Obviously the distribution function f_i is amplified or damped depending on its orientation with respect to the wall velocity $\boldsymbol{u}_o$. Note that this boundary treatment preserves mass only for tangential wall velocity vectors $\boldsymbol{u}_o$.

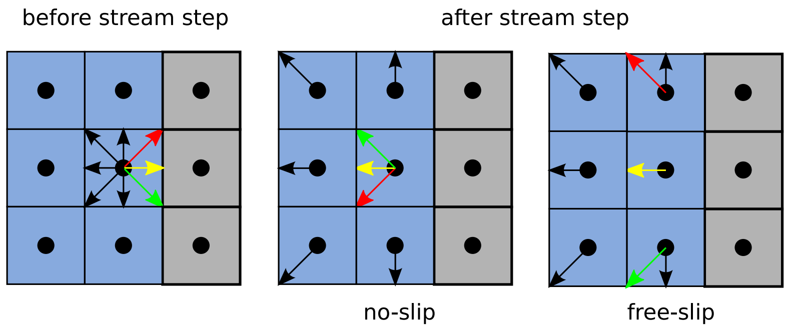


Figure 3.6. Illustration of no-slip and free-slip boundary conditions.

The counterpart of the no-slip boundary condition is *free-slip*, and also leads to a zero normal velocity, but does not affect the tangential velocity. Free-slip is therefore suitable to simulate smooth surfaces, where friction is negligible. *Part-slip* is another boundary condition that interpolates between no-slip and free-slip and can be used to model surfaces with properties lying in between the two extremes. Free-slip and Part-slip both are not used for the simulations in this thesis.

3.3.2 Curved Boundaries

The bounce-back schemes introduced so far all assume the reflecting wall to be located exactly between two grid nodes. If a flow around curved obstacles, for example spherical or cylindrical bodies is to be simulated, this assumption is clearly broken and will lead to errors.

One possibility to approximate curved walls is to introduce a linear interpolation scheme with the actual wall distance Δ as relaxation parameter. Such a model is given in [BFL01].

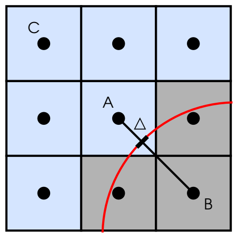


Figure 3.7. Lattice link between cells A and B cut in half by a curved boundary in $A + \Delta(B - A)$. In this case ($\Delta < \frac{1}{2}$) the distribution functions from A and C will be interpolated for the reflected particle distribution.

For a liquid cell A and a neighbouring obstacle cell at $B = A + \boldsymbol{v}_i\, \delta_t$, let $A + \Delta\, (B - A)$ be the intersection point of the corresponding lattice link with the wall surface as shown in Fig. 3.7. The missing distribution function coming from the wall direction is then set to

$$f_{\bar{i}}'(A, t+\delta_t) = 2\,\Delta\; f_i(A, t) + (1 - 2\,\Delta)\; f_i(A - \delta_t\, \boldsymbol{v}_i, t) + \frac{6}{c^2}\, w_i\, \rho\, \boldsymbol{v}_i\, \boldsymbol{u}_o \qquad (\Delta < 1/2), \tag{3.12}$$

$$f_{\bar{i}}'(A, t+\delta_t) = \frac{1}{2\,\Delta} f_i(A, t) + \frac{(2\,\Delta - 1)}{2\,\Delta} f_{\bar{i}}(A, t) + \frac{1}{\Delta}\frac{3}{c^2}\, w_i\, \rho\, \boldsymbol{v}_i\, \boldsymbol{u}_o \qquad (\Delta \geqslant 1/2), \tag{3.13}$$

where $\boldsymbol{u}_o$ is the velocity of the wall. Note that in case of $\Delta = \frac{1}{2}$ the formula reduces to the usual no-slip boundary condition.

Since in the case of a no-slip boundary the wall is assumed to be exactly between to grid points, by simply reflecting them back into the fluid cell, particles still travel exactly the distance between two grid points. For a wall distance $\Delta < 1/2$, however, interpolation is done between $f_i(A)$ and $f_i(A - \delta_t\, \boldsymbol{v}_i)$, since the particles that arrive at $f_{\bar{i}}'(A, t + \delta_t)$ after streaming, must be expected to come from a point in between the grid nodes A and $C = A - \delta_t\, v_i$ (Fig. 3.7). For $\Delta \geqslant 1/2$ a similar consideration leads to the interpolation of $f_i(A)$ and $f_{\bar{i}}(A)$.

A higher order interpolation is possible and can be found in [LL03]. Note that there is a fundamentally different method to improve the approximation of curved boundaries, known as grid refinement. For a comparison of different approaches see [YMLS03].

The curved boundary conditions as described above are useless, unless the exact intersection point of the affected lattice link with the wall surface is known. In this thesis three types of geometrical primitives were considered, namely spheres, cubes and capsules.

Sphere Intersection

To determine the intersection point of a lattice link with a sphere an algorithm from [AMH02] was used. Let $\boldsymbol{c}$, r be the sphere center and radius, respectively, and $\boldsymbol{d} = B - A$ the lattice link intersecting the object. The original algorithm tests if a ray cast from a point $\boldsymbol{o}$ in direction $\boldsymbol{d}$ hits the sphere and if so it returns the intersection point. We have $\boldsymbol{o} = A$ and know that there must be an intersection - therefor some steps of the algorithm can be omitted.

As first step $\boldsymbol{l} = \boldsymbol{c} - \boldsymbol{o}$ is calculated, which is the connecting vector of ray origin and sphere center. Now the projection $s = \boldsymbol{l} \cdot \boldsymbol{d}$ is determined. If ($s < 0$ and $\boldsymbol{l}^2 > \boldsymbol{r}^2$), the ray misses the sphere, which is located 'behind' $\boldsymbol{o}$ then. Now $m = \sqrt{\boldsymbol{l}^2 - \boldsymbol{s}^2}$ is obtained by the Pythagorean theorem. For $m > r$ the ray misses the sphere. Else $q = \sqrt{\boldsymbol{r}^2 - m^2}$ can be calculated, and the intersection points are given by $\boldsymbol{o} + (s \pm q)\, \boldsymbol{d}$.

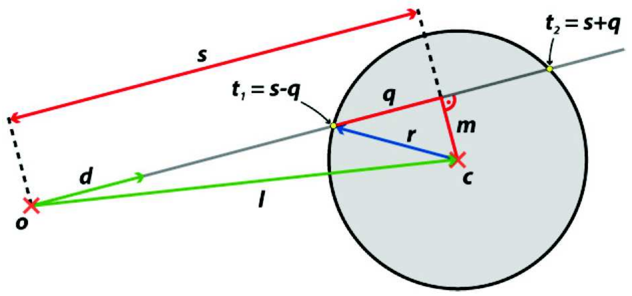


Figure 3.8. Picture taken from [Igl05] illustrates the functionality of the sphere intersection according to an algorithm from [AMH02].

In our case, the intersection point will always be $A + (s - q)\, \boldsymbol{d}$, and therefore $\Delta = s - q$. The other case can occur only if $\boldsymbol{o}$ lies inside the sphere.

Box Intersection

The algorithm for box intersections was also taken from [AMH02] and is based on the slabs method. A slab is simply two parallel planes. For every slab one has an inside (the space between the two planes) and an outside, thus any box can be defined by three slabs.

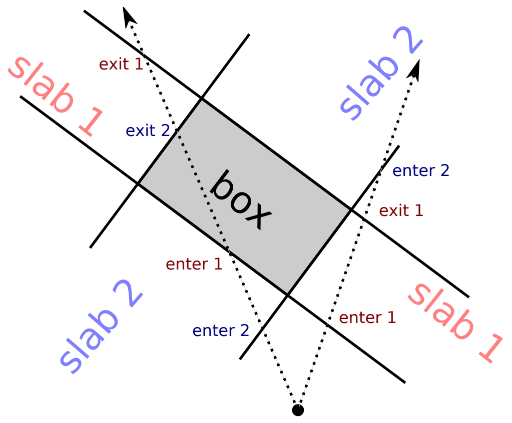


Figure 3.9. The slabs method in two dimensions: Here the box can be defined by two slabs. A ray (dotted lines) that hits the box has to enter all slabs before first exiting a slab (like the ray directed left), otherwise it does not hit the box (right directed ray).

For each slab, the points of intersection with the two planes are calculated. Unless the ray is parallel to the planes they will always exist. One intersection point marks the entrance of the ray into the slab, the other is where the ray leaves the slab. If now all of the three entrance points are nearer to the ray origin than the exit points, there is an intersection. The box intersection point closest to the ray origin is given by the entrance point farthermost from the ray origin.

Capsule Intersection

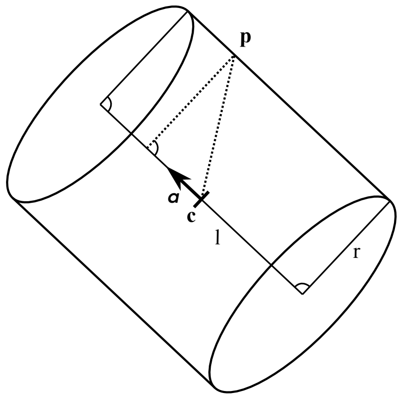


Figure 3.10. A cylinder with center $\boldsymbol{c}$ and orientation $\boldsymbol{a}$, length l and radius r.

A capsule is made up of a cylinder and two hemispheres as caps. Therefor the algorithm for the sphere intersection can be reused for the caps, and the problem reduces to the determination of the intersection points of the lattice link with a cylinder. If the cylinder is given by its center $\boldsymbol{c}$ and orientation $\boldsymbol{a}$, length l and radius r (Fig. 3.10), then an intersection point can be determined from the following equation, which holds for any point $\boldsymbol{p}$ lying on the (infinite) cylinder surface.

$$|(\boldsymbol{p}-\boldsymbol{c})-(\boldsymbol{a}\cdot(\boldsymbol{p}-\boldsymbol{c}))\,\boldsymbol{a}\,|=r \tag{3.14}$$

$$((\boldsymbol{p}-\boldsymbol{c})-(\boldsymbol{a}\cdot(\boldsymbol{p}-\boldsymbol{c}))\,\boldsymbol{a})^2=r^2 \tag{3.15}$$

$(\boldsymbol{a}\,(\boldsymbol{p}-\boldsymbol{c}))\,\boldsymbol{a}$ is simply the projection of $\boldsymbol{p}$ onto the cylinder axis $\boldsymbol{a}$. Now the lattice link with unknown intersection parameter Δ can be substituted into the equation as $\boldsymbol{p}=\boldsymbol{o}+\Delta\,\boldsymbol{d}=A+\Delta(B-A)$. The intersection point is obtained by solving the quadratic cylinder equation for Δ:

$$\Delta_{1,2}=\frac{-\beta\pm\sqrt{\beta^2-4\,\alpha\,\gamma}}{2\,\alpha}, \tag{3.16}$$

where

$$\begin{aligned}
\alpha &= (\boldsymbol{d}-\boldsymbol{a}\,(\boldsymbol{a}\cdot\boldsymbol{d}))^2, && (3.17)\\
\beta &= 2\,(\boldsymbol{d}-\boldsymbol{a}\,(\boldsymbol{a}\cdot\boldsymbol{d}))\,((\boldsymbol{o}-\boldsymbol{c})-\boldsymbol{a}\,(\boldsymbol{a}\cdot(\boldsymbol{o}-\boldsymbol{c}))), && (3.18)\\
\gamma &= (\boldsymbol{o}-\boldsymbol{c})-\boldsymbol{a}\,(\boldsymbol{a}\cdot(\boldsymbol{o}-\boldsymbol{c}))-r^2. && (3.19)
\end{aligned}$$

There are two roots, whereof the smaller one represents the intersection point closer to the origin. To determine, whether the intersection point $\boldsymbol{o}+\Delta\,\boldsymbol{d}$ actually lies on the finite cylinder with length l, we project it onto the cylinder axis. If not, then the sphere intersection algorithm above is utilized to intersect the lattice link with the caps.

3.4 Moving Obstacles

In order to incorporate the floating objects, the lattice Boltzmann model was coupled with a rigid body physics engine (see Chapter 4). By making use of the moving wall boundary condition presented in Sec. 3.3.2, the obstacle movement may influence the fluid flow. Conversely, the obstacle movement may be affected by the fluid, for example by exerting a drag force on the object. Three types of primitive rigid bodies, namely spheres, boxes and capsules, and arbitrary combinations of those primitives are used in this thesis. A primitive is a solid object defined by its material, i.e. density, and extents (e.g. sphere with radius r_o and density ρ_o). During a time step, a rigid body's position and orientation in the simulation domain will change according to its linear and angular momentum. The latter may be affected by collisions with other bodies (directly handled by the physics engine), or by directed forces applied on the object. Both forces on the body's center of mass and forces on arbitrary points on the body's surface are supported.

The task now lies in the calculation of the force exerted on the object by the fluid, which can be done by the *momentum exchange method* described in section 3.4.2. To apply this method, the actual boundaries of the obstacle have to be known, which means that the rigid body shape needs to be mapped into the lattice and updated for each time step in accordance with the object movement.

3.4.1 Obstacle Mapping

Since obstacles are allowed to move around freely within the domain, and thus the fluid boundary may change during the simulation, the lattice has to be updated accordingly by mapping the shape of the obstacle to the lattice (Fig. 3.11). The mapping is done by checking whether a lattice node lies inside the rigid body or not. If so, then the cell is marked as an obstacle and a no-slip moving wall boundary condition is imposed as described in Sec. 3.3, setting $\boldsymbol{u}_o$ to the velocity of the obstacle in that point. This velocity, that can be queried from the physics engine is a combination of the body's linear and angular velocity. Thus fluid flow is induced by obstacle movement.

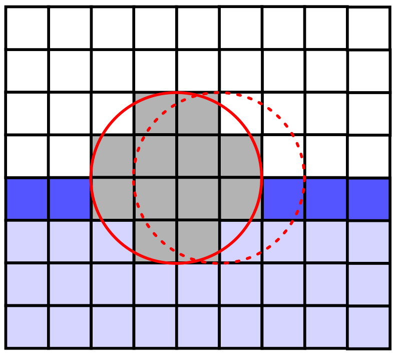


Figure 3.11. A sphere moving rightwards. According to the new sphere position (dotted line), lattice cells must be converted either from obstacle to gas, interface or liquid, or the other way around.

Obstacle mapping was adopted in principle from [GFI+08]. By comparing the cell mapping of an obstacle at time $t + \delta_t$ to the mapping at the preceding time step, it is possible to spot lattice cells changing from obstacle to non-obstacle. For an all liquid domain such a cell $\boldsymbol{x}$ is converted into liquid by initializing the full set of distribution functions f_i with the equilibrium function $f_{\mathrm{eq},i}$. Thereby the equilibrium is parametrized with the obstacle velocity in the cell center from the previous time step t. The local density $\rho(\boldsymbol{x}, t + \delta_t)$ is obtained by interpolation of the ρ values from other liquid cells in the neighbourhood of $\boldsymbol{x}$ at time t. If density variations throughout the domain can be assumed to be small, it is also possible to simply use the reference density for $\rho(\boldsymbol{x}, t + \delta_t)$.

For obstacles interacting with a free surface flow the situation is more complicated: Here the bodies might be only partly immersed in the liquid, or in no liquid surrounding at all and hence setting a prior obstacle cell to liquid might lead to invalid lattice configurations. Luckily the free surface model used (Sec. 3.2) is keeping track of the free surface position and marks cells at the transition to the gas phase as interface. This can be utilized to propose a local decision strategy for the conversion of obstacle to non-obstacle cells without disrupting the interface layer representing the free surface. The possible state changes are illustrated in Fig. 3.12.

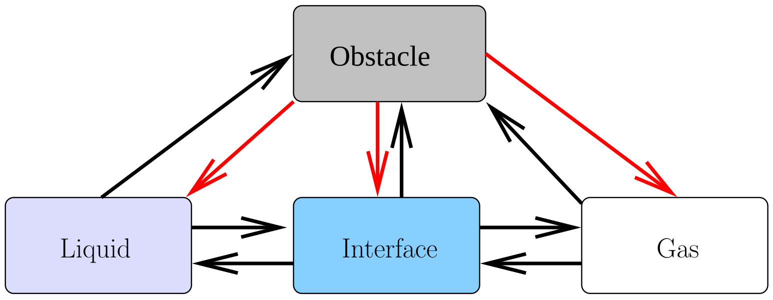


Figure 3.12. Prior obstacle cells have to be converted into either gas, interface or liquid (red transitions).

For a prior obstacle cell at $\boldsymbol{x}$ that is about to switch to a non-obstacle state, we define the set of non-obstacle cells in the neighbourhood as $N := \{\boldsymbol{x} + \boldsymbol{v}_i\, \delta_t \,|\, \boldsymbol{x} + \boldsymbol{v}_i\, \delta_t \text{ is no obstacle}\}$. Now the decision for the state of $\boldsymbol{x}$ at $t + \delta_t$ subject to N is as follows.

If N contains only liquid, then $\boldsymbol{x}$ is converted into liquid.
If N contains only gas, then $\boldsymbol{x}$ is converted into gas.
If N contains multiple cell-types and $n \geqslant 0$ interface cells, the decision is critical:

- If N contains liquid cells but no gas cell, then $\boldsymbol{x}$ is set to liquid, as the cell can be assumed to lie “under” the liquid surface. $\rho(\boldsymbol{x}, t + \delta_t)$ is interpolated from N and $\boldsymbol{u}(\boldsymbol{x}, t + \delta_t) = \boldsymbol{u}_o$ as above.

- If N contains gas cells but no liquid cell, then $\boldsymbol{x}$ is set to gas, as it can be assumed to lie outside the liquid. Its bubble index is set to the index of the surrounding gas cells.
- If N contains liquid cells as well as gas cells, then $\boldsymbol{x}$ is set to interface. $\rho(\boldsymbol{x}, t + \delta_t)$ is interpolated from all non-gas cells in N, with $\boldsymbol{u}(\boldsymbol{x}, t + \delta_t) = \boldsymbol{u}_o$ as before. Interface cells may be only partially filled with liquid, depending on the exact position of the free surface, hence a fill - value has to be specified, such that $M(\boldsymbol{x}) = \varphi(\boldsymbol{x})\ \rho(\boldsymbol{x})$ holds, where $M(\boldsymbol{x})$ is the liquid mass inside the cell. So either φ or M has to be interpolated for the new interface cell $\boldsymbol{x}$ from the interface cells in N. If the number of nearby interface cells n is 0, then we set $M(\boldsymbol{x}) = \varphi(\boldsymbol{x}) = 0$.
  Again the bubble index has to be copied from nearby gas or interface cells.

Mass is not conserved during obstacle movement due to the following effects: First of all because the moving wall boundary condition, Eq. (3.11), is not mass conserving. For an all liquid domain the resulting errors are usually small, but for obstacles penetrating the liquid surface they may become more apparent depending on the obstacle movement. For a partially immersed object, at a planar liquid surface with normal $\boldsymbol{n}$, the mass defect ΔM can be expected to be proportional to $-\boldsymbol{u}_o \cdot \boldsymbol{n}$. Besides mass fluctuations due to flow induced by obstacle movement, the liquid mass in the surrounding of the object may change due to the discretization error involved by the obstacle mapping. The number of cells covered by an obstacle depends on the exact position and orientation of the obstacle (Fig. 3.13). This discretization error can be expected to reduce if the standard moving wall boundary condition is replaced by the curved boundary approximation of Eq. (3.12) and Eq. (3.13).

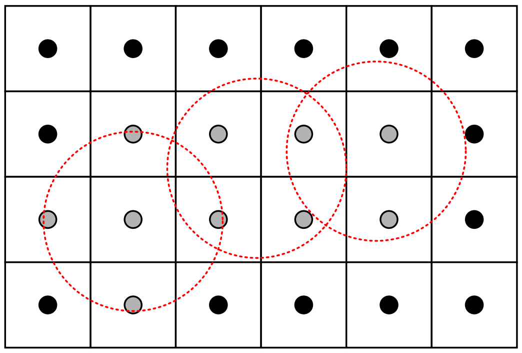


Figure 3.13. Sphere moving through the lattice covers a different numbers of cells depending on its actual position.

3.4.2 Momentum Exchange Method

When a body is situated in a liquid, it will be subjected to stresses and forces exerted by the surrounding pressure and flow. The most common way to calculate the fluid impact on an immersed body for lattice Boltzmann models is the *momentum exchange method* [Lad93, Lad07, MYSL02], which approximates the forces from the particle distribution functions reflected from a surface.

Assume now that a single particle with velocity $\boldsymbol{v}$ is colliding with a fixed solid wall. During the collision its momentum $\boldsymbol{p} = m\ \boldsymbol{v}$ will change by $\Delta\boldsymbol{p} = \boldsymbol{F}\ \Delta t$, by Newton's second law for elastic collisions. Thereby $\boldsymbol{F}$ is the force exerted on the particle. Because of the way boundary conditions are handled in the LBM, the determination of $\Delta\boldsymbol{p}$ is an easy task: Take, for example, the fraction of particles in cell $\boldsymbol{x}$ with velocity $\boldsymbol{v}_i$ that are reflected back at the surface of an obstacle. After the collision the velocity will be $\boldsymbol{v}_{\bar{i}}$ (provided that one of the no-slip based boundary conditions from Sec. 3.3. is applied), and thus

$$\Delta\boldsymbol{p} = f'_{\bar{i}}(\boldsymbol{x}, t + \delta_t)\, \boldsymbol{v}_{\bar{i}} - f_i(\boldsymbol{x}, t)\, \boldsymbol{v}_i = \big(f'_{\bar{i}}(\boldsymbol{x}, t + \delta_t) + f_i(\boldsymbol{x}, t)\big)\, \boldsymbol{v}_{\bar{i}}. \tag{3.20}$$

Whereupon the particle distribution functions, being fractions of the cell density, have been identified with the fluid mass involved in the collision, which is perfectly correct when a uniform cell volume $V_{\text{cell}} = \delta_x^3 = 1$ is presumed. Let O be the set of obstacle cells covered by the rigid body. Let N be the set of all cells neighboured to an obstacle cell in O. Then the net force exerted on the body is obtained by

$$\boldsymbol{F} = \sum_{\boldsymbol{x}\in N} \sum_{i=1}^{N} \boldsymbol{v}_i \left(f_{\bar{i}}'(\boldsymbol{x}, t+\delta_t) + f_i(\boldsymbol{x}, t) \right) \cdot \omega_O(\boldsymbol{x} + \delta_t \, \boldsymbol{v}_i), \tag{3.21}$$

where $\omega_O(\boldsymbol{x})$ is an indicator function with $\omega_O(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \in O$ and $w_O(\boldsymbol{x}) = 0$ otherwise.

If a cell $\boldsymbol{x} \in N$ is not a liquid but a gas cell, then the missing distribution functions have to be constructed in consistency with the free surface boundary condition, Eq. (3.2.3). To take the gas pressure P_G acting on the obstacle surface into account, the equilibrium function is substituted for the missing particle distribution:

$$f_i(\boldsymbol{x}) = f_{\text{eq},i}(\frac{1}{c_s^2} P_G, \boldsymbol{u}_o(\boldsymbol{x} + \delta_t \, \boldsymbol{v}_i))$$

For interface cells in N with a fill level $\varphi \in (0,1)$, the liquid momentum for lattice direction i becomes $f_i \cdot \varphi \cdot \boldsymbol{v}_i$, whereas the proportion $(1-\varphi)$ of the cell volume is filled with gas and has to be treated as above. This gives rise to

$$\boldsymbol{v}_i \left(\varphi(\boldsymbol{x}) \cdot f_i + (1 - \varphi(\boldsymbol{x})) \cdot f_{\text{eq},i}(\frac{1}{c_s^2} P_G, \boldsymbol{u}_o(\boldsymbol{x} + \delta_t \, \boldsymbol{v}_i)) \right)$$

as momentum in direction i.
Using the same approach for the outgoing direction $\bar{i}$, the momentum change $\Delta \boldsymbol{p}$ is obtained by Eq. (3.20) and the net force on the obstacle can be determined as before.

Chapter 4
Implementation Outline

The methods presented in the previous chapters have been implemented in the waLBerla (*widely applicable Lattice Boltzmann solver from erlangen*) software framework [GDF+07a, GDF+07b] and the *P.E.* physics engine [Igl07]. In this chapter a small overview of waLBerla will be given with a highlight on the parts that were of special importance or had to be customized for free surface flow with floating objects.

4.1 WaLBerla Software Framework

The waLBerla software framework is a large tool set, providing all data structures, input - and output methods for lattice Boltzmann based fluid simulations in three dimensions. Information on waLBerla can be found in the waLBerla Technical Reports [GDF+07a, GDF+07b]. During simulation the cell information is stored in fields:

- *PDFField* (particle distribution functions)
- *FlagField* (cell state, e.g. liquid, obstacle, interface, ..)
- *DensityField* (macroscopic density)
- *VelocityField* (macroscopic velocity)
- *FSFillField* (fill values for interface cells)
- ...

For some fields two instances are needed, since intact values from the last time step are needed for calculations. Most important the stream step (see Sec. 2.3.3), needs for each cell the particle distributions of the neighbouring cells of the preceding time step. Therefore it *reads* from a *source* PDFField while *writing* to a *destination* PDFField.

The fields are held in a *domain* data structure. Only the fields actually needed for the current applications will be instantiated (e.g. the FSFillField is needles in an all liquid domain without gas phase). According to the simulation applications various *sweeps* will be executed during each time step. Sweeps can be thought of as methods, that access the fields and alter the cell data. For example there is a *PDFSweep* that goes over all cells and executes the stream step and the collide step.

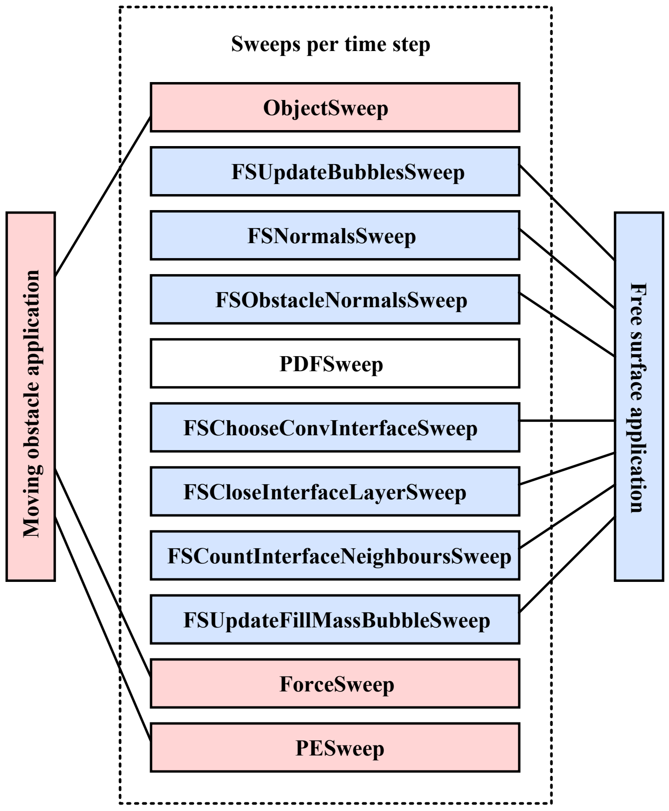

Sweeps per time step
ObjectSweep
FSUpdateBubblesSweep
FSNormalsSweep
FSObstacleNormalsSweep
PDFSweep
FSChooseConvInterfaceSweep
FSCloseInterfaceLayerSweep
FSCountInterfaceNeighboursSweep
FSUpdateFillMassBubbleSweep
ForceSweep
PESweep
Moving obstacle application
Free surface application

Figure 4.1. Outline of the sweep sequence that is executed for each time step. The light-red ones are executed for an application with moving obstacles, while the light-blue ones are only done for a free surface flow simulations.

The following sweeps are needed for floating objects in free surface flow (in the order of execution for each time step):

- *ObjectSweep:* Updates the lattice with actual obstacle flags according to obstacle positions as described in Sec. 3.4.1, sets up according boundary conditions, and handles the cell conversions of obstacle $\rightarrow$ {liquid, interface, gas}.
- *FSUpdateBubblesSweep, FSNormalsSweep, FSObstacleNormalsSweep:* Updates all gas regions and calculates the normals of the free surface (see Sec. 3.2.3 and 3.2.4).
- *PDFSweep:* The stream step and the collide step is executed for all cells. The latter also takes the external force into account and modifies the fluid momentum accordingly. The former also insures correct boundary treatment (see Sec. 3.3 and 3.2.3).
- *FSChooseConvInterfaceSweep, FSCloseInterfaceLayerSweep, FSCountInterfaceNeighboursSweep, FSUpdateFillMassBubbleSweep:* Handles the cell conversions due to mass exchange in the interface cells and assures a closed interface layer according to Sec. 3.2.1 and Sec. 3.2.2. Then the volume of each bubble is updated (see Sec. 3.2.4).
- *ForceSweep:* Calculates the force on each object as described in Sec. 3.4.2.
- *PESweep:* The P.E. is called with the updated forces on objects to update the object positions.

Some sweeps have been refined to better meet the application needs. Since these refinements only make sense if they are really needed, different versions of a sweep may be called depending on the defined simulation scenario. The PDFSweep, for instance, needs to do a lot more calculations for a free surface flow (e.g. mass exchange) than for a one-phase simulation. Hence the PDFSweep is implemented differently for this case.

4.2 Modified Sweeps

As shown in Fig. 4.1, sweeps are only added to the queue if they are needed in the current simulation scenario specified by the user. For the case of moving obstacles in free surface flow one therefore has a moving obstacles application as well as a free surface application. In the following we will see, how the sweeps had to be changed to combine both applications.

4.2.1 PDF Sweep

The PDFSweep, i.e. the stream and the collide step of waLBerla had already been split up in a version for one-phase applications (with or without moving obstacles) and a special version for free surface flow applications, respectively. Here only minor changes were necessary: For both versions the external force term, Eq. (2.29), is added to the particle distributions in the collide step. For free surfaces applications the force term is weighted by the fill value of the cell if necessary, and the moving wall boundary conditions were added according to Eq. (3.11).

The usual boundary treatment is done during the stream step and based on the flag field solely. If there are curved boundaries in the simulation, however, the sweep will additionally iterate over the list of wall intersections prepared by the object sweep (see next section).

4.2.2 Object Sweep

The object sweep is split up in 4 phases that are executed for all moving obstacles. They are *ReInit*, *SetObjectFlags*, *Restore* and *Remove*. ReInit removes all obstacle flags from the cells and marks them with a special *oldObstacle* flag. Then SetObjectFlags sets all cells to obstacle that are now covered by the obstacle according to the actual object position, oldObstacle flags are removed from these cells. If curved boundary treatment is to be done, the wall distances according to Sec. 3.3.2 are calculated and stored inside a list.

In the Restore phase the remaining oldObstacle flags have to be converted back to either liquid, interface or gas as described in Sec. 3.4.1. Here it proved to be especially important to interpolate the values like density, fill, etc. for the restored cell only from the prior non-obstacle cells - that is why there is an additional Remove - phase to finally erase the oldObstacle flags. Otherwise the interpolated values accumulate if more than one neighboured cells are restored at once, which in turn may produce unrealistic asymmetric behaviour in the obstacle movement: E.g., objects might get a non-zero momentum in x- and y-direction in a falling-sphere experiment, where gravity acts in z-direction only.

The ObjectSweep stays the same for all applications containing moving obstacles, whereas the ForceSweep has been further specialized for free surface simulations. Only the third phase of the object sweep could have been sped up by introducing an adapted sweep for free surface flow and thus sparing two conditional branches per near-obstacle cell in the standard procedure - a minor performance gain, but still, that might be considered in the future.

4.2.3 Force Sweep

The force sweep calculates the force exerted on a moving object by the liquid according to Sec. 3.4.2. Since in a free surface flow the force calculation differs depending on the neighbouring cell types, there is a specialized sweep for this case.

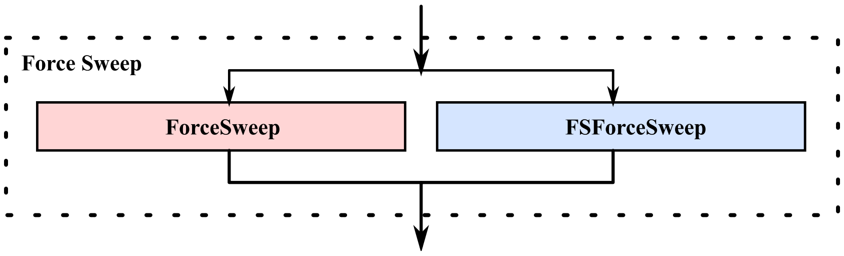


Figure 4.2. The force sweep has been specialized for different application needs. The conventional force sweep (left branch) is for one-phase all-liquid domains, whereas for two-phase free surface simulations the right branch is executed.

If curved boundary treatment is done, the force calculation changes, too, according to Eq. (3.12) and Eq. (3.13). In this case the force sweep iterates over the list of curved boundary intersection-values instead of all boundary cells of an obstacle.

After the force sweep, the P.E. engine is called with the new forces to update the object positions.

4.3 Other Improvements

Gravity Initialization

When using the lattice Boltzmann model with an external force it is usually necessary to initialize the lattice with the correct density gradient as described in Sec. 3.1.2. This was added to the *Create* method of the domain data structure. WaLBerla already had a user interface, that supported the definition of an axis aligned box shaped gas region to resemble a gas atmosphere in a *parameter file*. The external force can be specified as an acceleration vector in that file.

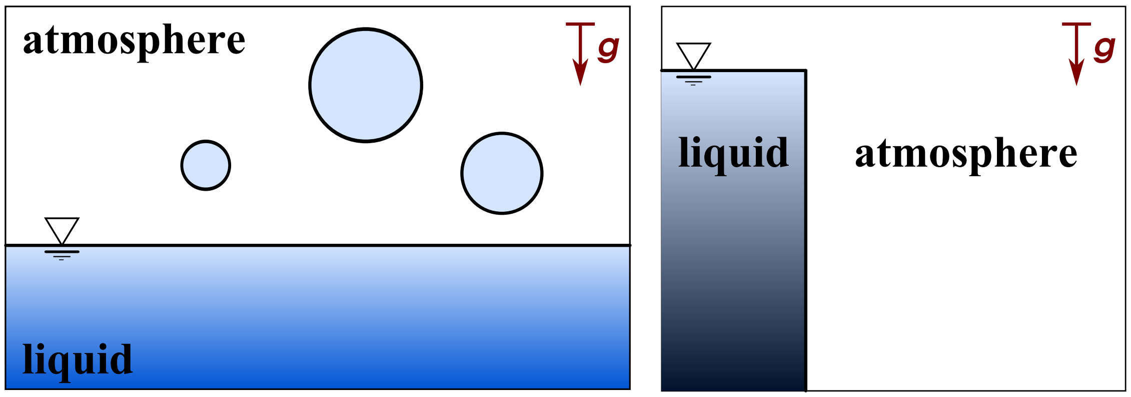


Figure 4.3. *Left:* If no reference point for depth calculation is specified, then a 'water level' will be automatically set by the program according to the initial free surface position. The hydrostatic pressure gradient is set in the liquid region below that level in direction of the gravity vector, while the spherical drops above are omitted and set to the reference density. *Right:* The lowest free surface point in this domain was on its bottom. However, as depicted, it is possible to set a reference point by hand, to have a correct hydrostatic pressure gradient in the water wall ready to burst.

The gravity initialization was implemented to be user toggled. When enabled, it will initialize the liquid region *below* the atmosphere with the correct density values. 'Below', in this case, means that all cells that have a positive depth in direction of the gravity vector relative to the minimal height of the initial free surface are initialized with a density $\geqslant 1$ (see figure 4.3). If there are liquid cells above the free surface, e.g. a drop specified inside the gas region, they will be omitted. For simulations without gas phase, the whole lattice is initialized starting downwards from the *highest* cell in the domain. Alternatively the user can also specify a point in the domain as reference point for the depth calculation during initialization. Then all cells lying below the reference point will be initialized with a density according to their depth relative to the reference point. This might be used for a breaking dam scenario, for instance, where initially a wall made up of liquid is put in the gas region.

Free Surface Visualization

In Sec. 3.2.3 the local surface tension approximation for the free surface was outlined, at which also the topology (e.g. surface points and normals, curvature, ..) of the free surface is calculated. This information can be used to create visualizations of the liquid. Therefor a routine was written to transform the free surface information into output readable by the POV-Ray ray tracer [oVPL04].

In a first effort the planar approximation within single cells was visualized by fitting polygons into the interface cells and through the calculated surface points using the surface normals as slope. This does not lead to a closed surface but is extremely handy to monitor the approximated surface normals (Fig. 4.5).

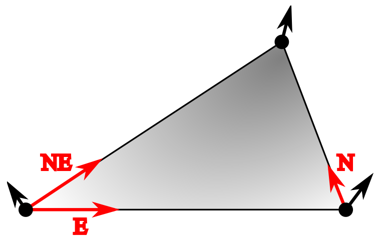


Figure 4.4. For three neighboured interface cells, a triangle is created through the surface points of the cells. Interpolation is done among the surface normals at the vertices for the inner points of the triangle, and thus the illusion of a smooth surface is created.

Visually more appealing is to create triangles for every subset of three neighboured surface points with surface normals of the same orientation. As indicated in the picture, the lattice topology is exploited to find those triplets. The method is simple to implement and yielded good results as long as the free surface geometry was not too complex (Fig 4.6). For more sophisticated visualization approaches of free surfaces see [Thü07].

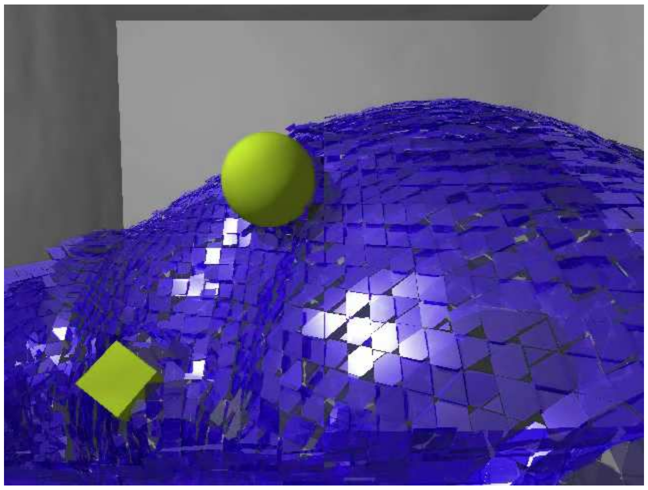


Figure 4.5. Free Surface visualized by polygonal surface elements in every interface cell (the local skewness becomes visible in each cell). Here a steady wave was generated by setting a tangential accelerating boundary condition to the floor of a cubic domain.

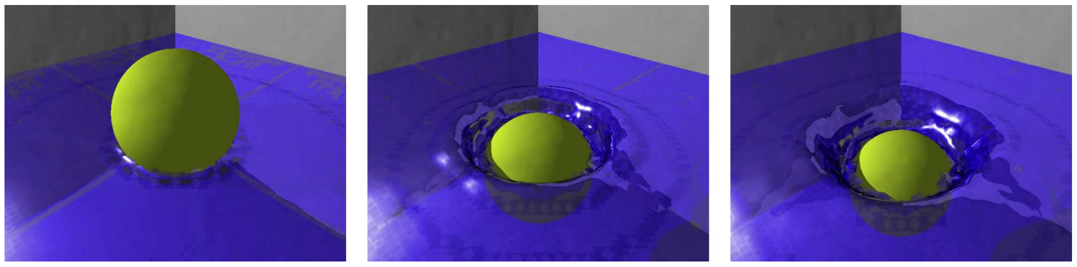


Figure 4.6. Free surface deformed upon the impact of a rigid sphere at selected time steps, visualized by smooth triangles.

Chapter 5

Test Cases and Results

In the following, some experiences with the model described in Chapter 3 are stated. The test cases have been selected to show the capabilities and limitations of the described methods. The gravity model and initialization was tested separately without moving objects in an all-liquid domain. Incorrect initialization or infringement of the incompressibility condition results in unintentional shock waves, that could influence the object movement (Sec. 5.1).
A serial of test cases has been dedicated to the force calculation, using Stokes' Law for validation. The focus lies on the error of the obstacle discretization to lattice cells and its influence on drag force and buoyant force calculation (Sec. 5.2).
Some possible examples for simulation setups with rigid bodies in a free surface flow are presented (Sec. 5.3). With moving obstacles mass is not strictly conserved by the used method. The problem of mass fluctuation is investigated for the special case of objects penetrating the interface layer (Sec. 5.4).

5.1 Gravity Model

Since gravity was a crucial factor for the effect of a floating object, the external force model as described in Chapter 2 was tested separately in the first place.

Density Gradient

A first qualitative test was performed on a $64 \times 64 \times 64$ lattice, with initial uniform density $\rho = 1.0$, resembling a similar setup as described in [BG00], with relaxation parameter $\tau = 1.0$. No-slip boundary conditions have been used at all edges of the domain. Gravity was applied with strength $g = -\ 0.001$ in direction of the z-axis. Density was repeatedly measured along a vertical line in the center of the domain.

Because of the external force that is instantly applied with the start of the simulation, the density profile starts to fluctuate. The fluid can be seen to oscillate up and down in shock waves, until after several 1000 time steps the simulation converges towards a final density gradient. Figure 5.1 shows the fluctuations of the density profile along the vertical line.

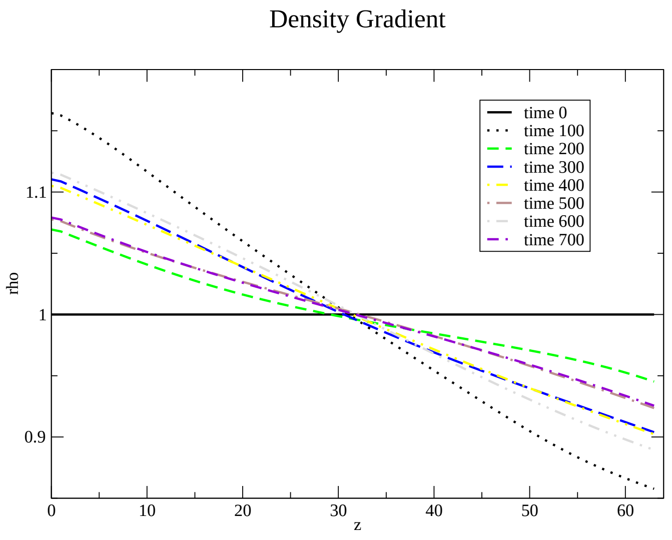


Figure 5.1. Density as a function of height at different time steps when gravity is applied on a lattice with initial uniform density $\rho = 1.0$. The Density gradient can be seen to oscillate about its final position.

Next the initialization as described in Sec. 3.1.2 was tested. Therefor, the same setup was initialized with a density gradient: In a first run (a), the ideal formula from Eq. (3.2) was applied. Then, in a second run (b), Eq. (3.5), that was derived from the lattice model, has been used to initialize the lattice. In both cases the reference density was set to $\rho_0 = 1.0$. Then, over several 1000 time steps, the maximal and the averaged fluid velocity were monitored, to make the shock-waves due to wrong initialization visible.

As expected, because of the density-pressure relation of the lattice Boltzmann model, initialization (a) does not lead to a hydrostatic equilibrium: The fluid oscillates with a velocity of the order of g. The initial average velocity is more then 10^4 times as high as with method (b). This shows the importance of an initialization with the correct density values by method (b).

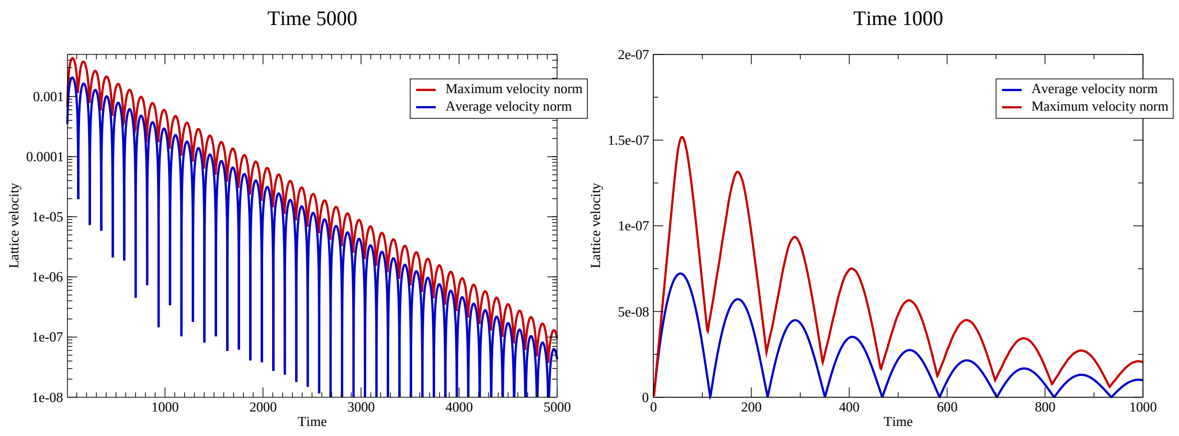


Figure 5.2. Average and maximal velocity induced by an external force field with $g = -0.001$ are compared between the two initialization methods. The decay of shock-waves with method (a) over 5000 time steps is shown on the left (a logarithmic scale was used). It takes about 5000 time steps until the velocity drops under that of the initial state of method (b) shown on the right.

Note that the selected gravity value $g = -0.001$ is near the threshold imposed by the incompressibility condition (2.41). For reasonable values of g, the density gradient is closer to the ideal linear formula and the shock-waves are of negligible influence on immersed obstacles, which has been the primary concern in respect to the simulations in the following sections. Table 5.1

shows the initial shock wave velocity for smaller values of g with $\frac{g\,L}{c_s^2} < 1$ according to Eq. (2.41) for the initialization according to Eq. (3.5).

g along z-axis	$\frac{g\,L}{c_s^2}$	initial max. velocity	initial avg. velocity
-0.001	0.192	1.5×10^{-7}	7.1×10^{-8}
-0.0001	0.0192	1.5×10^{-10}	7.1×10^{-11}
-0.00001	0.00192	1.5×10^{-13}	7.1×10^{-14}

Table 5.1. Initial shock waves with initialization formula from Sec. 3.1.2 (method (b)).

Breaking Dam

In contrast to the above paragraph gravity induced fluid motion can sometimes be the desired effect. According to Fig. 4.3 a block of liquid in a gas atmosphere was initialized with hydrostatic pressure as if constrained by an invisible wall. This way a miniature breaking dam was realized in a water-like fluid ($\nu = 1 \times 10^{-6}\frac{m^2}{s}$). High resolutions have to be chosen to simulate fluids of low viscosity: $\delta_x = 0.0001m$, $\delta_t = 3.4 \times 10^{-5}s$ ($\tau = 0.510207$) with a gravity of $g = 1.1355 \times 10^{-4} = 9.81\,\frac{m}{s^2}$ (earth's gravitational constant). A lattice of $75 \times 25 \times 25$ cells has been used for the simulation, with a fixed wall in the middle of the domain. Figure 5.3, shows selected frames visualized with povray.

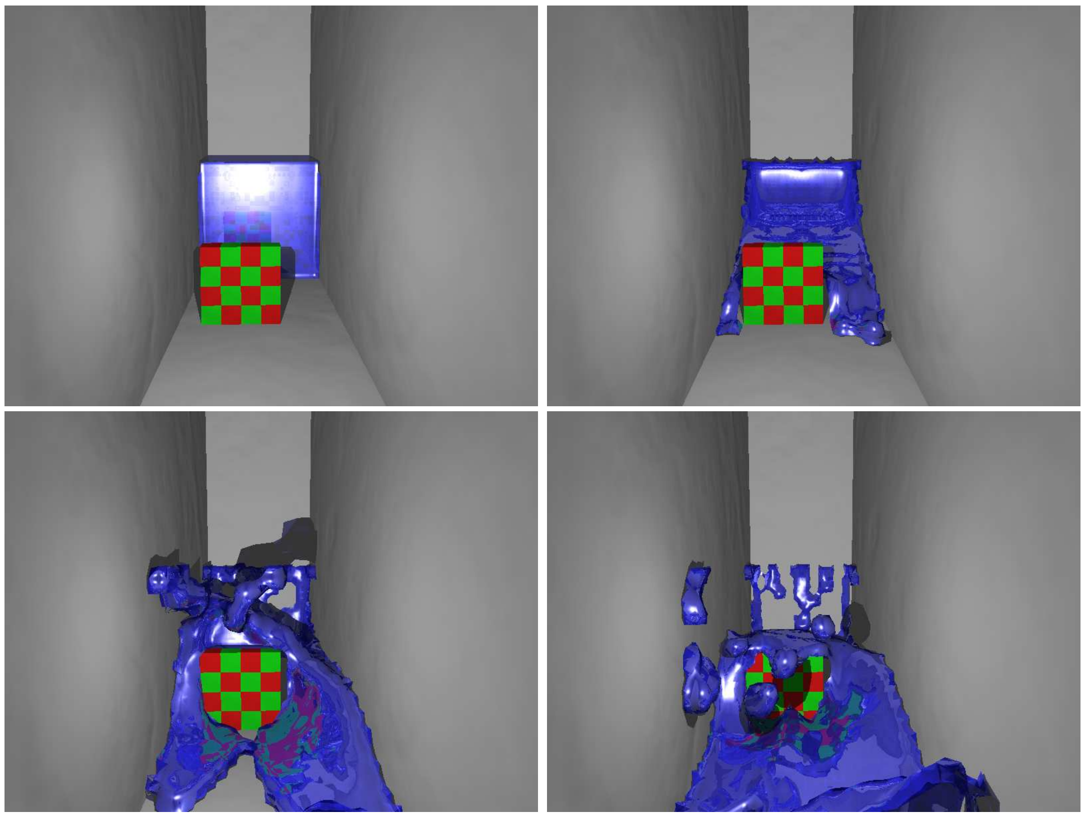


Figure 5.3. Breaking dam at time step 1, 700, 1500 and 1900.

5.2 Force Calculation

Drag Force

As a general test of the *momentum exchange method* from Sec. 3.4.2 the drag force on a sphere moving with a constant velocity was measured and the result was compared to the ideal *Stokes' law* given by Eq. (1.8). The simulation was carried out with a staircase-approximated sphere and then separately with the curved boundary conditions from Sec. 3.3.2. Then both test cases were repeated in the presence of an external force to exhibit changes in the drag force as consequence of a density gradient along z-axis in the domain. The simulation domain consisted of 100^3 cells filled with liquid of viscosity $\nu = 0.4$, the sphere (radius $r = 5$) was positioned in the center, and moved with $u = 10^{-4}$ along the x-axis. The relaxation time was chosen to $\tau = 1.7$. The external force was set to $g = 9.81 \times 10^{-6} = 9.81\frac{m}{s^2}$. In physical values one lattice space and time step can be interpreted as $d_x = 0.01m$ and $\delta_t = 10^{-4}s$, respectively. The viscosity ($\nu = 0.4 = 0.4\frac{m^2}{s}$) is that of liquid somewhere between cooking oil and castor oil. The ideal drag force according to Stokes' law was in this case

$$F_{D,\,\mathrm{ideal}} = 3.7699 \cdot 10^{-3}.$$

Fig. 5.4 shows the simulation results without gravity. Since the sphere is started impulsively to move at a constant speed, it takes about 1000 time steps until the actual drag force is reached. The averaged relative error in the drag force calculation (in absence of gravity) was:

15.70% (Staircase Approximation) 14.72% (Curved Boundary Treatment).

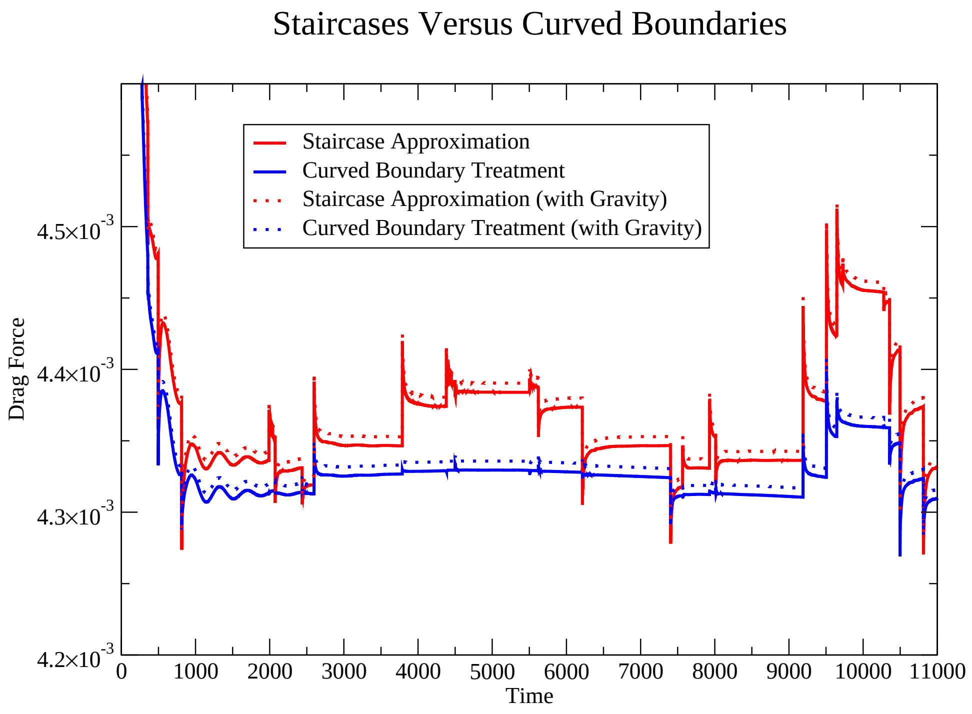


Figure 5.4. Drag force on a sphere moving at a constant velocity, variation of imposed boundary conditions and influence of a hydrostatic pressure gradient.

All curves lie significantly above the ideal value, which is caused for one reason by the discretization of the sphere to lattice cells. Also, the curved boundary treatment is not able to smooth out all the approximation errors, which are visible as peaks in the graph, whenever the mapping of the sphere changes. This can be verified by comparing the drag force to the graph in Fig. 5.5, which shows the exact number of lattice cells covered by the sphere over time. Since after 10000 time steps the sphere has traversed exactly one cell, the mapping then starts to repeat periodically. Also note the symmetry in the graph about the axis $t = 5000$. The influence of discretization error will be even greater for the buoyant force on the sphere (see next paragraph). Furthermore one also has to expect a significant wall influence in the simulation, since the ideal law assumes an infinite domain of liquid around the sphere. If the no-slip boundary conditions at the domain borders are replaced by free-slip walls one can expect the error to reduce. More information on drag force calculation with lattice Boltzmann and different boundary conditions can be found in [Fei05].

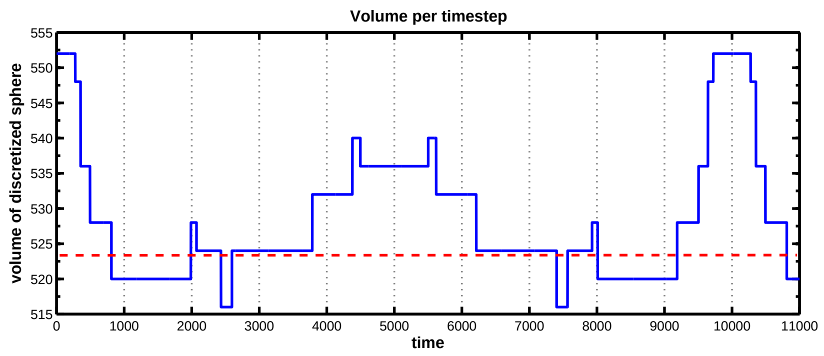


Figure 5.5. Volume (i.e. number of cells) of the discretized sphere, depending on its offset in the lattice. The ideal volume of the sphere ($r = 5$) is ≈ 523.6 (red dotted line). Changes in the mapping reappear in the force graphs, Fig. 5.4 and 5.6.

The slightly increased drag force in presence of gravity is caused by increased fluid density around the sphere. In this case the average density with gravity was around $\rho_{\text{average}} = 1.0015$ which is a $+0.15\%$ deviation from the reference density 1.0 of the former simulation and equals the deviation of the drag force.

Buoyant Force

In the simulation setup described in this section, also the buoyant forces on the sphere were measured (see Fig. 5.6). Here the *Archimedean law* reappears, which describes the buoyant force to be proportional to the volume of the immersed body. Since the volume of the sphere changes because of the staircase approximation in the lattice model, its buoyant force is also affected by this error. Interestingly the curved boundary conditions even increased the error.

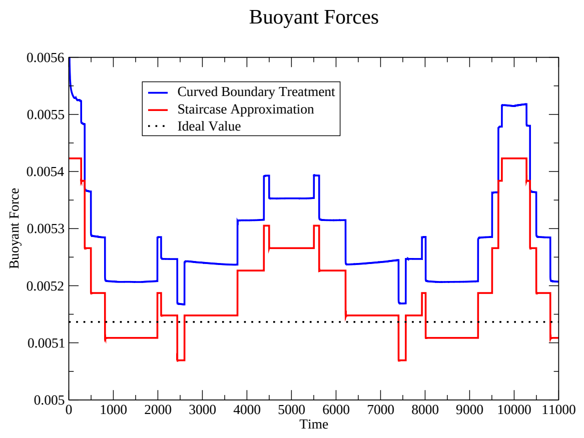


Figure 5.6. Buoyant force in the simulation is influenced by the obstacle discretization error. Averaged relative error: 1.31% (staircase approximation) and 2.80% (curved boundary condition).

For movement along a straight line, the deviations in the sphere volume can be precomputed: Figure 5.7 shows that the averaged error has a local maximum for $r=5$ (Note that in our model a point with integral coordinates (P_x, P_y, P_z) is located exactly in the center of a cube of eight lattice cells, such that the next grid node in direction BSE in the lattice has coordinates $(P_x - 0.5, P_y - 0.5, P_z - 0.5)$).
Unsurprisingly the error generally reduces for larger radii; for $r \leqslant 3.5$ there are immense deviations. But then, there are still a lot of local minima with an average error around and below 3%.

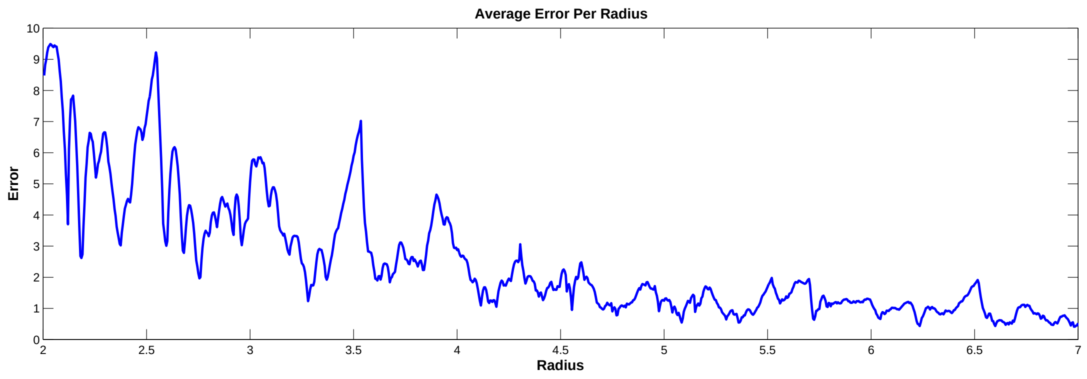


Figure 5.7. Average percental volume error for spheres traveling along a straight axis-aligned line.

Falling Sphere

For this experiment, the sphere velocity is no longer fixed to stay constant, but a result of the interacting forces of gravity, buoyancy and friction. A sphere of radius $r = 5$ and density $\rho_s = 1.733944954$ was positioned in the center of the domain, with initial velocity $u_s = 0$. Thus the gravitational force on the sphere was constantly $F_G \approx 8.9064 \times 10^{-3}$. The sphere density was chosen, such that the ideal terminal velocity of the falling sphere according to equation (1.9) would be

$$u_{s,\,\mathrm{ideal}} = 1 \times 10^{-4}.$$

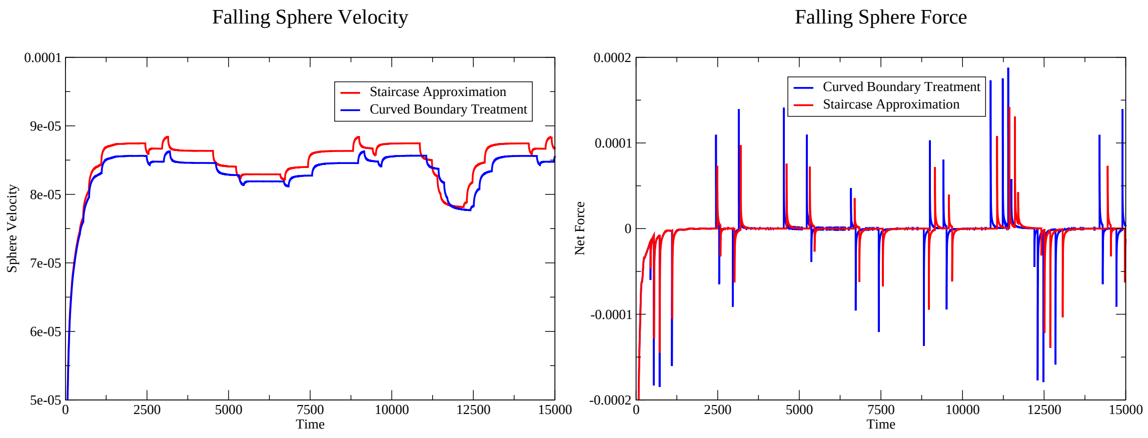


Figure 5.8. Falling sphere experiment. Sphere velocity (left) and net force (right).

As expected, because of the changes in the buoyant force and the drag force but the constant gravitational force on the object, the velocity of the sphere must change accordingly. These changes are depending on the sphere's exact position in the lattice (Fig. 5.8). The right graph shows that after each change in the sphere mapping, force balance is recovered quickly and accompanied by an acceleration or deceleration of the sphere velocity (left graph). The most obvious slow-down occurs around $t = 11000$, when the sphere occupies more than 550 cells. By slightly modifying the sphere radius from $r = 5$ to a local minimum according to Fig. 5.7 to, for example, $r = 5.085$ or $r = 4.975$ it was possible to reduce the velocity fluctuation in the falling sphere scenario.

Another workaround for the error due to fluctuations of the buoyant force is to calculate the gravitational force separately for each obstacle cell of the sphere. Fig. 5.9 shows that the resulting velocity curve is smoother. The gravitational force is directed opposite to the buoyant force and if it is scaled according to the staircase approximation of the sphere as well, the two errors cancel each other out.

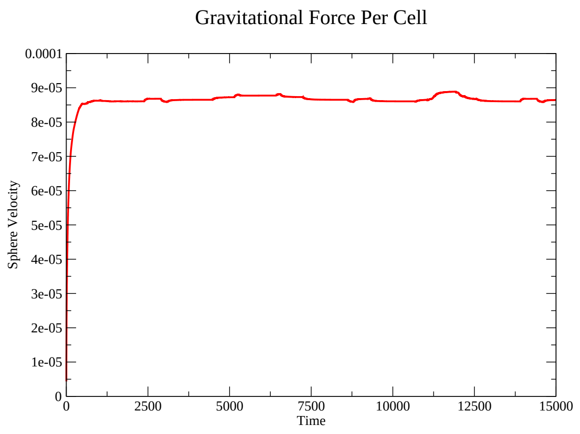


Figure 5.9. Falling sphere experiment with gravitational force calculated for every obstacle cell separately.

A general problem with the curved boundary conditions from Sec. 3.3.2, that showed up in all scenarios with a non-zero gravitational constant was the induction of flow in liquid cells around objects. The induced flow velocity near the curved boundaries was of the same order as the gravitational constant.

Also, it should be noted that for objects of asymmetric geometry the force calculation by the momentum exchange method, Sec. 3.4.2, can lead to unrealistic results: E.g., an open box was constructed from 5 box primitives (Fig. 5.10 shows a two-dimensional scheme of the construction). For the box on the right hand side in Fig 5.10 the resulting net force in a domain with constant density was directed towards the box primitive in the center. This is because additional edges lead to an unequally distributed number of lattice links, that transfer momentum in the downward direction.

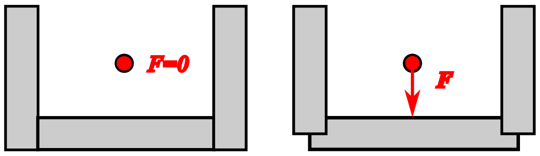


Figure 5.10. Force calculation failed for the box shape on the right.

5.3 Floating Objects

Advection Test

The advection test simulates floating obstacles exposed to a fluid in motion. A 100^3 cell domain was initialized as half liquid, half atmosphere (gas region). Three obstacles - a sphere, a cube, and a capsule with specific density 0.5 - were initialized at different locations to float on the liquid surface, advected by the fluid flow. The flow was thereby induced by a moving wall boundary condition at the North boundary of the domain, as depicted below (Fig. 5.11). The wall velocity was set to $\boldsymbol{u}_w = (0.001, 0, 0)$ to generate a circulating flow in the cavity and advect the objects. In this test the parameters were set to $\tau = 1.7$, $g = -9.81 \times 10^{-6}$ and $\nu = 0.4$.

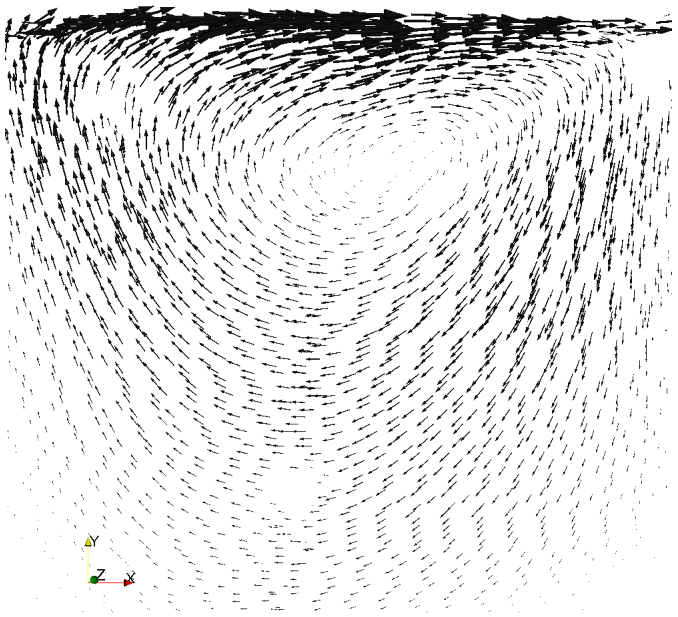


Figure 5.11. Flow induced by defining a moving wall boundary condition with a tangential velocity in the north-boundary of the domain. The figure shows the flow in the interface layer.

The floating objects have been visualized in figure 5.12. Notice the rotational movement of the blue capsule around the curl in the upper half of each picture. Three snapshots were taken after 1, 1.7 million and 3 million time steps.

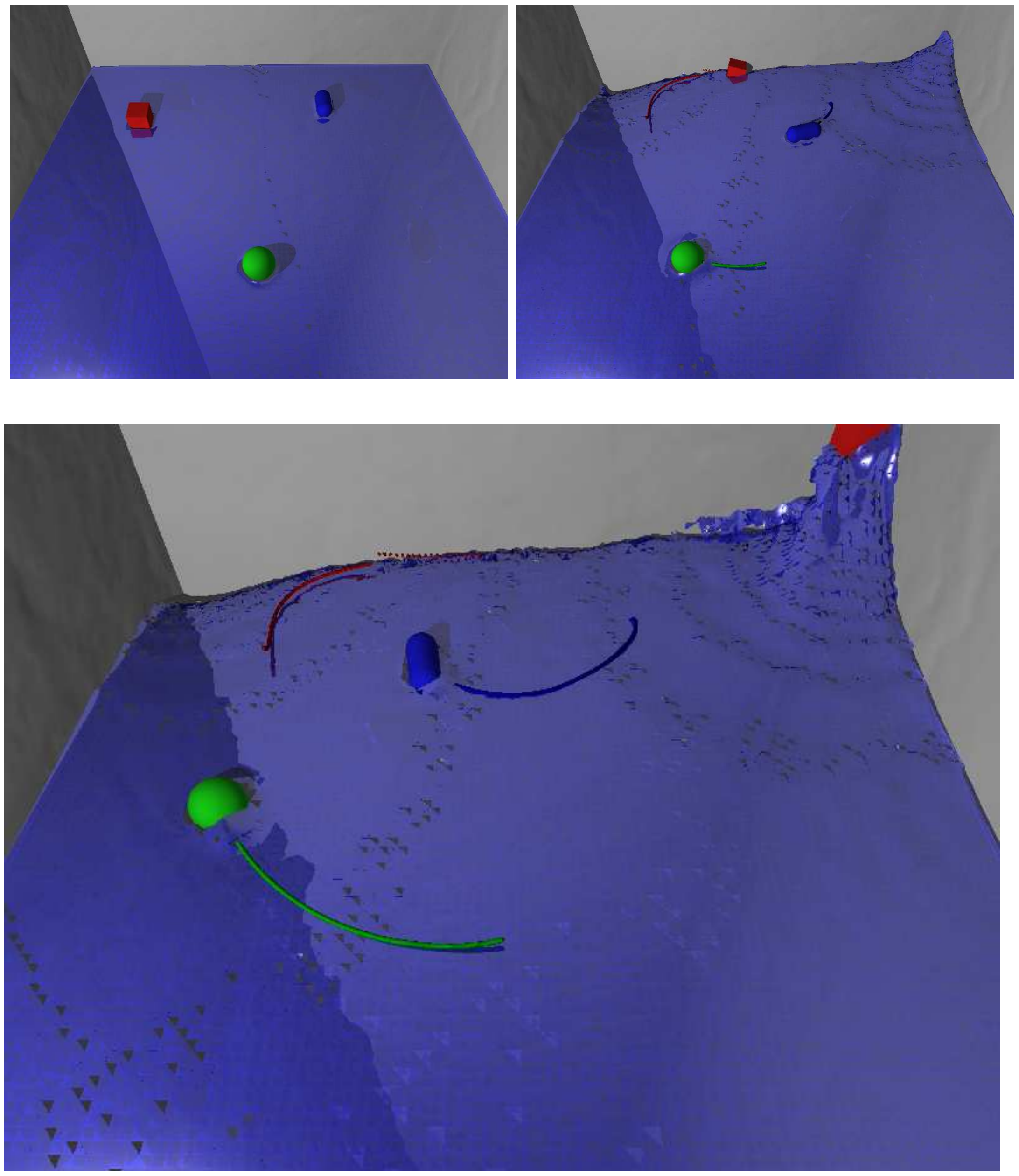
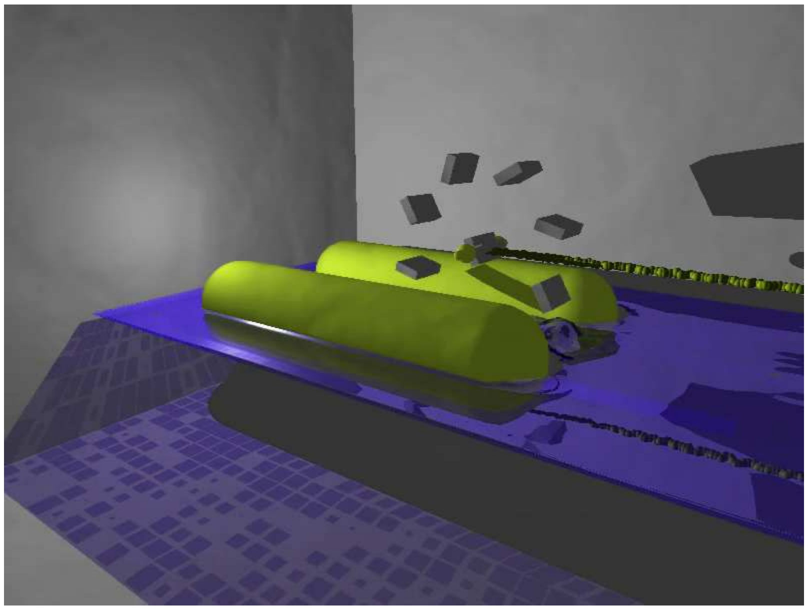


Figure 5.12. Object trajectories in a free surface flow. The blue capsule performs a rotational movement around a curl. The red box is pushed upwards at the right wall.

Moving Boat



In the advection test obstacle motion was induced by fluid motion. For the moving boat scenario multiple rigid bodies were combined to a miniature boat pushing itself forward by turning a paddle wheel. The paddle wheel was constructed of eight circularly aligned box primitives. This wheel was set to turn with a constant angular velocity around an axle (capsule primitive) connected with the two hulls of the boat. The result was an overall forward motion until the boat hit the wall. The trajectory of the wheel axle was made visible in the picture. A low viscosity $\nu = 10^{-6}\frac{m^2}{s}$ (water) was used for the test case.

Buoyancy with a Concave Object

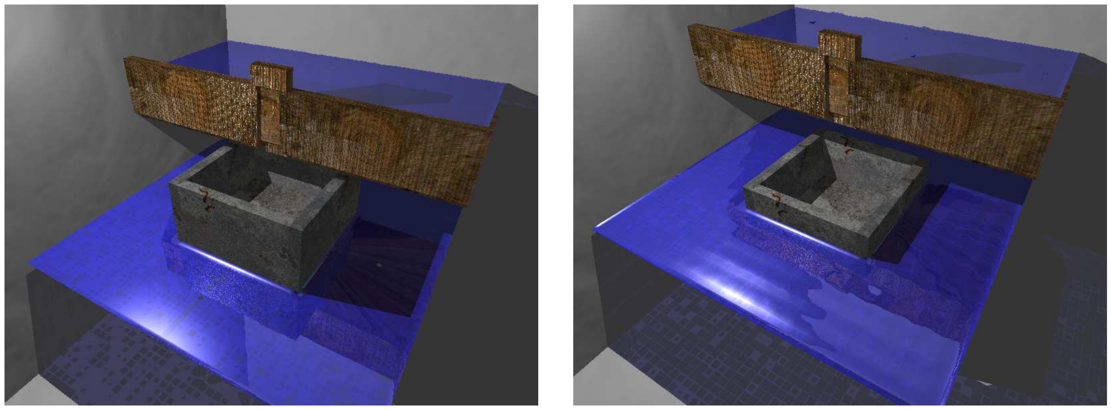


Figure 5.13. Buoyancy prevents an open box with specific density *greater* than 1.0 from sinking. The initial position is shown on the left. The right hand side shows the box in the position in which buoyancy and gravitational force are in balance.

So far the effect of a floating object has been considered only for objects with a lower material density than the liquid density. There is another form of buoyancy effect that occurs only with bodies of concave geometry in the presence of a gas phase. Even objects with a density higher than that of the liquid may float. To simulate this effect, an open box was constructed from five box primitives with material density $\rho_o = 1.5$. The open box was placed slightly above the free surface in the gas phase, with the opening pointing up (Fig. 5.13). A second basin filled with liquid above the box was realized by static obstacle cells. Water like liquid ($\nu = 10^{-6}\frac{m^2}{s}$) was simulated here in a 60^3 domain.

The buoyant force is given as the sum of the pressure forces on the particular areas of the box. Since the pressure force exerted downwards from the gas phase is lower than the upward force resulting from the higher pressure on the liquid side, the overall force compensates the gravitational force on the box and makes it float. The box sinks to the point where balance of forces is given.

Then the basin above was opened and the box slowly filled with liquid: Fig. 5.14 shows the box sinking deeper with the increasing amount of liquid in the box.

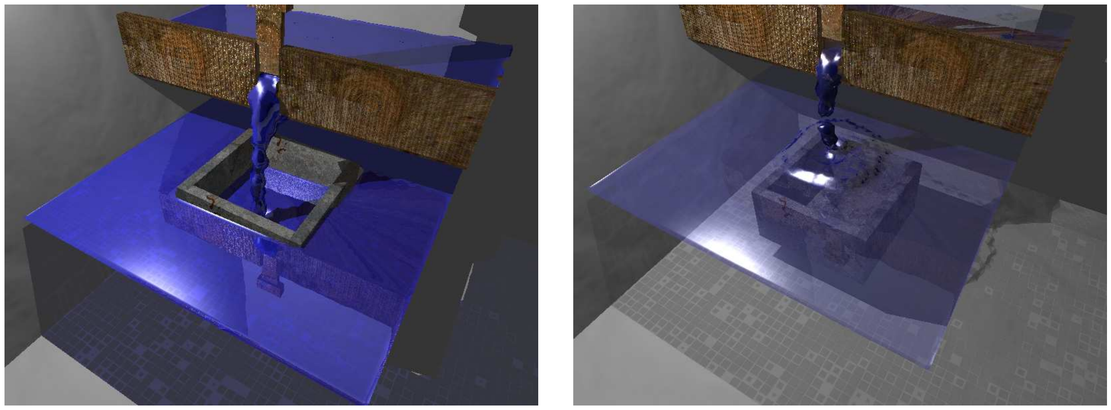


Figure 5.14. A Floating Box is filled up with liquid until it sinks.

5.4 Mass Fluctuation with Floating Objects

The moving wall boundary condition from Sec. 3.3 is known to be not strictly mass conserving but usually the mass fluctuations are neglected: At least in the case of one-phase simulations without free surface this is acceptable for applications, where strict conservation of mass is not required. However, for partially immersed objects in a free surface flow, the error is more considerable. This section explains where the mass fluctuations come from, and gives an estimation for the size of the error in the special case of a box penetrating the interface layer.

Submerging Cubes

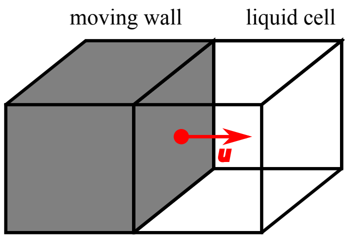


Figure 5.15. Moving wall with velocity $\boldsymbol{u} = (u_x, 0, 0)$ next to a liquid cell (left).

Figure 5.15 shows a moving wall: The liquid in the nearby cell is accelerated by adding an amount of $\Delta f_i = 6\, w_i\, \rho\, \boldsymbol{v}_i \cdot \boldsymbol{u}$ to the reflected distribution functions. Now assume that the wall moves along the x-axis, i.e. $\boldsymbol{u} = (u_x, 0, 0)$ with a constant velocity. If the front side of the wall with normal $\boldsymbol{n}_w = \boldsymbol{u}$ is considered only at one obstacle cell, one easily verifies that for the amounts Δf_i added to the reflected distribution functions the following constraint holds (only the directions TE, BE, NE, SE and E are affected in this case):

$$\begin{aligned}\sum_{i\,|\,\boldsymbol{v}_i\cdot\boldsymbol{u}>0} 6\, w_i\, \rho\, \boldsymbol{v}_i \cdot \boldsymbol{u} &= 6\,\rho\left[\frac{1}{36}\Big((1,0,1)\,\boldsymbol{u} + (1,0,-1)\,\boldsymbol{u} + (1,1,0)\,\boldsymbol{u} + (1,-1,0)\,\boldsymbol{u}\Big) + \frac{1}{18}(1,0,0)\,\boldsymbol{u}\right] \\ &= \rho\, u_x \qquad (5.1)\end{aligned}$$

This means that during one time step an amount of $\rho\, u_x$ is added to the density of the liquid cell. Now, if the wall moves $\frac{1}{u_x}$ time steps, it will have traversed exactly one cell, and the amount added due to the imposed boundary condition will be exactly ρ. This constraint is extremely important if an obstacle moves from the gas region through the free surface into the liquid phase. For a totally immersed object, one relies on that the amount of particles added to the cells at the front side of the object is about the amount subtracted from the cells at the back side. If the density around the wall is assumed as constant during the movement, mass is conserved[5.1]. For an obstacle about to penetrate the free surface, there are no liquid cells at the backside. Hence, to conserve mass if the wall moves forward one cell, exactly the amount of liquid that is lost at the conversion of the nearby cell from liquid to obstacle must be added. This is assured by Eq. (5.1), but unfortunately boxes, spheres and capsules are not just one-sided walls, and therefor additional mass is added to the system at the overall surface of sinking objects. For axis-aligned boxes entering or exiting the liquid region in a movement normal to the free surface the following lower bound can be used to estimate the amount of mass added or subtracted per time step.

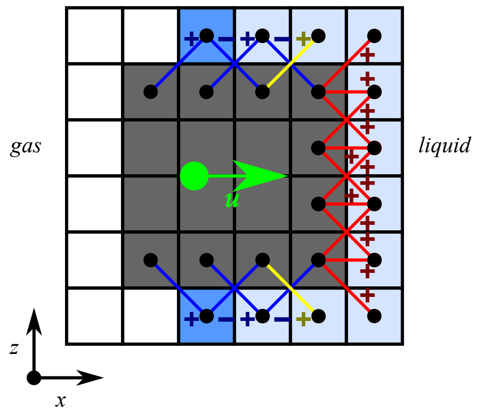


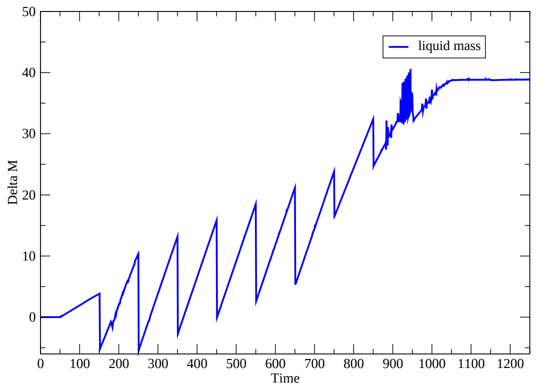
Figure 5.16. Reflected particle distribution functions that are additionally influenced by the moving wall boundary condition ($\boldsymbol{v}_i \cdot \boldsymbol{u} \neq 0$) visualized in the x-z-plane for a moving box.

Assume a box is moving with a constant velocity $\boldsymbol{u} = (u_x, 0, 0)$ like the wall before through the interface and into the liquid. Then according to Eq. (5.1) the distribution functions reflected on the front side of the box will add at every time step exactly the amount of liquid to the system that is displaced by the object. However, there are additional distribution functions reflected on the box, that are and also changed by the moving wall boundary condition (this is the case for directions i with $\boldsymbol{v}_i \cdot \boldsymbol{u} \neq 0$). In Fig. 5.16 the reflected distributions that are increased by the boundary condition where marked with a "+", decreased ones with a "-". Those reflections that do not occur on the front side, can be grouped by the liquid or interface cells they come from and were colored blue and yellow in the figure. When this is done, one either has a pair of accelerated distribution functions of different sign (blue), meaning that the cell velocity is accelerated without mass fluctuation, or a single function with a positive sign (yellow). The latter are exactly the reflections where constantly an amount of $6\, \rho \;\; w_j\, \boldsymbol{v}_j\, \boldsymbol{u} = \frac{1}{6}\, \rho\, u_x$ per time step is added to the system. Here j is one of the following directions: TE or BE (as in Fig. 5.16), NE or SE (for the y-z-plane). Thus, if the box is of height a and width b, the total amount added at each time step is

$$+\,\rho\,\frac{2}{6}\,(a+b)\,u_x. \tag{5.2}$$

If the velocity of the box is directed towards the gas phase all "+" signs change to "-" and vice versa, and the mass fluctuation is negative. Note that if the box was totally surrounded by liquid cells, there would occur additional 'yellow' lattice links with a "-" sign in Fig. 5.16, exactly compensating the amount of mass falsely added to the system.

5.1. Note that density can be identified with mass here, since the cell volume is normalized to 1.



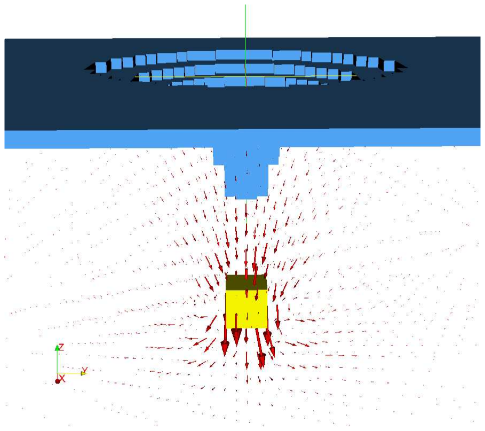


Figure 5.17. Cube submerged at a $\boldsymbol{u} = 1 \times 10^{-2}$. Left hand side shows the increase of mass until the cube is fully submerged after 1100 time steps. The graph drops instantly whenever a cell is converted from liquid to obstacle and increases linearly afterwards because of the mass added to the system by the moving wall boundary condition in every time step.

To test mass fluctuations a cubic domain was initialized with 60^3 cells, filled halfway with liquid. An axis-aligned cube of length 4 was started slightly above the interface layer and then submerged at constant velocity. Relaxation time was $\tau = 1.7$, gravitation $g = 9.81 \times 10^{-6}$. Other parameters were taken from the test case of 5.2. The total liquid mass in the domain was monitored at every time step for $\boldsymbol{u} = 1 \times 10^{-2}$, 1×10^{-3} and 1×10^{-4}. Then the experiment was reversed, and the fully immersed cube was pulled out of the liquid with the same velocity.

$\boldsymbol{u}$ movement	1×10^{-4}	1×10^{-3}	1×10^{-2}
in	+29.36	+38.20	+38.9
out	-29.36	-34.60	-39.32

Table 5.2. Mass fluctuation ΔM for a cube of length 4 moving *in* and *out* of the liquid at different velocities.

The error increases with higher velocities. For slow velocities $|\Delta M|$ was equal whether the cube was submerged or pulled out of the liquid. As soon as the interface layer is closed behind the object, the mass fluctuation stops up to compressibility errors due to the density gradient around the moving object. During the time frame in which the cube is passing through the interface (for $\boldsymbol{u} = 10^{-2}$ from time step 250 to 650 in Fig. 5.17) one has

$$\frac{\Delta M(t + \frac{1}{u_x}) - \Delta M(t)}{\frac{1}{u_x}} \approx \frac{8}{3} u_x,$$

so the mass fluctuation is very close to the lower bound given by Eq. (5.2) for $\rho = 1$ and $a = b = 4$ in this case. Before and after this time frame, interface cells are penetrated or restored as described in Sec. 3.4.1, which causes additional deviations in the mass balance.

Tangential Movement

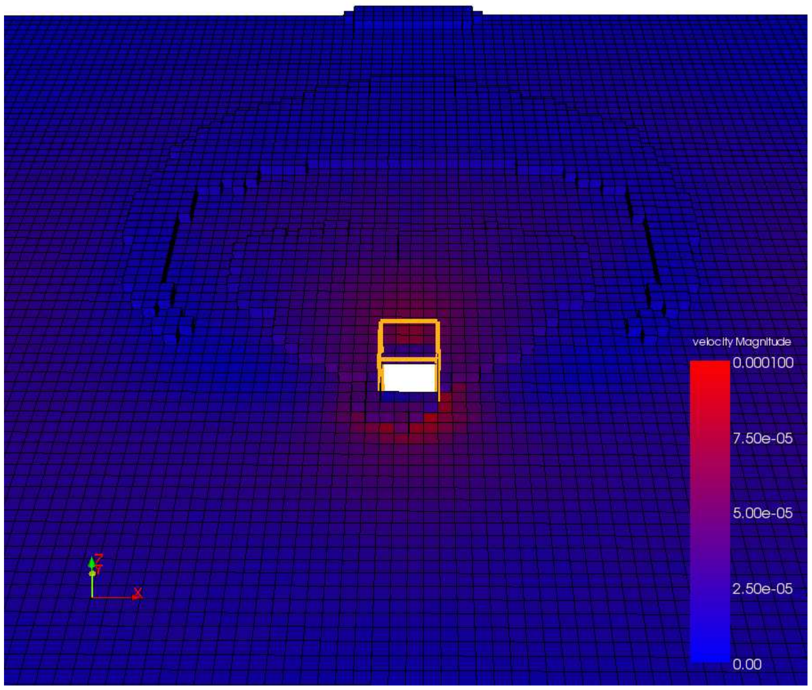


Figure 5.18. Bow wave of box moving tangential to the interface layer along the y-axis. Picture of the $100 \times 100 \times 60$ domain with a box velocity of $u = 10^{-4}$ (time step 50.000). Cells colored by macroscopic flow velocity.

The effect of mass fluctuation will also arise if an object moves tangential to free surface. Analogously to the preceding paragraph a series of tests was performed with a box of lengths $4 \times 4 \times 8$ moving at a constant velocity of 10^{-4}, 10^{-3} and 10^{-2}, respectively. The domain of 60^3 cells from above was reused for the first 3 runs, with the box moving along a straight line from $(30, 25, 30)$ to $(30, 35, 30)$. Then the domain was resized to $100 \times 100 \times 60$, and the box moved along the line from $(50, 40, 30)$ to $(50, 60, 30)$. The initial interface position was in height $z = 30$ for all cases.

As shown in Fig. 5.18, a bow wave emerges with the movement of the box. Thus the interface layer in direct neighbourhood of the box is not plane most of the time, and similar effects as in the above paragraph arise, since the number of distribution functions adding mass and the number of distribution functions subtracting mass will differ. The unpredictable advection of the interface around the object thus makes general statements about the mass balance difficult. Furthermore, the mapping algorithm (Sec. 3.4.1) hast to restore interface cells at the back side of the box, which is more problematic than the generation of liquid cells, since two mass-related variables (density and fill level) have to be approximated.

Generally, the mass deviations with tangential object movement at the free surface are less drastic than in the case of normal object velocity. A look at Fig. 5.19 might give the impression of an oscillating behaviour around zero for $u = 0.0001$ after the cell has covered a distance of 5 cells. However, the mass balance in the $100 \times 100 \times 60$ domain (Fig. 5.20) for $u = 0.0001$ shows that this behaviour is accidental and caused by the wall influence of the smaller domain. In the larger domain, the slow velocity even yielded the worst result in respect to mass balance. This, on the other hand, implies that mass fluctuation is not a question of object velocity. In the larger domain, the test case was repeated, with mass interpolation instead of fill levels for restored cells. However, no general evaluation in favor of one of the two methods can be made in this case, since the situation was slightly improved for $u = 0.0001$ only, but notably worsened at the higher velocities.

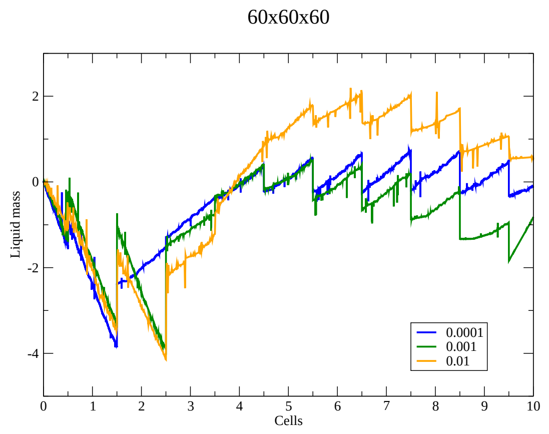
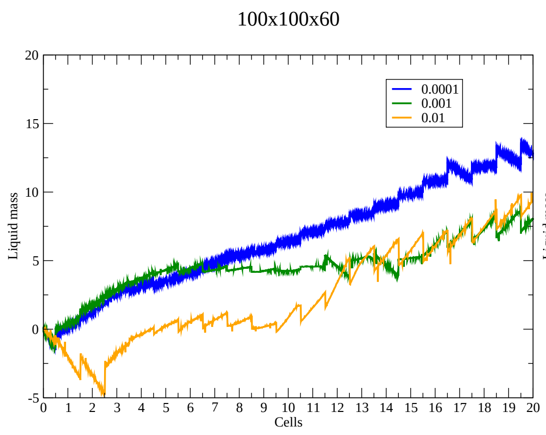


Figure 5.19. Mass fluctuation for tangential movement of a box at different velocities.



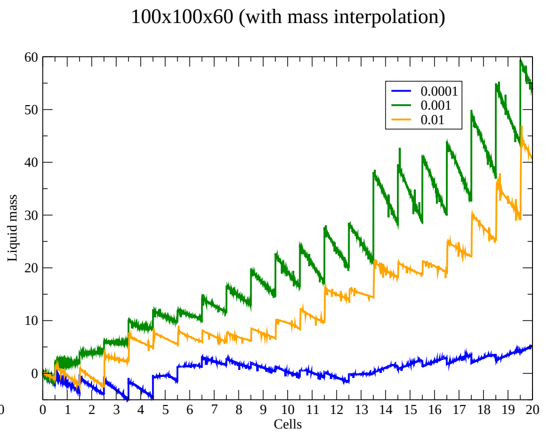


Figure 5.20. Mass fluctuation for tangential movement of a box at different velocities, with fill level interpolation (left) and mass interpolation, respectively.

Chapter 6
Conclusion

A simulation model for rigid bodies in free surface flow has been presented. The approach is tailored specific to process three-dimensional scenarios with freely moving objects under the influence of gravity, buoyancy, drag forces and advection. In this model the liquid phase is simulated with a lattice Boltzmann extension for free surface flows given by [KT04], which models a two-phase free surface flow with a gas as second phase. The flow is assumed to be governed by the fluid dynamics of the liquid such that the gas phase must only be taken into account by its pressure. We devoted one chapter to the mathematical background of the lattice Boltzmann based approach of fluid simulation: A step by step derivation based on [HL97, Luo98] from the continuous Boltzmann equation is presented, including the external body force for gravity. To combine the fluid simulation with moving objects, an approach based on [Lad93, Lad07] has been adopted and extended to the free surface model. This approach allows the calculation of forces exerted by the fluid on rigid bodies. The influence of body movement on the fluid, in turn, is included by applying appropriate boundary conditions at the surface of the objects. The free surface model needs a closed interface layer tracking the free surface position. Therefor a method to incorporate moving objects into the interface model has been presented, that allows the penetration of the liquid surface by objects.

The model was successfully implemented into the waLBerla software framework [GDF+07a, GDF+07b], coupled with the P.E. physics engine [Igl07] to handle the rigid body dynamics. Also a method to initialize the liquid phase with a density gradient according to gravitational pressure was provided.

As a result it was possible to simulate drag forces and buoyant forces on rigid bodies moving through a fluid medium. This has been verified for the case of the simple geometry of a sphere moving through a liquid at slow velocity. The influence of the discretization error of the mapping of the object shape to lattice grid points on the force calculation was examined: Both the drag force and the buoyant force suffered from fluctuations according to the discretization error, and affected the body dynamics. Whereas the drag force calculations improved under the linear interpolation scheme for curved boundaries according to [BFL01], the buoyant force calculation could not be improved in that way. Here, in presence of the gravitational pressure gradient the problem arose that an error in the fluid momentum of order of the external force is introduced at the curved boundaries. There are other approaches for curved boundary treatment with lattice Boltzmann, e.g. [YMLS03] or [FH98]. The latter interpolate is not between two fluid nodes, which could yield an improvement in respect to the problem with gravity. However, this has not been verified in this thesis. Local grid refinement methods will also improve the accuracy at curved obstacles. A survey is given in [YMLS03]. But these methods are computationally expensive and also more complex to implement. For the effect of a floating object, the balance of gravitational and buoyant force is most important. We were able to achieve this, by simply adjusting the calculation of the gravitational force on an object to its discrete shape. It is also possible to precalculate the volume error for an object of a certain size for a given path of movement. This can be used to find grid resolutions that minimize the discretization error.

The advection of objects floating in a free surface flow can be simulated successfully. Dynamic forces induced by motion of objects were tested: A propelling force arose from the rotation of a paddle wheel. We also showed that the presented model is capable to simulate buoyancy of concave objects resulting from the pressure difference between gas and liquid phase.

The major drawback of the method is that mass is not conserved with floating objects reaching through the interface between gas and liquid. It could be shown that the mass fluctuation depends to a significant part on the boundary condition imposed at the surface of the moving obstacles. For a partially immersed object the balance of mass added and subtracted by the boundary condition is no more given. This raises the question whether it was possible to reduce this error by additional local boundary treatment in this case. But the answer is likely to be negative: Even for the simple case of a cube that moves perpendicular to a plane liquid surface, the distribution functions causing the mass defect were not necessarily located at the interface layer. An approach to even out the mass fluctuations was given by [Thü07], and consists in calculating the mass added or subtracted due to the movement of an object for each time step. Then a mass defect can be balanced by adding or subtracting mass at the mass diminishing or mass generating boundaries, respectively. Howevre, a model of true mass exchange, similar to the one used for interface cells, in liquid cells at the obstacle surface would need significantly larger implementation efforts.

Nevertheless, the presented method yields convincing results for floating objects with a small size in relation to the simulation domain, and can be used for applications that do not need a strict conservation of mass.

Bibliography

[AMH02] T. Akenine-Möller and E. Haines. *Real-time Rendering (Second Edition)*. AK Peters, 2002.

[BFL01] M. Bouzidi, M. Firdaouss, and P. Lallemand. Momentum transfer of a boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13(11), 2001.

[BG00] J.M. Buick and C.A. Greated. Gravity in a lattice boltzmann model. *Phys. Rev. E*, 61(5), 2000.

[Fei05] C. Feichtinger. Drag force simulations of particle agglomerates with the lattice-boltzmann method. Master's thesis, University of Erlangen-Nuremberg, Lehrstuhl für Informatik 10 (Systemsimulation), Institut für Informatik, 2005.

[FH98] O. Filippova and D. Hänel. Boundary fitting and local grid refinement for lattice-bgk models. *Int. J. of Modern Phys. C*, 9(36), 1998.

[GDF+07a] J. Götz, S. Donath, Ch. Feichtinger, K. Iglberger, and U. Rüde. Concepts of walberla prototype 0.0. Technical report, 2007.

[GDF+07b] J. Götz, S. Donath, Ch. Feichtinger, K. Iglberger, and U. Rüde. Concepts of walberla prototype 0.1. Technical report, 2007.

[GFI+08] J. Götz, C. Feichtinger, K. Iglberger, S. Donath, and U. Rüde. Large scale simulation of fluid structure interaction using lbm and the physics engine pe. Lehrstuhl für Informatik 10 (Systemsimulation), Institut für Informatik, University of Erlangen-Nuremberg, 2008.

[GL88] R.B. Guenther and J.W. Lee. *Partial Differential Equations of Mathematical Physics and Integral Equations*. Prentice Hall, 1988.

[GS03] I. Ginzburg and K. Steiner. Lattice boltzmann model for free-surface flow and its application to filling process in casting. *Journal of Comp. Physics*, 185, 2003.

[HL97] X. He and L.-S. Luo. Theory of the lattice boltzmann method: From the boltzmann equation to the lattice boltzmann equation. *Phys. Rev. E*, 56(6), 1997.

[Hän04] Dieter Hänel. *Molekulare Gasdynamik*. Springer, 2004. german.

[Igl05] K. Iglberger. Lattice boltzmann simulation of flow around moving particles. Master's thesis, University of Erlangen-Nuremberg, Lehrstuhl für Informatik 10 (Systemsimulation), Institut für Informatik, 2005.

[Igl07] K. Iglberger. Doxygen documentation on physics engine pe. Technical report, 2007.

[KT04] C. Körner and M. Thies. Lattice boltzmann model for free surface flow. Material Science and Technology WTM, Universität Erlangen-Nürnberg, 2004.

[Lad93] A.J.C. Ladd. Numerical simulations of particulate suspensions via a discretized boltzmann equation part i. theoretical foundation. Lawrence Livermore National Laboratory, 1993.

[Lad07] A.J.C. Ladd. Numerical simulations of particulate suspensions via a discretized boltzmann equation part ii. numerical results. Lawrence Livermore National Laboratory, 2007.

[LL03] P. Lallemand and L.-S. Luo. Lattice boltzmann method for moving boundaries. *J. Comp Phys.*, 184, 2003.

[Luo98] L.-S. Luo. Unified theory of lattice boltzmann models for nonideal gases. *Phys. Rev. Lett.*, 81(8), 1998.

[Luo00] L.-S. Luo. Theory of the lattice boltzmann method: Lattice boltzmann models for nonideal gases. *Phys. Rev. E*, 62(4), 2000.

[MYSL02] R. Mei, D. Yu, W. Shyy, and L.-S. Luo. Force evaluation in the lattice boltzmann method involving curved geometry. *Physical Review E.*, 65, 2002.

[oVPL04] Persistence of Vision Pty. Ltd. Persistence of vision (tm) raytracer, 2004. `http://www.povray.org`.

[Poh07] T. Pohl. *High Performance Simulation of Free Surface Flows Using the Lattice Boltzmann Method*. PhD thesis, University of Erlangen-Nuremberg, 2007.

[Sch99] Franz Schwabl. *Statistische Mechanik*. Springer, 1999. german.

[Suc01] Sauro Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford Science Publications, 2001.

[Thi05] M. Thies. *Lattice Boltzmann Modeling with Free Surfaces Applied to Formation of MetalFoams*. PhD thesis, University of Erlangen-Nuremberg, 2005.

[Thü02] N. Thürey. A single-phase free-surface lattice boltzmann method. Master's thesis, University of Erlangen-Nuremberg, Lehrstuhl für Informatik 10 (Systemsimulation), Institut für Informatik, 2002.

[Thü07] N. Thürey. *Physically based Animation of Free Surface Flows with the Lattice Boltzmann Method*. PhD thesis, University of Erlangen-Nuremberg, 2007.

[YMLS03] D. Yu, R. Mei, L.-S. Luo, and W. Shyy. Viscous flow computations with the method of lattice boltzmann equation. *Progress in Aerospace Sciences*, 39, 2003.

[ZB08] J. Zierep and K. Bühler. *Grundzüge der Strömungslehre*. Teubner, 2008. german.