Fluid Flow Simulation and Optimisation with Lattice Boltzmann Methods on High Performance Computers

Application to the
Human Respiratory System

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN

von der Fakultät für Mathematik der Universität Karlsruhe (TH) genehmigte

DISSERTATION

von

Dipl.-Math. oec. Mathias J. Krause

aus Görlitz

Tag der mündlichen Prüfung: 14. Juli 2010

Referent:	Prof. Dr. Vincent Heuveline
Korreferent:	Prof. Dr. Michael Junk
Korreferent:	PD Dr. Gudrun Thäter

Vorwort

Diese Dissertationsschrift entstand im Rahmen meiner Tätigkeit am Institut für Angewandte und Numerische Mathematik, Lehrstuhl IV (NumHPC) und am Steinbuch Centre for Computing (SCC). Beide Einrichtungen gehören dem Karlsruher Institut für Technologie (KIT) an, welches der Nachfolger der Universität Karlsruhe (TH) ist. Der Anstoß zu dieser Arbeit fand sich durch das Mitwirken an den zwei Projekten *OpenLB* (vgl. Appendix A) und *United Airways* (vgl. Appendix B) sowie am Sonderforschungsprogramm 1253 der *Deutschen Forschungsgemeinschaft* (DFG). Das erstgenannte Projekt bietet in einer freien Softwarebibliothek die Umsätzung von Lattice Boltzmann Methoden (LBM) zur Lösung von, unter anderen, Strömungsproblemen an, wie sie z.B. im Rahmen des United Airways Projekts zur numerischen Simulation der menschlichen Atmung eingesetzt werden. Das SPP 1253 widmet sich der Optimierung mit Partiellen Differentialgleichungen, was mich dazu inspirierte, Numerische Lösungsverfahren für Optimierungsprobleme unter Anwendung von LBM zu entwickeln.

Ich möchte all jenen danken, die während dieser Zeit zum Gelingen meiner Arbeit beigetragen haben.

Mein besonderer Dank gilt Herrn Prof. Dr. Vincent Heuveline, der diese Arbeit erst ermöglicht hat, sowie Herrn Dr. Jonas Latt und Frau PD Dr. Gudrun Thäter für die vielen wertvollen und hilfreichen Anregungen und die immerwährende Unterstützung in sämtlichen Angelegenheiten. Mein weiterer Dank gilt Herrn Prof. Dr. Michael Junk für die nützlichen Hinweise und die fruchtbaren Gespräche. Den Mitarbeitern des Institutes danke ich für die angenehme Atmosphäre und die hervorragende Zusammenarbeit.

Meine Familie, meine Freunde und meine Freundin Juliana möchte ich für die seelische, geistige und moralische Unterstützung, die Geduld und die wunderbare Zeit, die ich mit Ihnen verbringen durfte, an dieser Stelle in besonderem Maße würdigen.

Herzlichen Dank Euch allen!

Karlsruhe im Juni 2010 Mathias J. Krause

Preface

This dissertation originates from my work at the Institut für Angewandte und Numerische Mathematik, Lehrstuhl IV (NumHPC) and at the Steinbuch Centre for Computing (SCC). Both institutions are part of the Karlsruhe Institute of Technology (KIT) which is the successor of the Universität Karlsruhe (TH). The presented work was initially inspired through my participation in the priority program 1253 of the *Deutsche Forschungsgemeinschaft* (DFG) as well as two projects. The aim of the first project *United Airways* (cf. Appendix B) is to simulate human respirations numerically. The second project *OpenLB* (cf. Appendix A) is an open source library which enables solving e.g. fluid flow problem by means of lattice Boltzmann methods (LBM). The subject of the DFG priority program is optimisation with partial differential equations, which motivated me to develop numerical strategies to solve optimisation problems by applying LBM.

I want to thank all who were supporting my work during that time.

I express my special thanks to Prof. Dr. Vincent Heuveline, who made this work possible in the first place. Further, I am very grateful to Dr. Jonas Latt and PD Dr. Gudrun Thäter for their many valuable suggestions and their unending support in all aspects related to my work. My thanks also goes to Prof. Dr. Michael Junk who gave me many hints in our fruitful discussions. Equally, I thank all my colleagues for the excellent working atmosphere as well as rewarding collaborations.

In a special way, I want to honor my family, my friends and my girlfriend Juliana. I am grateful for their mental and moral support, for their patience and for the wonderful time I have spent with them.

Many thanks to all of you!

Karlsruhe, June 2010 Mathias J. Krause

Contents

Introduction

Clearly, the main function of the human respiratory system is to enable the exchange of gas with the blood circuit. Oxygenated air is inhaled through the nose or mouth and transported to the lungs. There, the oxygen is passed to the blood circuit in which the oxygen is exchanged for carbon dioxide. Then, the carbon dioxide-enriched air is exhaled. The permanent supply of oxygen is vitally important. A rule of thumb states that a human can survive about three weeks without food, three days without water but just three minutes without air.

Few details are presently known about this roughly described breathing process which also comprises other functionality like the humidifying, cleaning or warming of inhaled air. An extensive research of the highly complex system is required in order to explain the many occurring disorders of the human respiratory system. A vast field of applications would benefit from a deeper understanding of this system. For example, being able to predict the possible implications on the respiratory tract due to surgery or environmental impact has the potential to greatly improve health care in this domain.

The full description of the human respiratory system is an enormous challenge. The difficulties are not only related to the complex geometry but also to the highly complex multi-physics phenomenology involving multi-scale features. Considering the transport and exchange of gas as significant factor in order to obtain a full description, the research field of *fluid dynamics* is adequate to provide some answers. Driven by the rapid increase of available computing power in recent years especially in the context of *high performance computing* (HPC), the relevance of *computational fluid dynamics* (CFD) for medical research and development in this area has grown steadily. Nowadays, numerical simulations help accelerate the research progress. For example, numerical simulation is used to extend experimental-based studies or provide new insights where experiments are not feasible. Notwithstanding the great achievements in recent years, the numerical simulation of the full human respiratory system corresponds to one of the *grand challenges* in scientific computing.

Some aspects of this *grand challenge* are taken on in this thesis. They are formulated by means of two main aims. The first one is dedicated to invent numerical strategies to simulate human respiratory flows efficiently and accurately. Thereby, the proposed approaches are meant to enable other scientists to get a better fundamental understanding of the full respiratory functionality but also to accelerate the progress towards patient-specific treatment strategies. The latter issue is also related to the second main aim of this thesis, namely the integration of mathematical optimisation strategies in order to control or optimise respiratory flows.

In the last two decades, *lattice Boltzmann methods* (LBM) have become a matured technique in the context of CFD. The simplicity of the core algorithms as well as the locality properties resulting from the underlying kinetic approach lead to methods which have already shown to be very attractive in at least two respects. One of them is the simulation of flows with underlying complex computational domains and the other is related to parallelism. Hence, they seem well suited to reach the posed goals. However, aspects related to fluid flow control and optimisation taking advantage of LBM are barely discussed in the literature and have by no means been deeply analysed. Therefore, this thesis strongly focuses on the development of a general framework in this context.

The overall strategy to realistically solve the mentioned problems relies on an integrative approach consisting of three pillars, namely numerical simulation, high performance computing and mathematical optimisation which are all based on a mesoscopic model governed by the Boltzmann equation or its simplification, the BGK-Boltzmann equation (cf. Figure 0.1). It is taken advantage of LBM which provide the necessary discretisation strategies for the numerical solutions of these equations.

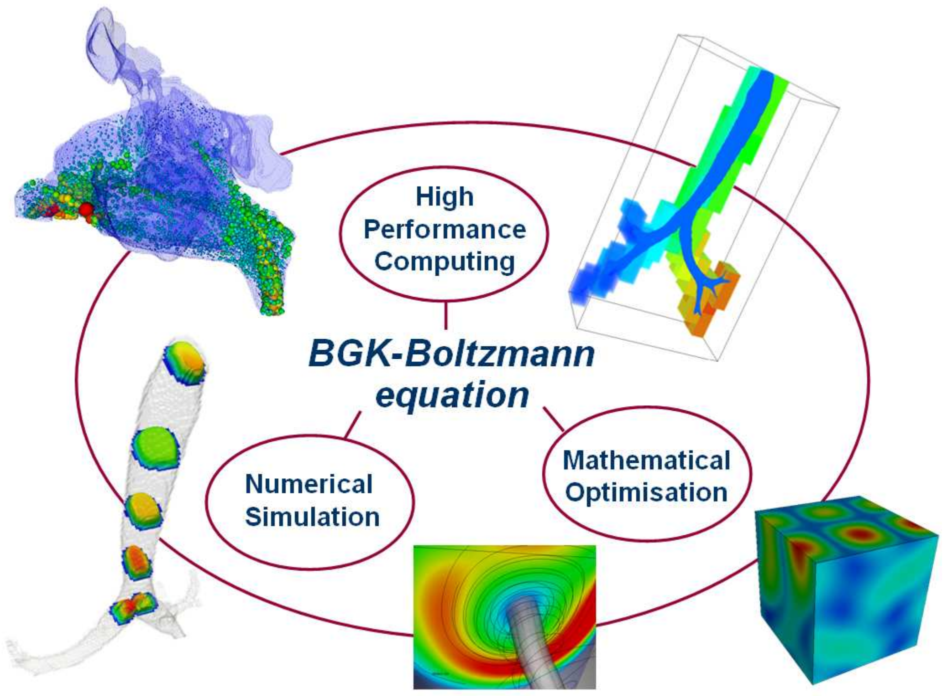


FIGURE 0.1. Integrative strategy and a selection of its applications considered within this work. Challenges concerning the simulation of flows in the human respiratory system are met by the combined application of numerical simulation, high performance computing and mathematical optimisation techniques based on a mesoscopic model, which is governed by the BGK-Boltzmann equation.

The integrative strategy and its applications are reflected in the organisation of this thesis. The main part of the thesis is organised as follows:

In the first chapter, two physical models both dedicated to describe the dynamics of incompressible Newtonian fluids are introduced and compared to each other. The first is a macroscopic model governed by the incompressible Navier-Stokes equation and the second is a mesoscopic model governed by the Boltzmann equation or its simplification, the BGK-Boltzmann equation. Emphasis is placed on accenting the connection between and the incompressible Navier-Stokes equation a family of BGK-Boltzmann equations in a regime of small Knudsen and Mach numbers.

In Chapter 2, lattice Boltzmann methods are introduced in the light of discretisation techniques for families of BGK-Boltzmann equations. This is in contrast to many other approaches where they are considered in connection with the incompressible Navier-Stokes equation. Afterwards, topics related to an efficient implementation strategy of LB algorithms are addressed. Two numerical experiments complete the chapter. The first example to show the connection of LBM to the incompressible Navier-Stokes equation. For this purpose, an analytical stationary solution of a particular incompressible Navier-Stokes equation is given which is to be reached by applying certain LB schemes. The second numerical experiment is dedicated to solve the famous benchmark problem *lid-driven cavity* (LDC). This is realised by modelling the problem both macroscopically and mesoscopically and solving the resulting governing equation by a *finite element method* (FEM) and an LBM, respectively. Both solutions are finally compared with experimental data.

A hybrid parallelisation concept especially developed for LBM and its realisation is the subject of Chapter 3. The proposed strategy allows coping with platforms sharing the properties of both shared and distributed architectures. The approach relies on spatial domain decomposition where each domain represents a basic block entity which is solved on a *symmetric multiprocessing* (SMP) system. Its realisation is finally evaluated by means of performance results obtained for two three-dimensional test cases. The first one is a problem with an underlying simple geometry, namely the benchmark problem LDC. The second example is characterised by the comparatively complex computational domain of an upper part of the human lungs.

Chapter 4 and Chapter 5 are both dedicated to developing general solution strategies to numerically solve fluid flow control and optimisation problems where the side conditions are governed by a BGK-Boltzmann equation. Both approaches are gradient-based and hence require the computation of certain derivatives. However, they differ fundamentally in the way that the derivatives are obtained.

In the first of the two chapters, the application of a *first-discretise-then-optimise* strategy based on *automatic differentiation* AD techniques is proposed. Details regarding the implementation of this approach for parallel use are discussed before the realisation is tested for a parameter identification, respectively distributed control problem. The numerical results are verified and the parallel performance is analysed.

The approach presented in Chapter 5 relies on the *first-optimise-then-discretise* ansatz and is based on obtaining the required derivatives by means of solutions of adjoint problems. In this chapter, a necessary condition for an optimum of the continuous optimisation problem is derived by applying Lagrange's formalism. Further, the *adjoint BGK-Boltzmann equation* is derived as the governing equation of a prototypical adjoint problem. Then, methods similar to LBM are proposed to discretise the adjoint BGK-Boltzmann equation equation. Afterwards, aspects concerning its parallel realisation are addressed. At the end of the chapter, the distributed control problem formulated in Chapter 4 is considered as a test case to numerically verify the realisation and to test its parallel performance. Furthermore, both the numerical and the performance results are compared to those obtained by applying the AD-based strategy.

In Chapter 6, the focus is placed on the numerical simulation of the respiration in two parts of the respiratory tract, namely the upper human lungs and the nose. The underlying geometry data are obtained by computer tomography (CT) scans of a patient with a diagnosed ventilation disorder. Hence, secondary objective is, to discover a pathology which might be responsible for this disorder by the undertaken numerical simulations. At first, a concept for a complete preprocessing dedicated to simulate fluid flows in complex geometries applying LBM is presented. Its realisation is illustrated by means of extracting the computational domains of the lungs and nose from the mentioned CT data. The flow of air in the human nose is simulated for two test configurations. The first one aims to establish numerical evidence for pseudo steady states and, furthermore, to validate the results by means of a comparison with numerical results obtained by others as well as experimentally obtained data for other similar geometries. The second test suite is formulated in a way which enables a comparison with measurements obtained for the actual considered patient. Finally, a feasibility study is presented. The considered problem is an exhalation at a fixed flow rate in the upper part of the human lungs. The discussed approach is meant to establishing the basis for a two-scale model describing the respiration in the complete human lungs.

The last chapter of this work is dedicated to summarise the obtained results and address open questions both stimulating further research. Finally, the bibliography provides references for all sources relevant to this thesis. An appendix provides information related to two associated projects, namely *OpenLB* and *United Airways*. At the very end of this work, one finds important frequently used abbreviations, fluid constants, parameters and variables, norms and spaces as well as differential operators collected in a nomenclature.

CHAPTER 1

Modelling Fluid Flows: A Mesoscopic and Macroscopic View

The motion of liquids and gases is observed, investigated and modelled in the research field of *fluid dynamics*. Experiments build an empirical basis for the models, i.e. measurements are taken to discover correlations of various quantities of the fluids. These observations manifest basic model assumptions which finally lead to equations governing the model itself. Depending on the scale where the observations take place and, consequently, the level where the model assumptions are formulated, one can distinguish *macroscopic*, *mesoscopic* and *microscopic models* for the description of fluid flows.[1] In Table 1.1 this classification of fluid flow models is characterised and examples for models, observed properties and numerical methods used to solve corresponding equations are given.

model type	macroscopic	mesoscopic	microscopic
characteristic model assumption	interaction of molecules in the fluid neglected	distribution of molecules in the fluid considered	interaction of single molecules in the fluid considered
examples: models	Navier-Stokes, Euler, Stokes, Heat equation, ...	Liouville, Boltzmann, BGK-Boltzmann equation, ...	molecular dynamics (Newton's laws)
examples: observed quantities	fluid velocity, pressure, density, temperature, ...	mean free path, mean molecular velocity, density, ...	molecular mass, velocity, extent and form, ...
examples: numerical methods	spectral, finite differences, volumes and elements, ...	Monte Carlo, lattice Boltzmann, finite differences, volumes and elements, ...	molecular dynamics

Table 1.1. A Classification of models to describe fluid flows. Examples for models, considered fluid quantities are given for each class.

Assuming that models of different classes are consistent, the classification can be seen in a hierarchical manner in the sense that models based on small scales are more general than those based on large scales. That means in particular that the correlation of macroscopic flow quantities can be explained by mesoscopic or microscopic models but not vice versa. However, in practice this idealised picture of consistent models is often not given or not obvious. Furthermore, if relations of

[1]The here presented classification for fluid flow models is not complete. Models based on observations in even smaller scales exist, e.g. quantum mechanical models.

models exist it can be hard to prove them with mathematical rigour. An example in that context is the relation of the mesoscopic model for rarefied gases to macroscopic models for viscous flows - in particular, the correlation of the Boltzmann equation or its simplifications to the incompressible Navier-Stokes equation. These two models are in the focus of the explanations to come in this chapter.

The first two sections of this chapter are of introductory character. In particular, a notation is established and elementary equations are recorded to which the explanation in all other chapters refer. In Section 1.1 model assumptions are presented for both macroscopic models in general and an incompressible Newtonian fluid flow model in particular. Based on them, conservation equations are derived. This finally leads to the incompressible Navier-Stokes equation. In Section 1.2 a mesoscopic model for rarefied gases is introduced. As for the macroscopic model, at first, the model assumptions are formulated. Afterwards, as governing equation for this model the Boltzmann equation is derived. A bridge between the "mesoscopic" and "macroscopic world" is built in the last part of this chapter within Section 1.3. Thereto, at first some fundamental properties of the Boltzmann equation are stated. Then, the BGK-Boltzmann equation is considered. This equation is a simplification of the Boltzmann equation which preserves its substantial properties. Finally, a family of BGK-Boltzmann equations is considered in a regime of small Knudsen and Mach numbers. It turns out that in a limiting process solutions of such families are connected with the incompressible Navier-Stokes equation.

1.1. Macroscopic Approach: A Model for Incompressible Newtonian Fluids

In the context of continuum mechanics the underlying micro-structure of a fluid, i.e. the whereabouts and interactions of atoms or molecules, respectively, is not considered and therefore not modelled. Instead, macroscopic quantities are related to each other. The relations are established phenomenologically by means of observation found to hold for certain fluids. This finally leads to a system of the model governing equations.

In the following, exclusively such viscous fluids are considered which are said to be *incompressible Newtonian fluids*. For that class of fluids a macroscopic model can be derived which is governed by the *incompressible Navier-Stokes equation*. To build the model, at first in Subsection 1.1.1 the required terminology and assumptions are stated. Afterwards, in the following subsections the incompressible Navier-Stokes equation is derived. The presented derivation aims to emphasise the assumptions needed to establish the model, whereas questions regarding the existence and uniqueness of a solution are not discussed. The books of Feistauer [**36**] and Batchelor [**13**] and the derivations Baumann presented in [**14**] serve as guideline for the contents of this section.

1.1.1. Terminology and Model Assumptions. The dynamics of a fluid is considered in a domain $\Omega \subseteq \mathbb{R}^d$ $(d = 2, 3)$ over a time $I = [t_0, t_1] \subseteq \mathbb{R}$, $0 \leq t_0 < t_1 < \infty$. In the here presented model, the motion is described by the two following macroscopic quantities:

- velocity

$$\boldsymbol{u} : \begin{cases} I \times \Omega & \to \mathbb{R}^d \\ (t, \boldsymbol{r}) & \mapsto \boldsymbol{u}(t, \boldsymbol{r}) \ , \end{cases} \tag{1.1}$$

- pressure

$$p : \begin{cases} I \times \Omega & \to \mathbb{R} \\ (t, \boldsymbol{r}) & \mapsto p(t, \boldsymbol{r}) \ . \end{cases} \tag{1.2}$$

A fundamental assumption postulated for continuum mechanical models is the *continuum hypothesis*. Under this premise, a fluid is seen as a substance that fills a considered domain with an arbitrary small volume completely. This premise implies the negligence of the interaction of single atoms or molecules. From a mesoscopic point of view, the macroscopic quantities can be interpreted as averages obtained from the distribution of n particles (cf. Subsection 1.2.2). With the continuum hypothesis the averages are considered in the limit $n \to \infty$. Thus, this picture suggests the consequence that in the limit all macroscopic quantities can be assumed to be smooth in Ω and I. This is in accordance to observations as long as the considered volumes where the averages are measured are large enough (cf. Batchelor [**13**, p. 5, Figure 1.2.1]). The term "large enough" can be quantified by the *Knudsen number Kn* which is defined as the quotient of a typical mesoscopic and a typical macroscopic length scale (cf. Definition (1.69)). In this context, the continuum hypothesis can be seen as the limit case $Kn = 0$.

A characterising property of *incompressible fluids* is that the mass density $\rho \in \mathbb{R}_{>0}$ is constant in Ω and I. As well as the continuum hypothesis the assumption of incompressibility is an idealisation and phenomenologically founded. It is observed (cf. Batchelor [**13**, p. 168]) that steady fluid flows with solely small measured macroscopic flow velocities show only little variations of their mass density. Here, the term "small" can be quantified by comparing a typical macroscopic to a typical mesoscopic speed. The quotient defines the *Mach number Ma* (cf. Definition (1.70)). In this framework, the incompressibility hypothesis corresponds to the limit case $Ma = 0$.

Another postulate is that the *dynamic viscosity* $\mu \in \mathbb{R}_{>0}$ is a constant in Ω and I as well. Further, a restriction is made to fluids for which it is observed that their *strain rate tensor*

$$\boldsymbol{D} : \begin{cases} I \times \Omega & \to \mathbb{R}^{d \times d} \\ (t, \boldsymbol{r}) & \mapsto \boldsymbol{D}(t, \boldsymbol{r}) := \frac{1}{2} \left(\boldsymbol{\nabla_r} \otimes \boldsymbol{u} + (\boldsymbol{\nabla_r} \otimes \boldsymbol{u})^T \right) (t, \boldsymbol{r}) \end{cases} \tag{1.3}$$

is linear to the stress tensor $\boldsymbol{P} \in \mathbb{R}^{d \times d}$. This property characterises so-called *Newtonian fluids*. Newton's hypothesis is broadened and finally formulated in the *Stokes' postulates*, which are:

(1) $\boldsymbol{P} = -p\boldsymbol{I_d} + \boldsymbol{L}(\boldsymbol{D})$ with a linear continuous function $\boldsymbol{L}$,
(2) the fluid is isotropic, i.e. especially that $\boldsymbol{L}(\boldsymbol{S}\boldsymbol{D}\boldsymbol{S}^T) = \boldsymbol{S}\boldsymbol{L}(\boldsymbol{D})\boldsymbol{S}^T$ for all transformations $\boldsymbol{S} \in \mathbb{R}^{d \times d}$ with $\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{I_d}$ and $det(\boldsymbol{S}) = 1$,

where $\boldsymbol{I_d} \in \mathbb{R}^{d \times d}$ denotes the identity matrix.

The two parameters ρ and μ characterise the material properties of the considered fluid. They are both assumed to be given in advance. With them and with the three principles of mechanics, namely

(1) conservation of mass,
(2) conservation of linear momentum,
(3) conservation of angular momentum,

the governing equation of the model for incompressible Newtonian fluids can be derived. To be mentioned is that the latter principles basically rely on Newton's observations manifested in his famous first and second law.

In the following three subsections, namely Subsection 1.1.2 to Subsection 1.1.4, the three mentioned principles of mechanics are derived. In Subsection 1.1.5 Stokes' postulates are motivated. Finally, the section is completed by Subsection 1.1.6 where the incompressible Navier-Stokes equation is stated.

1.1.2. Conservation of Mass. The law of conservation of mass states that the mass of a considered fluid in a closed system will remain constant. For a general derivation of the equation which describes the conservation law, the mass density ρ is understood as a function of time and space, i.e. the restriction to incompressible fluids is dropped momentarily. Then, for $B \subseteq \Omega$ and $t \in I$ the mass $m(B,t)$ is given by

$$m(B,t) := \int_B \rho(t,\boldsymbol{r})\ d\boldsymbol{r}\ . \tag{1.4}$$

In the here considered context, B does not represent a closed system in the sense that through the boundary ∂B mass can be gained or lost during a considered time interval $[t_0', t_1'] \subseteq I$. The total win minus loss of mass in B during $[t_0', t_1']$ is

$$\begin{aligned} m(B,t_1') - m(B,t_0') &= \int_B \rho(t_1',\boldsymbol{r})\ d\boldsymbol{r} - \int_B \rho(t_0',\boldsymbol{r})\ d\boldsymbol{r} \\ &= \int_{t_0'}^{t_1'} \int_B \frac{\partial}{\partial t}\rho(t,\boldsymbol{r})\ d\boldsymbol{r}dt\ . \end{aligned} \tag{1.5}$$

The mass flux through ∂B that enters and leaves B during $[t_0', t_1']$ is obtained as the following difference

$$\int_{t_0'}^{t_1'} \int_{\partial B} -\boldsymbol{n}(\boldsymbol{r}) \cdot \boldsymbol{u}(t,\boldsymbol{r})\ \rho(t,\boldsymbol{r})\ dSdt\ , \tag{1.6}$$

where $\boldsymbol{n}(\boldsymbol{r})$ denotes the outward pointing normal on the boundary at $\boldsymbol{r} \in \partial B$. It is assumed that mass is neither created nor destroyed. Therefore, with (1.5) and (1.6) the describing equation is

$$\begin{aligned} \int_{t_0'}^{t_1'} \int_B \frac{\partial}{\partial t}\rho(t,\boldsymbol{r})\ d\boldsymbol{r}dt &= \int_{t_0'}^{t_1'} \int_{\partial B} -\boldsymbol{n}(\boldsymbol{r}) \cdot \boldsymbol{u}(t,\boldsymbol{r})\ \rho(t,\boldsymbol{r})\ dSdt \\ &= -\int_{t_0'}^{t_1'} \int_B \boldsymbol{\nabla}_{\boldsymbol{r}} \cdot (\boldsymbol{u}(t,\boldsymbol{r})\ \rho(t,\boldsymbol{r}))\ d\boldsymbol{r}dt\ , \end{aligned} \tag{1.7}$$

whereby for the transformation Gauss' theorem is applied. Since (1.7) is derived for arbitrary $[t_0', t_1'] \subseteq I$ and $B \subseteq \Omega$ and providing $\boldsymbol{u}$, ρ and their derivatives are assumed to be continuous, the integrals can be omitted and the equation unfolds as

$$\frac{\partial}{\partial t}\rho + \boldsymbol{\nabla}_{\boldsymbol{r}} \cdot (\boldsymbol{u}\ \rho) = 0 \qquad\qquad \text{in } I \times \Omega\ . \tag{1.8}$$

In case of considering an incompressible fluid ρ is constant so that (1.8) simplifies to

$$\boldsymbol{\nabla}_{\boldsymbol{r}} \cdot \boldsymbol{u} = 0 \qquad\qquad \text{in } I \times \Omega\ . \tag{1.9}$$

1.1.3. Conservation of Linear Momentum. The law of conservation of linear momentum states that the total linear momentum of a fluid in a closed system will remain constant if no interactions with any forces exist. In the following, a general derivation of the equation that describes the conservation law is given. Therefore, the mass density ρ is understood as a function of time and space, i.e. the restriction to incompressible fluids is dropped for the moment. For $B \subseteq \Omega$ and $t \in I$ the momentum $\boldsymbol{M}(B,t)$ is given by

$$\boldsymbol{M}(B,t) := \int_B \rho(t,\boldsymbol{r})\boldsymbol{u}(t,\boldsymbol{r})\ d\boldsymbol{r}\ . \tag{1.10}$$

In the here presented derivation external forces are considered. According to Newton's second law, the rate of change of momentum is equal to the forces applied. One can distinguish two kinds of forces, namely volume and surface forces. Furthermore, the considered domain B cannot be assumed to represent a closed system since the flux of momentum through the boundary ∂B needs to be considered, too.

The total change of momentum in B during $[t'_0, t'_1] \subseteq I$ is given as

$$\begin{aligned}\boldsymbol{M}(B,t'_1) - \boldsymbol{M}(B,t'_0) &= \int_B \rho(t'_1,\boldsymbol{r})\boldsymbol{u}(t'_1,\boldsymbol{r})\ d\boldsymbol{r} - \int_B \rho(t'_0,\boldsymbol{r})\boldsymbol{u}(t'_0,\boldsymbol{r})\ d\boldsymbol{r} \\ &= \int_{t'_0}^{t'_1} \int_B \frac{\partial}{\partial t}\left(\rho(t,\boldsymbol{r})\boldsymbol{u}(t,\boldsymbol{r})\right)\ d\boldsymbol{r}dt\ .\end{aligned} \tag{1.11}$$

A volume force like gravity, inertial, electrostatic or electromagnetic force is expressed by a function called the *density of volume force*

$$\boldsymbol{F} : \begin{cases} I \times \Omega & \to \mathbb{R}^d \\ (t,\boldsymbol{r}) & \mapsto \boldsymbol{F}(t,\boldsymbol{r})\ . \end{cases} \tag{1.12}$$

Then, the corresponding *volume force* acting in B at time t is

$$\int_B \rho(t,\boldsymbol{r})\boldsymbol{F}(t,\boldsymbol{r})\ d\boldsymbol{r}\ . \tag{1.13}$$

Due to Newton's second law the change of momentum in B during $[t'_0, t'_1]$ as result of applied volume forces obeys

$$\int_{t'_0}^{t'_1} \int_B \rho(t,\boldsymbol{r})\boldsymbol{F}(t,\boldsymbol{r})\ d\boldsymbol{r}dt\ . \tag{1.14}$$

A surface force is given in the form of the *stress tensor*

$$\boldsymbol{P} : \begin{cases} I \times \Omega & \to \mathbb{R}^{d\times d} \\ (t,\boldsymbol{r}) & \mapsto \boldsymbol{P}(t,\boldsymbol{r})\ . \end{cases} \tag{1.15}$$

Let $\boldsymbol{n}(\boldsymbol{r}) \in \mathbb{R}^d$ denote the unit outer normal at $\boldsymbol{r} \in \partial\Omega$. Then, a surface force acting on ∂B at time t is

$$\int_{\partial B} \boldsymbol{P}(t,\boldsymbol{r}) \cdot \boldsymbol{n}(\boldsymbol{r})\ dS\ . \tag{1.16}$$

With (1.16) and again with Newton's second law one gets the change of momentum in B during $[t'_0, t'_1]$ caused by surface forces as

$$\int_{t'_0}^{t'_1} \int_{\partial B} \boldsymbol{P}(t,\boldsymbol{r}) \cdot \boldsymbol{n}(\boldsymbol{r})\ dSdt = \int_{t'_0}^{t'_1} \int_B \boldsymbol{\nabla}_{\boldsymbol{r}} \cdot \boldsymbol{P}(t,\boldsymbol{r})\ d\boldsymbol{r}dt\ , \tag{1.17}$$

which is transformed applying Gauss' theorem.

Gauss' theorem is employed again for the next transformation. There, it is applied for each space dimension i, $(i = 1, 2, ..., d)$. The left hand side displays the momentum which enters and leaves B through the boundary ∂B during $[t'_0, t'_1]$:

$$\int_{t'_0}^{t'_1} \int_{\partial B} - \boldsymbol{n}(\boldsymbol{r}) \cdot \boldsymbol{u}(t, \boldsymbol{r}) \rho(t, \boldsymbol{s}) u_i(t, \boldsymbol{r}) \; dSdt$$
$$= - \int_{t'_0}^{t'_1} \int_B \boldsymbol{\nabla_r} \cdot (\boldsymbol{u}(t, \boldsymbol{r}) \rho(t, \boldsymbol{r}) u_i(t, \boldsymbol{r})) \; d\boldsymbol{r}dt \; . \tag{1.18}$$

With (1.11) the total gained minus lost momentum in B during $[t'_0, t'_1]$ is given. This change is caused by volume forces (1.14), surface forces (1.17) and flux through the boundary of B (1.18). All derivations are valid for arbitrary $[t'_0, t'_1] \subseteq I$ and $B \subseteq \Omega$ and $\boldsymbol{u}$, ρ, $\boldsymbol{F}$, $\boldsymbol{P}$ and their derivatives are assumed to be continuous. Therefore, the integrals can be omitted and the *conservative form of the momentum equation* unfolds as

$$\frac{\partial}{\partial t} (\rho \; \boldsymbol{u}) = \rho \; \boldsymbol{F} + \boldsymbol{\nabla_r} \cdot \boldsymbol{P} - \boldsymbol{\nabla_r} \cdot (\rho \; \boldsymbol{u} \otimes \boldsymbol{u}) \qquad \text{in } I \times \Omega \; . \tag{1.19}$$

Applying the chain rule, the equation can be further transformed. The governing equation of the conservation of mass law (1.8) serves as a substitute. Further, terms are added such that the negative terms on both sides vanish. Thus, the following equivalences for (1.19) hold in $I \times \Omega$

$$\begin{aligned}
\frac{\partial}{\partial t} (\rho \; \boldsymbol{u}) &= \rho \; \boldsymbol{F} + \boldsymbol{\nabla_r} \cdot \boldsymbol{P} - \boldsymbol{u} \; \boldsymbol{\nabla_r} \cdot (\rho \; \boldsymbol{u}) - (\rho \; \boldsymbol{u} \cdot \boldsymbol{\nabla_r}) \, \boldsymbol{u} \\
\frac{\partial}{\partial t} \rho \; \boldsymbol{u} + \rho \; \frac{\partial}{\partial t} \boldsymbol{u} &= \rho \; \boldsymbol{F} + \boldsymbol{\nabla_r} \cdot \boldsymbol{P} - \boldsymbol{u} \; \boldsymbol{\nabla_r} \cdot (\rho \; \boldsymbol{u}) - (\rho \; \boldsymbol{u} \cdot \boldsymbol{\nabla_r}) \, \boldsymbol{u} \\
- \boldsymbol{\nabla_r} \cdot (\rho \; \boldsymbol{u}) \;\; \boldsymbol{u} + \rho \; \frac{\partial}{\partial t} \boldsymbol{u} &= \rho \; \boldsymbol{F} + \boldsymbol{\nabla_r} \cdot \boldsymbol{P} - \boldsymbol{u} \; \boldsymbol{\nabla_r} \cdot (\rho \; \boldsymbol{u}) - (\rho \; \boldsymbol{u} \cdot \boldsymbol{\nabla_r}) \, \boldsymbol{u} \\
\rho \; \frac{\partial}{\partial t} \boldsymbol{u} + (\rho \; \boldsymbol{u} \cdot \boldsymbol{\nabla_r}) \, \boldsymbol{u} &= \rho \; \boldsymbol{F} + \boldsymbol{\nabla_r} \cdot \boldsymbol{P} \; .
\end{aligned} \tag{1.20}$$

The finally obtained equation is said to be the *non-conservative form of the momentum equation.*

1.1.4. Conservation of Angular Momentum. The law of conservation of angular momentum states that the total angular momentum of a fluid in a closed system will be constant in the case of absent forces. In the derivation to come, the mass density ρ is understood as function of time and space, i.e. the restriction to incompressible fluids is dropped momentarily. Further, in the following exclusively the case $d = 3$ is considered. For $B \subseteq \Omega$ and $t \in I$ the angular momentum $\boldsymbol{L}(B, t)$ is defined by

$$\boldsymbol{L}(B, t) := \int_B \boldsymbol{r} \times (\rho(t, \boldsymbol{r}) \boldsymbol{u}(t, \boldsymbol{r})) \; d\boldsymbol{r} \; . \tag{1.21}$$

The cross product for vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$ is given by

$$(\boldsymbol{a} \times \boldsymbol{b})_i := \sum_{j,k=1}^{3} \epsilon_{ijk} a_j b_k \qquad (i = 1, 2, 3) \; , \tag{1.22}$$

where ϵ_{ijk} is the Levi-Civita symbol, which is defined as

$$\epsilon_{ijk} := \begin{cases} 1 & : \text{ if } (i,j,k) \text{ is } (1,2,3),\ (3,1,2) \text{ or } (2,3,1) \\ -1 & : \text{ if } (i,j,k) \text{ is } (3,2,1),\ (1,3,2) \text{ or } (2,1,3) \\ 0 & : \text{ otherwise .} \end{cases}$$

Similarly to the derivations presented in the previous subsection, the flux of momentum through the boundary, volume and surface forces need to be considered to derive another condition as a result of the law of conservation of angular momentum. The total change of angular momentum in B during $[t'_0, t'_1] \subseteq I$ is given as

$$\begin{aligned} \boldsymbol{L}(B, t'_1) - \boldsymbol{L}(L, t'_0) &= \int_B \boldsymbol{r} \times (\rho(t'_1, \boldsymbol{r})\boldsymbol{u}(t'_1, \boldsymbol{r}))\ d\boldsymbol{r} - \int_B \boldsymbol{r} \times (\rho(t'_0, \boldsymbol{r})\boldsymbol{u}(t'_0, \boldsymbol{r}))\ d\boldsymbol{r} \\ &= \int_{t'_0}^{t'_1} \int_B \boldsymbol{r} \times \frac{\partial}{\partial t} (\rho(t, \boldsymbol{r})\boldsymbol{u}(t, \boldsymbol{r}))\ d\boldsymbol{r}dt\ . \end{aligned} \tag{1.23}$$

According to Newton's second law one gets the change of angular momentum in B during $[t'_0, t'_1]$ caused by a volume force $\boldsymbol{F}$ as

$$\int_{t'_0}^{t'_1} \int_B \boldsymbol{r} \times (\rho(t, \boldsymbol{r})\boldsymbol{F}(t, \boldsymbol{r}))\ d\boldsymbol{r}dt\ . \tag{1.24}$$

In a similar way, the change of the angular momentum as a result of a surface force $\boldsymbol{P}$ is obtained. For each space dimension $i = 1, 2, 3$ the term is rewritten as a sum and Gauss' theorem is applied:

$$\begin{aligned} &\int_{t'_0}^{t'_1} \int_{\partial B} (\boldsymbol{r} \times (\boldsymbol{n}(\boldsymbol{r}) \cdot \boldsymbol{P}(t, \boldsymbol{r})))_i\ dSdt \\ &\qquad = \int_{t'_0}^{t'_1} \int_{\partial B} \sum_{j,k,l=1}^{3} \epsilon_{ijk} r_j n_l(\boldsymbol{r}) P_{lk}(t, \boldsymbol{r})\ dSdt \\ &\qquad = \int_{t'_0}^{t'_1} \int_B \sum_{j,k,l=1}^{3} \epsilon_{ijk} \frac{\partial}{\partial r_l} (r_j P_{lk}(t, \boldsymbol{r}))\ d\boldsymbol{r}dt \\ &\qquad = \int_{t'_0}^{t'_1} \int_B \sum_{j,k,l=1}^{3} \epsilon_{ijk} \left(\frac{\partial r_j}{\partial r_l} P_{lk}(t, \boldsymbol{r}) + r_j \frac{\partial}{\partial r_l} P_{lk}(t, \boldsymbol{r}) \right)\ d\boldsymbol{r}dt \\ &\qquad = \int_{t'_0}^{t'_1} \int_B \sum_{j,k=1}^{3} \epsilon_{ijk} P_{jk}(t, \boldsymbol{r}) + (\boldsymbol{r} \times (\boldsymbol{\nabla}_{\boldsymbol{r}} \cdot \boldsymbol{P}(t, \boldsymbol{r})))_i\ d\boldsymbol{r}dt\ . \end{aligned} \tag{1.25}$$

The last source of change of the angular momentum to consider is the flux through the boundary ∂B during $[t'_0, t'_1]$. With the definition of the cross product, Gauss' theorem and the chain rule, one obtains the following equalities for each of the three space dimension $i = 1, 2, 3$:

$$\begin{aligned}
&\int_{t_0'}^{t_1'} \int_{\partial B} - \boldsymbol{n}(\boldsymbol{r}) \cdot \boldsymbol{u}(t,\boldsymbol{r}) \left(\boldsymbol{r} \times \rho(t,\boldsymbol{r})\boldsymbol{u}(t,\boldsymbol{r})\right)_i \, dSdt \\
&\qquad = -\int_{t_0'}^{t_1'} \int_B \boldsymbol{\nabla_r} \cdot \left(\boldsymbol{u}(t,\boldsymbol{r}) \sum_{j,k=1}^{3} \epsilon_{ijk} r_j u_k(t,\boldsymbol{r})\rho(t,\boldsymbol{r}) \right) \, d\boldsymbol{r}dt \\
&\qquad = -\int_{t_0'}^{t_1'} \int_B \sum_{j,k,l=1}^{3} \epsilon_{ijk} \frac{\partial}{\partial r_l} \left(u_l(t,\boldsymbol{r}) r_j u_k(t,\boldsymbol{r})\rho(t,\boldsymbol{r})\right) \, d\boldsymbol{r}dt \\
&\qquad = -\int_{t_0'}^{t_1'} \int_B \sum_{j,k,l=1}^{3} \epsilon_{ijk} \left(\frac{\partial}{\partial r_l} r_j \left(u_l(t,\boldsymbol{r}) u_k(t,\boldsymbol{r})\rho(t,\boldsymbol{r})\right) \right. \\
&\qquad\quad \left. + \frac{\partial}{\partial r_l} \left(u_l(t,\boldsymbol{r}) u_k(t,\boldsymbol{r})\rho(t,\boldsymbol{r})\right) r_j \right) \, d\boldsymbol{r}dt \\
&\qquad = -\int_{t_0'}^{t_1'} \int_B \left(\boldsymbol{r} \times \left(\boldsymbol{\nabla_r} \cdot \left(\boldsymbol{u}(t,\boldsymbol{r}) u_i(t,\boldsymbol{r})\rho(t,\boldsymbol{r})\right)\right)\right)_i \, d\boldsymbol{r}dt \; .
\end{aligned} \tag{1.26}$$

The total gained minus lost angular momentum in B during $[t_0', t_1']$ is given in (1.23). This change is caused by volume forces (1.24), surface forces (1.25) and flux through the boundaries of B (1.26). All derivations are valid for arbitrary $[t_0', t_1'] \subseteq I$ and $B \subseteq \Omega$ and $\boldsymbol{r}$, $\boldsymbol{u}$, ρ, $\boldsymbol{F}$, $\boldsymbol{P}$ and their derivatives are assumed to be continuous, so that the integrals can be omitted. The resulting balance equation is

$$\begin{aligned}
\left(\boldsymbol{r} \times \frac{\partial}{\partial t} (\rho \, \boldsymbol{v}) \right)_i &= (\boldsymbol{r} \times \rho \, \boldsymbol{F})_i + \sum_{j,k=1}^{3} \epsilon_{ijk} P_{jk} + (\boldsymbol{r} \times (\boldsymbol{\nabla_r} \cdot \boldsymbol{P}))_i \\
&\quad - (\boldsymbol{r} \times (\boldsymbol{\nabla_r} \cdot (\boldsymbol{u} u_i \rho)))_i \;\; (i = 1,2,3) \; .
\end{aligned} \tag{1.27}$$

Subtracting the cross product of $\boldsymbol{r}$ and the conservative form of the momentum equation (1.19) from the latter equation yields

$$0 = \sum_{j,k=1}^{3} \epsilon_{ijk} P_{jk} \qquad\qquad (i = 1,2,3) \tag{1.28}$$

which is, according to the definition of the cross product, equivalent to

$$P_{23} = P_{32}, \;\; P_{31} = P_{13}, \;\; P_{12} = P_{21} \; . \tag{1.29}$$

That is in turn the symmetry of the stress tensor $\boldsymbol{P}$.

1.1.5. A Viscosity Model for Newtonian Fluids. In the three-dimensional case ($d = 3$), the three conservation laws (mass, linear and angular momentum) set up seven conditions for twelve unknown scalars, that are three for the velocity $\boldsymbol{u}$ and nine for the stress tensor $\boldsymbol{P}$. For $d = 2$ four conditions are obtained for seven unknowns. The gap of the underdetermined system of equations is closed within the framework of rheology for fluids. In this field the functional dependence of the stress tensor $\boldsymbol{P}$ on the deformation velocity tensor $\boldsymbol{D}$ is studied. At this, the material properties of the considered fluid are described phenomenologically, i.e. macroscopic quantities are measured and related. The ansatz for Newtonian fluids finds a motivation in a simple experiment of a flow between two parallel planes where one of them is fixed and the other one moves slowly parallel to the fixed plane in direction of the r_i-axis of the Cartesian coordinate system ($i \in \{1, 2, 3\}$).

A steady flow develops with a linear increasing velocity on a fictitious line connecting the two planes in an orthogonal manner. Without loss of generality, the line can be assumed as parallel to the r_j-axis of the coordinate system with $j \in \{1,2,3\}$ and $j \neq i$. Then, the linearity can be expressed by

$$\boldsymbol{P}_{ji} = \mu \frac{\partial u_i}{\partial r_j} \ , \tag{1.30}$$

which implicitly defines the *dynamic viscosity* μ. That justifies relating the stress tensor $\boldsymbol{P}$ and the Jacobian matrix of $\boldsymbol{u}$ in a similar way. From the conservation law for angular momentum it is known that the stress tensor $\boldsymbol{P}$ is symmetric. Therefore, the skew-symmetric part of $\boldsymbol{\nabla_r} \otimes \boldsymbol{u}$ can be neglected. The symmetric part, which is the strain rate tensor $\boldsymbol{D}$, remains to be related to $\boldsymbol{P}$. The restriction to Newtonian fluids finds its characterising formulation in Stokes' postulates which are stated in Subsection 1.1.1. Under this conditions it can be proved [**36**] that the stress tensor has the form

$$\boldsymbol{P} = (-p + \lambda \boldsymbol{\nabla_r} \cdot \boldsymbol{u})\, \boldsymbol{I}_d + 2\mu \boldsymbol{D} \ , \tag{1.31}$$

where $\lambda \in \mathbb{R}$, $\lambda \geq 0$ is called the *volume viscosity* and $\mu \in \mathbb{R}$, $\mu \geq 0$ is the dynamic viscosity. For the restriction of considering only incompressible fluids (1.31) simplifies to

$$\boldsymbol{P} = -p\boldsymbol{I}_d + 2\mu \boldsymbol{D} \ . \tag{1.32}$$

The stress tensor $\boldsymbol{P}$ can now be substituted using relation (1.32). Thereby, it is to note that the dynamic viscosity μ is given in advance as a material property. Thus, considering the case $d = 3$ four unknown scalars remain, namely three for the velocity $\boldsymbol{u}$ and one for the pressure p. The conservation laws for mass and linear momentum provide four independent conditions so that the system of equations is no longer underdetermined. The same holds in the case where $d = 2$. Here, three conditions are obtained for three unknown scalars. In both cases, the system is consolidated to the governing equation for incompressible Newtonian fluids, which is called the incompressible Navier-Stokes equation.

1.1.6. The Incompressible Navier-Stokes Equation. As a result of the viscosity model for Newtonian fluids, the stress tensor $\boldsymbol{P}$ can now be related to the pressure $\boldsymbol{p}$, the dynamic viscosity μ and the deformation velocity tensor $\boldsymbol{D}$ as formulated in (1.32). To obtain the Navier-Stokes equation $\boldsymbol{P}$ needs to be substituted in the non-conservative form of the momentum equation (1.20). Thereunto, the following transformations are required, whereas for the last transformation the mass conservation law formulated in (1.9) is employed:

$$\begin{aligned} \boldsymbol{\nabla_r} \cdot \boldsymbol{P} &= \boldsymbol{\nabla_r} \cdot (-p\boldsymbol{I}_d + 2\mu\boldsymbol{D}) \\ &= -\boldsymbol{\nabla_r} p + \mu(\boldsymbol{\Delta_r u} + \boldsymbol{\nabla_r}(\underbrace{\boldsymbol{\nabla_r} \cdot \boldsymbol{u}}_{=0})) \\ &= -\boldsymbol{\nabla_r} p + \mu \boldsymbol{\Delta_r u} \ . \end{aligned} \tag{1.33}$$

Then, the latter term is substituted into (1.20). This leads together with the conservation law for mass (1.9) to the *incompressible Navier-Stokes equation*:

$$\begin{aligned} \rho \frac{\partial}{\partial t}\boldsymbol{u} + (\rho\, \boldsymbol{u} \cdot \boldsymbol{\nabla_r})\, \boldsymbol{u} &= \rho\, \boldsymbol{F} - \boldsymbol{\nabla_r} p + \mu \boldsymbol{\Delta_r u} && \text{in } I \times \Omega \\ \boldsymbol{\nabla_r} \cdot \boldsymbol{u}(t, \boldsymbol{r}) &= 0 && \text{in } I \times \Omega \ . \end{aligned} \tag{1.34}$$

1.2. Mesoscopic Approach: A Model for the Dynamics of Rarefied Gases

Models of statistical mechanics are of mesoscopic character. A state of a physical system is described by probabilities for possible states of an underlying microscopic system. In particular, such models provide a framework to explain macroscopic thermodynamic quantities like velocity, heat, free energy or entropy as a result of averages of states of atoms or molecules. In 1872 Boltzmann established the basic governing equation for the kinetic theory of dilute gases, namely the famous Boltzmann equation. Later, the equation, its generalisations and simplifications proved to be valuable to describe other physical phenomena, for example radioactive transfer, electronic transport in solids or plasmas [**25**] and viscous flows. The problems of rigorous derivation of the equation starting from a microscopic model with assumptions on this level and finally the existence and uniqueness of a solution has attracted many physicists as well as mathematicians (see e.g. [**8, 25**] and references therein). The objective of this section is to introduce the basic terminology, assumptions and concepts building the theory of the dynamics of rarefied gases. Further, it is intended to establish the nomenclatures and notations which are of great importance for all other sections and chapters to come. The presented approach is mainly inspired by the work of Cercignani [**25**] and Hänel [**59**].

1.2.1. Terminology and Model Assumptions. The mathematical description of the mesoscopic model for the dynamics of rarefied gases is based on a microscopic picture of the fluid. A volume of a rarefied gas is represented by a domain $\Omega \subseteq \mathbb{R}^d$ wherein a huge number $N \in \mathbb{N}$ of ball-shaped and indivisible particles interact. The gas particles can be atoms or molecules. Simplifying, it is assumed that all particles have the same *particle diameter* $\sigma \in \mathbb{R}_{>0}$ and the same *particle mass* (*or molecule mass*) $m \in \mathbb{R}_{>0}$.

It is observed that the particles interact permanently and accidentally. This phenomenon is known as *Brownian motion*. It is modelled by assuming free flows of particles, respecting Newton's laws without considering any forces except external given volume forces like gravity. The path between two collisions is called *free path* and the average of free paths of a volume over a given time is called the *mean free path* $l_f \in \mathbb{R}_{>0}$. Then, if an atom or a molecule collides with another one, it abruptly changes its speed and direction. Simplifying, collisions are assumed to be elastic, i.e. the total momentum and kinetic energy is conserved. Another assumption is that the particles interact weakly, i.e. they collide relatively seldom. This is justified heuristically, since a volume of a rarefied gas is seen to consist of a large number of particles with a relatively small diameter σ. Idealising, this is formulated mathematically by postulating the existence of the limes $N\sigma^2$ for $N \to \infty$ and $\sigma \to 0$. And furthermore it is determined that only two particles can collide at the same time (binary collision assumption). In Figure 1.1 a possible motion of a particle as a consequence of free flows and collisions is illustrated.

In this microscopic model the state of one gas particle is fully characterised by its current centre position $\boldsymbol{r} \in \Omega$ and its current centre velocity $\boldsymbol{v} \in \Xi = \mathbb{R}^d$. The following terms for the spaces are common:

- *position space* $\Omega \subseteq \mathbb{R}^d$,
- *velocity space* $\Xi = \mathbb{R}^d$,
- *phase space* $\mu = \Omega \times \Xi \subseteq \mathbb{R}^{2d}$.

Both $\boldsymbol{r}$ and $\boldsymbol{v}$ are functions of a considered time interval $I = [t_0, t_1] \subseteq \mathbb{R}$ into Ω and Ξ, respectively. The state of the whole system at a time $t \in I$ is given by

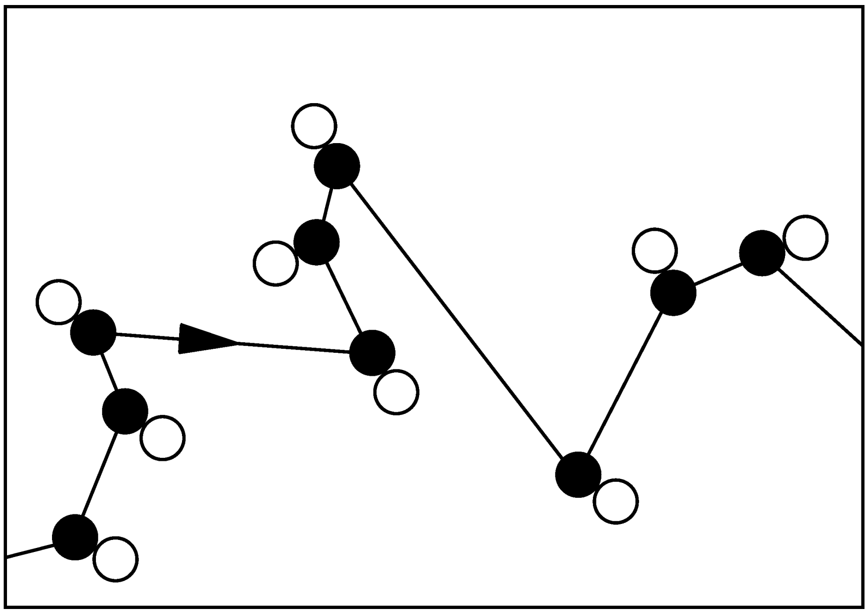


FIGURE 1.1. Brownian motion: Trajectory of a molecule (black ball) as a result of free flight and collision with other molecules (white balls).

the state of all N particles. It is represented by the phase space $\mu^N = \Omega \times \Xi \subseteq \mathbb{R}^{2dN}$.

The so far introduced model is of microscopic character. The dynamics of a rarefied gas can be reconstructed and also predicted if the states, i.e. the positions in the phase space, of all its particles at a time $t \in I$ are known. If this is the case the process will be reversible in time. However, due to Heisenberg's uncertainty principle it is not possible to know both the position and the velocity of a particle with arbitrary precision. And even if one would knew an exact start distribution it would be impossible to simulate the dynamics of a quantum gas of a macroscopic scale with a computer of today. For instance, the simulation of one cm^3 of air at 20 °C and 1000 hPa requires considering the interaction of about $2,69 \cdot 10^{19}$ molecules. To save only one state of such a quantity of particles would cost more than 10^8 tera bytes. Yet, it is possible to obtain macroscopic quantities without knowing the exact states of the molecules - averages of possible states are sufficient. And that is the basic idea of mesoscopic models. The reversible character of the microscopic process of particle interaction is waived. It is abstained knowing the exact states of all particles, instead probabilities for the states are introduced and build the fundamental object of investigation.

For every particle $i = 1, ..., N$ the position $(\boldsymbol{r}(t), \boldsymbol{v}(t))_i$ in its corresponding phase space μ_i with $\mu_i \neq \emptyset$ is in the mesoscopic model considered as a realisation of a random variable. The phase space μ_i generates a σ-algebra $\sigma(\mu_i)$. $P_i^t(A_i)$ is defined as the probability to find $(\boldsymbol{r}(t), \boldsymbol{v}(t))_i$ in $A_i \in \sigma(\mu_i)$. All together, they define a probability space $(\mu_i, \sigma(\mu_i), P_i^t)$ where P_i^t is a probability measure.

Based on the family of probability spaces $(\mu_i, \sigma(\mu_i), P_i^t)$, for a given $A \in \sigma(\mu)$ a finite discrete probability space (Θ, P_A^t) is introduced. The variable $k \in \Theta = \{1, 2, ..., N\}$ represents the number of particles found in A with probability $P_A^t(k)$.

The probability measure is given by

$$P_A^t : \begin{cases} \Theta & \to [0,1] \\ k & \mapsto P_A^t(k) = \displaystyle\sum_{B \subseteq \Theta, |B|=k} \left(\prod_{i \in B} P_i^t(A) \prod_{j \in \Theta / B} \left(1 - P_j^t(A)\right) \right) \end{cases} ,$$

With it, the expected value of the number of particles found in A is

$$E_A^t = \sum_{k=1}^{N} k \; P_A^t(k) \; .$$

$E_A^t / N =: P_{\mathcal{N}}^t(A)$ as function of $A \subseteq \mu$ defines a probability measure itself. The underlying probability space is the phase space μ and a sigma-algebra is $\sigma(\mu)$. Both the expected value E_A^t and the probability measure $P_{\mathcal{N}}^t(A)$ are the fundamental objects of the derivation to come.

It is to be noted that generally the realisations $(\boldsymbol{r}(t), \boldsymbol{v}(t))_i$ are not identically-distributed and independent of each other. The reason is that P_i^t depends on all $P_j^{t'}$ where $j = 1, ..., N$ and $t' \in I$ with $t' < t$. This is due to the correlation property of the collision process. If only one particle at time t' has a different state distribution, for example due to an applied external force, other particle states P_i^t will depend on it. However, the random variables $(\boldsymbol{r}(t), \boldsymbol{v}(t))_i$ are assumed to be identically-distributed and independent of each other. This is justified by the assumption of weak interaction of rarefied gas molecules which is also known as *molecular chaos assumption* (see e.g. [**8**]). In the case of identically-distributed $(\boldsymbol{r(t)}, \boldsymbol{v(t)})_i$, P_A^t is the binomial distribution $B(N, P_1^t(A))$. Its expectation is $E_A^t = N \; P_1^t(A)$ and therefore one gets that $P_{\mathcal{N}}^t(A) = P_1^t(A)$ for all $t \in I$ and $A \in \sigma(\mu)$. That means in turn that the distribution of the expectation of N particles is the same as the distribution of one particle.

1.2.2. Particle Density Function and Macroscopic Moments. With the probability measure $P_{\mathcal{N}}^t$ as it is introduced in the previous subsection it is possible to derive macroscopic quantities. Thereto, the density function $p_{\mathcal{N}}^t$ of $P_{\mathcal{N}}^t$ is multiplied with the number of particles N in Ω. The resulting function is called the *(expected) particle density function*[2] and is given by

$$f : \begin{cases} I \times \Omega \times \mathbb{R}^d & \to \mathbb{R}_{\geq 0} \\ (t, \boldsymbol{r}, \boldsymbol{v}) & \mapsto f(t, \boldsymbol{r}, \boldsymbol{v}) \; . \end{cases} \tag{1.35}$$

Due to the definition of $P_{\mathcal{N}}^t$ and that $f(t, \cdot, \cdot)/N$ is the density function of $P_{\mathcal{N}}^t$ one gets for any $B \times C \in \sigma(\mu)$ and $t \in I$

$$E_{B \times C}^t = N \; P_{\mathcal{N}}^t(A) = N \int_B \int_C p_{\mathcal{N}}^t(\boldsymbol{r}, \boldsymbol{v}) \; d\boldsymbol{r} d\boldsymbol{v} = \int_B \int_C f(t, \boldsymbol{r}, \boldsymbol{v}) \; dr dv \; . \tag{1.36}$$

That is the expected number of particles found in $B \times C$ at time t. Thus, f describes the distribution of the expected number of particles over the phase space μ. Based on that interpretation, various macroscopic quantities are obtained as moments of f by integrating over the velocity space $\mathbb{R}^d$. An extract is stated in the following:

[2]By definition, f is not a density in a statistical sense. However, in the literature it is often called density or distribution function (see e.g. [**8, 59**].). In [**25**] f is called a (expected) mass density which is given by $f := m \; N \; p_{\mathcal{N}}^t$

- *particle density*

$$n : \begin{cases} I \times \Omega & \to \mathbb{R}_{>0} \\ (t, \boldsymbol{r}) & \mapsto n(t, \boldsymbol{r}) := \displaystyle\int_{\mathbb{R}^d} f(t, \boldsymbol{r}, \boldsymbol{v}) \, d\boldsymbol{v} \ , \end{cases} \tag{1.37}$$

- *mass density*

$$\rho : \begin{cases} I \times \Omega & \to \mathbb{R}_{>0} \\ (t, \boldsymbol{r}) & \mapsto \rho(t, \boldsymbol{r}) := m \ n(t, \boldsymbol{r}) \ , \end{cases} \tag{1.38}$$

- *velocity*

$$\boldsymbol{u} : \begin{cases} I \times \Omega & \to \mathbb{R}^d \\ (t, \boldsymbol{r}) & \mapsto \boldsymbol{u}(t, \boldsymbol{r}) := \dfrac{1}{n(t, \boldsymbol{r})} \displaystyle\int_{\mathbb{R}^d} \boldsymbol{v} \ f(t, \boldsymbol{r}, \boldsymbol{v}) \, d\boldsymbol{v} \ , \end{cases} \tag{1.39}$$

- *stress tensor*

$$\boldsymbol{P} : \begin{cases} I \times \Omega & \to \mathbb{R}^{d \times d} \\ (t, \boldsymbol{r}) & \mapsto \boldsymbol{P}(t, \boldsymbol{r}) := m \displaystyle\int_{\mathbb{R}^d} \big(\boldsymbol{v} - \boldsymbol{u}(t, \boldsymbol{r})\big) \otimes \big(\boldsymbol{v} - \boldsymbol{u}(t, \boldsymbol{r})\big) f(t, \boldsymbol{r}, \boldsymbol{v}) \, d\boldsymbol{v} \ , \end{cases} \tag{1.40}$$

- *pressure*

$$\boldsymbol{p} : \begin{cases} I \times \Omega & \to \mathbb{R}_{>0} \\ (t, \boldsymbol{r}) & \mapsto \boldsymbol{p}(t, \boldsymbol{r}) := \dfrac{1}{d} \left(\displaystyle\sum_{i=1}^{d} P_{ii}(t, \boldsymbol{r}) \right) \ . \end{cases} \tag{1.41}$$

In Theorem 1.2 it is remarked that the here proposed definitions of macroscopic quantities are consistent with the definitions established for the macroscopic model introduced in Section 1.1 in the sense that for an assumed solution f of the Boltzmann equation the corresponding macroscopic quantities fulfil the macroscopic conservation law of mass and momentum.

1.2.3. A Molecular Collision Model for Rigid Spheres. The subject of this subsection is to derive the expected number of particles that enter or leave, respectively, $B \times C \in \sigma(\mu)$ during $[t'_0, t'_1] \subseteq I$ due to the collision process. With this quantity, the Boltzmann equation can be derived in the next subsection. Furthermore, if one knows the particle density function f and with it the number of collisions in a volume during a time, the mean free path l_f and other characteristic mesoscopic quantities of a fluid can be calculated.

As mentioned before, particles are considered as hard spheres with equal diameter σ and mass m. To model the collision process, it is assumed that only two particles $i, j \in \{1, ..., N\}$, $i \neq j$ can collide at a time $t \in I$. Then, one finds that the function $\boldsymbol{v_i}$ is not continuous in t and therefore the following notation for the function is introduced:

- velocity before the collision $\boldsymbol{v_i}(t) := \lim_{\tilde{t} \nearrow t} \boldsymbol{v_i}(\tilde{t})$ and
- velocity after the collision $\boldsymbol{v'_i}(t) := \lim_{\tilde{t} \searrow t} \boldsymbol{v_i}(\tilde{t})$.

Analogously, for particle j its velocities before and after the collision $\boldsymbol{v_j}$ and $\boldsymbol{v'_j}$ are redefined. It is assumed that the encounter is elastic, i.e. the following two equations hold:

$$\boldsymbol{v_i} + \boldsymbol{v_j} = \boldsymbol{v'_i} + \boldsymbol{v'_j} \ , \tag{1.42}$$

$$\boldsymbol{v_i}^2 + \boldsymbol{v_j}^2 = \boldsymbol{v'_i}^2 + \boldsymbol{v'_j}^2 \ . \tag{1.43}$$

Relation (1.42) ensures the conservation of the momentum and (1.43) the conservation of the energy. To simplify the derivations to come the following variables are defined:

- relative velocity before collision $\boldsymbol{g} := \boldsymbol{v_j} = -\boldsymbol{v_i}$ and
- relative velocity after collision $\boldsymbol{g'} := \boldsymbol{v'_j} - \boldsymbol{v'_i}$.

The two particles i and j will collide at time t if

$$\begin{aligned} \boldsymbol{r_j}(t) \in S_i &:= \{\boldsymbol{r} \in \Omega : \|\boldsymbol{r_i} - \boldsymbol{r}\| = \sigma\} \ , \\ \boldsymbol{g}(t) \in \mathbb{R}^d_{n^+_{ij}} &:= \left\{\boldsymbol{g}(t) \in \mathbb{R}^d : \boldsymbol{n_{ij}} \cdot \boldsymbol{g}(t) \ > 0\right\} \end{aligned}$$

holds whereas $\boldsymbol{n_{ij}} := (\boldsymbol{r_i} - \boldsymbol{r_j})/\sigma$ denotes the unit normal vector pointing into the sphere S_i at $\boldsymbol{r_j} \in S_i$. Similarly, one gets the following conditions that two particles have collided at time t:

$$\begin{aligned} \boldsymbol{r_j}(t) \in S_i &:= \{\boldsymbol{r} \in \Omega : \|\boldsymbol{r_i} - \boldsymbol{r}\| = \sigma\} \ , \\ \boldsymbol{g'}(t) \in \mathbb{R}^d_{n^-_{ij}} &:= \left\{\boldsymbol{g'}(t) \in \mathbb{R}^d : \boldsymbol{n_{ij}} \cdot \boldsymbol{g'}(t) \ < 0\right\} \ . \end{aligned}$$

Further, with (1.42) and (1.43) the following equations hold [**25**]:

$$\begin{aligned} \boldsymbol{v'_i} &= \boldsymbol{v_i} - \boldsymbol{n_{ij}}\,(\boldsymbol{n_{ij}} \cdot \boldsymbol{g}) \ , \\ \boldsymbol{v'_j} &= \boldsymbol{v_j} + \boldsymbol{n_{ij}}\,(\boldsymbol{n_{ij}} \cdot \boldsymbol{g}) \ , \\ \boldsymbol{n_{ij}} \cdot \boldsymbol{g'} &= -\boldsymbol{n_{ij}} \cdot \boldsymbol{g} \ . \end{aligned} \tag{1.44}$$

Providing that particle i is in $A = B \times C$, one gets the probability for a collision of particle i with particle j during $[t'_0, t'_1]$ through

$$P^{[t'_0,t'_1]}_{ij}(A) := \int_{t'_0}^{t'_1} \int_B \int_C p^t_i(\boldsymbol{r_i}, \boldsymbol{v_i}) \int_{S_i} \int_{\mathbb{R}^d_{n^+_{ij}}} \boldsymbol{n_{ij}} \cdot \boldsymbol{g}\ p^t_j(\boldsymbol{s}, \boldsymbol{v_j})\ d\boldsymbol{v_j} d\boldsymbol{s} d\boldsymbol{v_i} d\boldsymbol{r_i} dt \ .$$

The expected number of particles which collide with another particle during $[t'_0, t'_1]$ and which are in A before the collision has taken place is then

$$M^+_{coll} := \sum_{i=1}^{N} \sum_{j=1, i\neq j}^{N} P^{[t'_0,t'_1]}_{ij}(A) \ .$$

The assumption of molecular chaos (cf. Subsection 1.2.1) leads to Boltzmann's famous *Stoßzahlansatz*. Due to the assumed identical one-particle distributions P^t_i and the definition of f, see (1.36), one finds

$$\begin{aligned} M^+_{coll} &= N(N-1) \int_{t'_0}^{t'_1} \int_B \int_C p^t_{\mathcal{N}}(\boldsymbol{r}, \boldsymbol{v}) \int_S \int_{\mathbb{R}^d_{n^+_*}} \boldsymbol{n_*} \cdot \boldsymbol{g}\ p^t_{\mathcal{N}}(\boldsymbol{s}, \boldsymbol{v_*})\ d\boldsymbol{v_*} d\boldsymbol{s} d\boldsymbol{v} d\boldsymbol{r} dt \\ &\approx N^2 \int_{t'_0}^{t'_1} \int_B \int_C p^t_{\mathcal{N}}(\boldsymbol{r}, \boldsymbol{v}) \int_S \int_{\mathbb{R}^d_{n^+_*}} \boldsymbol{n_*} \cdot \boldsymbol{g}\ p^t_{\mathcal{N}}(\boldsymbol{s}, \boldsymbol{v_*})\ d\boldsymbol{v_*} d\boldsymbol{s} d\boldsymbol{v} d\boldsymbol{r} dt \\ &= \int_{t'_0}^{t'_1} \int_B \int_C f(t, \boldsymbol{r}, \boldsymbol{v}) \int_S \int_{\mathbb{R}^d_{n^+_*}} \boldsymbol{n_*} \cdot \boldsymbol{g}\ f(t, \boldsymbol{s}, \boldsymbol{v_*})\ d\boldsymbol{v_*} d\boldsymbol{s} d\boldsymbol{v} d\boldsymbol{r} dt \end{aligned} \tag{1.45}$$

at which terms of $\mathcal{O}(N)$ are neglected because the leading order term is of $\mathcal{O}(N^2)$ and N is assumed to be large.

In a similar manner, the expected number of particles which collide with another particle during $[t_0', t_1']$ and which are in A after the collision has taken place can be derived:

$$M_{coll}^- \approx \int_{t_0'}^{t_1'} \int_B \int_C f(t, \boldsymbol{r}, \boldsymbol{v}') \int_S \int_{\mathbb{R}^d_{n_*^-}} -\boldsymbol{n}_* \cdot \boldsymbol{g}' \; f(t, \boldsymbol{s}, \boldsymbol{v}_*') \; d\boldsymbol{v}_*' d\boldsymbol{s} d\boldsymbol{v}' d\boldsymbol{r} dt \; .$$

Substituting $\boldsymbol{v}'$, $\boldsymbol{v}_*'$ and $\boldsymbol{g}'$ as stated in (1.44) one finds

$$M_{coll}^- \approx \int_{t_0'}^{t_1'} \int_B \int_C f'(t, \boldsymbol{r}, \boldsymbol{v}) \int_S \int_{\mathbb{R}^d_{n_*^+}} \boldsymbol{n}_* \cdot \boldsymbol{g} \; f'(t, \boldsymbol{s}, \boldsymbol{v}_*) \; d\boldsymbol{v}_* d\boldsymbol{s} d\boldsymbol{v} d\boldsymbol{r} dt \; , \tag{1.46}$$

whereas $f' := f(t, \boldsymbol{r}, \boldsymbol{v} - \boldsymbol{n}_* (\boldsymbol{n}_* \cdot \boldsymbol{g})$ and $f_*' := f(t, \boldsymbol{r}, \boldsymbol{v}_* - \boldsymbol{n}_* (\boldsymbol{n}_* \cdot \boldsymbol{g}) \boldsymbol{n}_*)$.

Finally, the expected difference of particles between those that enter and those that leave $A = B \times C \in \sigma(\mu)$ during $[t_0', t_1']$, which is represented by ∂M_{coll}, can be derived. A particle that is in A before a collision has taken place can either be still in or outside A after the encounter. In either case it is counted in M_{coll}^+. However, if it is still in A it is also counted in M_{coll}^-. The other particles expected counted in M_{coll}^- have not been in A and therefore represent the expected gain. Thus, with (1.45) and (1.46) M_{coll} is given approximately, neglecting terms of $\mathcal{O}(1/N)$, by

$$\begin{aligned} \partial M_{coll} &= M_{coll}^- - M_{coll}^+ \\ &\approx \int_{t_0'}^{t_1'} \int_B \int_C \int_S \int_{\mathbb{R}^d_{n_*^+}} \boldsymbol{n}_* \cdot \boldsymbol{g} \; (f' f_*' - f f_*) \; d\boldsymbol{v}_* d\boldsymbol{s} d\boldsymbol{v} d\boldsymbol{r} dt \; . \end{aligned} \tag{1.47}$$

1.2.4. The Boltzmann Equation. The expected number of particles at a time $t \in I$ in $B \times C \in \sigma(\mu)$ is given by $E_{B\times C}^t$. All $E_{B\times C}^t / N = P_N^t(B \times C)$ are probability measures as indicated in Subsection 1.2.1 and related to f by (1.36). Based on the model assumptions stated in Subsection 1.2.1 the Boltzmann equation sets up a condition for f. It can be seen as a conservation equation for the expected number of particles in a volume μ over a time I.

Let ∂M denote the expected difference of particles entering to those leaving $B \times C$ during $[t_0', t_1']$. According to the model assumptions, there are only three reasons why particles can enter or leave the volume. The first one is that they cross ∂B in their free flight during $[t_0', t_1']$ respecting Newton's laws. The number of such particles is captured by the variable $\partial M_{\partial B}$ in form of the difference of expected entering to leaving ones. Analogously, $\partial M_{\partial C}$ represents the expectation of particles crossing ∂C as the result of an external force F, which is the second reason. Finally, the third one is the collision of particles in and outside $B \times C$, whereas they change their velocity abruptly and hence enter or leave C. ∂M_{coll} represents the expected difference of gained to lost particles due to the collision process. Consequently, the following conservation equation is obtained

$$\partial M = \partial M_{\partial B} + \partial M_{\partial C} + \partial M_{coll} \; . \tag{1.48}$$

The expected total change of the number of particles during $[t_0', t_1']$ is

$$\begin{aligned}\partial M &= \int_B \int_C f(t_1', \boldsymbol{r}, \boldsymbol{v})\ d\boldsymbol{v} d\boldsymbol{r} - \int_B \int_C f(t_0', \boldsymbol{r}, \boldsymbol{v})\ d\boldsymbol{v} d\boldsymbol{r} \\ &= \int_{t_0'}^{t_1'} \frac{\partial}{\partial t} \int_B \int_C f(t, \boldsymbol{r}, \boldsymbol{v})\ d\boldsymbol{v} d\boldsymbol{r} dt \\ &= \int_{t_0'}^{t_1'} \int_B \int_C \frac{\partial}{\partial t} f(t, \boldsymbol{r}, \boldsymbol{v})\ d\boldsymbol{v} d\boldsymbol{r} dt\ .\end{aligned} \tag{1.49}$$

The expected difference of incoming to outgoing particles in μ crossing ∂B is given by the flux through the boundary. Applying Gauss' theorem one gets

$$\begin{aligned}\partial M_{\partial B} &= -\int_{t_0'}^{t_1'} \int_C \int_{\partial B} \boldsymbol{n}(\boldsymbol{s}) \cdot \boldsymbol{v}\ f(t, \boldsymbol{s}, \boldsymbol{v})\ d\boldsymbol{s} d\boldsymbol{v} dt \\ &= -\int_{t_0'}^{t_1'} \int_B \int_C \boldsymbol{v} \cdot \boldsymbol{\nabla_r} f(t, \boldsymbol{r}, \boldsymbol{v})\ d\boldsymbol{r} d\boldsymbol{v} dt\ ,\end{aligned} \tag{1.50}$$

where $\boldsymbol{n}$ denotes the unit vector normal to ∂B in the outward direction. Similarly, the balance of expected gained and lost particles due to an external force is calculated:

$$\begin{aligned}\partial M_{\partial C} &= -\int_{t_0'}^{t_1'} \int_B \int_{\partial C} \boldsymbol{n}(\boldsymbol{s}) \cdot \frac{\boldsymbol{F}}{m}\ f(t, \boldsymbol{r}, \boldsymbol{s})\ d\boldsymbol{s} d\boldsymbol{r} dt \\ &= -\int_{t_0'}^{t_1'} \int_B \int_C \frac{\boldsymbol{F}}{m} \cdot \boldsymbol{\nabla_v} f(t, \boldsymbol{r}, \boldsymbol{v})\ d\boldsymbol{r} d\boldsymbol{v} dt\ .\end{aligned} \tag{1.51}$$

Finally, the expected number of gained minus lost particles due to the process of collision ∂M_{coll} is stated. According to the derivation presented in the previous subsection, for hard spheres and for the limiting case $N \to \infty$ ∂M_{coll} is given by

$$\partial M_{coll} = \int_{t_0'}^{t_1'} \int_B \int_C \int_S \int_{\mathbb{R}^d_{n_*^+}} \boldsymbol{n_*} \cdot \boldsymbol{g}\ \ (f' f_*' - f f_*)\ d\boldsymbol{v_*} d\boldsymbol{s} d\boldsymbol{v} d\boldsymbol{r} dt\ . \tag{1.52}$$

With (1.49), (1.50), (1.51) and (1.52) all terms of the balance equation (1.48) are derived for arbitrary $[t_0', t_1'] \subseteq I$ and $B \times C \in \sigma(\mu)$. Then, assuming f and its derivatives are continuous, the integrals can be omitted and the Boltzmann equation for the rigid spheres collision model unfolds as

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{\boldsymbol{F}}{m} \cdot \boldsymbol{\nabla_v}\right) f = \underbrace{\int_S \int_{\mathbb{R}^d_{n_*^+}} \boldsymbol{n_*} \cdot \boldsymbol{g}\ \ (f' f_*' - f f_*)\ d\boldsymbol{v_*} d\boldsymbol{s}}_{=:J(f)}\ . \tag{1.53}$$

The right hand side of the Boltzmann equation defines the so-called *collision operator* J. With it (1.53) shortens to

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{\boldsymbol{F}}{m} \cdot \boldsymbol{\nabla_v}\right) f = J(f)\ . \tag{1.54}$$

1.3. Model Interpretation: From the Mesoscopic to the Macroscopic Model

The aim of this section is to study the relation of the mesoscopic model for rarefied gases to the macroscopic model for incompressible viscous flows. Thereunto, at first some fundamental properties of the Boltzmann equation are stated. Then, a simplification of the Boltzmann equation is considered. This equation in known as BGK-Boltzmann equation. It is shown that it preserves substantial properties of the Boltzmann equation. For example, it is illustrated that the macroscopic quantities which are obtained as moments of a particle density function fulfilling a BGK-Boltzmann equation also fulfil the conservation laws of mass and momentum. Finally, a family of BGK-Boltzmann equations with their corresponding macroscopic moments is considered. In particular, the behaviour of the stress tensor is analysed in a dedicated regime of small Knudsen and Mach numbers. It turns out that in this particular regime Newton's hypothesis can be affirmed.

1.3.1. About Collision Invariants and Conservation Laws. Of great importance for the derivations to come are so-called collision invariants. Firstly, with their characterising property it is possible to derive a solution for an equilibrium state of the Boltzmann equation (cf. Subsection 1.3.2). And secondly, a relationship of solutions of Boltzmann equations to the conservation laws can be established.

Definition 1.1. A local integrable function $\chi : \mathbb{R}^d \to \mathbb{R}$ is called *collision invariant* if for all $f \in L^1(I \times \Omega \times \mathbb{R}^d)$ with integrable χf, it holds:

$$\int_{\mathbb{R}^d} \chi(\boldsymbol{v}) \; J(f)(\boldsymbol{v}) \; d\boldsymbol{v} = 0 \ ,$$

whereby χ is called *basis collision invariant* if additionally holds

$$\chi(\boldsymbol{v_i}) + \chi(\boldsymbol{v_j}) = \chi(\boldsymbol{v'_i}) + \chi(\boldsymbol{v'_j})$$

for arbitrary $\boldsymbol{v_i}, \boldsymbol{v_j} \in \mathbb{R}^d$ where $\boldsymbol{v'_i}, \boldsymbol{v'_j} \in \mathbb{R}^d$ satisfying (1.44).

A characterising property of basis collision invariants is subject of the following theorem. It is stated according to Babovsky's record in [**8**].

Theorem 1.2. *A continous function χ is a basis collision invariant if and only if it can be written in the form*

$$\chi(\boldsymbol{v}) = a + \boldsymbol{b} \cdot \boldsymbol{v} + c \; \boldsymbol{v} \cdot \boldsymbol{v} \ ,$$

with $a, c \in \mathbb{R}$ and $\boldsymbol{b} \in \mathbb{R}^d$.

Proof. See for example [**8**, p. 48]. □

In Subsection 1.2.2 the density ρ, the velocity $\boldsymbol{v}$ and the stress tensor $\boldsymbol{P}$ are defined as moments of the particle density function f. Providing that these integrals exist for an f which satisfies the Boltzmann equation, it can be shown that ρ, $\boldsymbol{v}$ and $\boldsymbol{P}$ satisfy the conservation law for mass (1.8) and momentum (1.19). It directly follows from Theorem 1.2 that $\chi_0(\boldsymbol{v}) := m$ and $\chi_i(\boldsymbol{v}) := v_i$ for $i = 1, 2, ..., d$ are basis collision invariants. Multiplying the Boltzmann equation (1.54) with $\chi_i(\boldsymbol{v})$ for every $\boldsymbol{v} \in \mathbb{R}^3$ and integrating over the whole velocity space $\Xi = \mathbb{R}^d$ results in

$$\int_{\mathbb{R}^d} \chi_i(\boldsymbol{v}) \left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{\boldsymbol{F}}{m} \cdot \boldsymbol{\nabla_v} \right) f(t, \boldsymbol{r}, \boldsymbol{v}) \; d\boldsymbol{v} = 0 \qquad ((t, \boldsymbol{r}) \in I \times \Omega) \tag{1.55}$$

for $i = 0, 1, ..., d$.

For χ_0 (1.55) is

$$\frac{\partial}{\partial t}\rho + \boldsymbol{\nabla_r} \cdot (\rho \boldsymbol{u}) + \underbrace{\int_{\mathbb{R}^d} \boldsymbol{F} \cdot \boldsymbol{\nabla_v} f \; d\boldsymbol{v}}_{=0} = 0 \qquad \text{in } I \times \Omega \ , \tag{1.56}$$

at which the force term vanishes because Corollary 5.2 with $g \equiv 1$ and $\boldsymbol{a} = \boldsymbol{F}$ can be applied. What remains is the governing equation of the conservation law for mass.

Now, (1.55) is transformed for the cases $i = 1, 2, ..., d$. Again Corollary 5.2 is applied setting $g(\boldsymbol{v}) = v_i$ and $\boldsymbol{a} = \boldsymbol{F}/m$. Then, the following holds for $i = 1, 2, ..., d$ in $I \times \Omega$:

$$\frac{\partial}{\partial t}(\rho u_i) + \boldsymbol{\nabla_r} \cdot \int_{\mathbb{R}^d} v_i \boldsymbol{v} f \, d\boldsymbol{v} + \int_{\mathbb{R}^d} v_i \frac{\boldsymbol{F}}{m} \cdot \boldsymbol{\nabla_v} f \, d\boldsymbol{v} = 0$$

$$\frac{\partial}{\partial t}(\rho u_i) + \boldsymbol{\nabla_r} \cdot \int_{\mathbb{R}^d} (\boldsymbol{v} - \boldsymbol{u})(v_i - u_i) f + \boldsymbol{u} v_i f - \boldsymbol{v} u_i f - \boldsymbol{u} u_i f \, d\boldsymbol{v} + \int_{\mathbb{R}^d} \frac{F_i}{m} f \, d\boldsymbol{v} = 0$$

$$\frac{\partial}{\partial t}(\rho u_i) + \boldsymbol{\nabla_r} \cdot \boldsymbol{P_{i*}} + \boldsymbol{\nabla_r} \cdot (\boldsymbol{u} u_i \rho) - \boldsymbol{\nabla_r} \cdot (\boldsymbol{u} u_i \rho) - \boldsymbol{\nabla_r} \cdot (\boldsymbol{u} u_i \rho) + F_i \rho = 0$$

$$\frac{\partial}{\partial t}(\rho u_i) + \boldsymbol{\nabla_r} \cdot \boldsymbol{P_{i*}} - \boldsymbol{\nabla_r} \cdot (\boldsymbol{u} u_i \rho) + F_i \rho = 0 \ ,$$

where $\boldsymbol{P_{i*}}$ denotes row i of the tensor $\boldsymbol{P}$. It is nothing else but the governing equation of the law of momentum (1.19) in the conservative form.

1.3.2. Equilibrium State: Maxwellian Distribution. In this subsection the existence of solutions f with $f > 0$ in $I \times \Omega \times \mathbb{R}^d$ of

$$J(f) = 0 \qquad \text{in } I \times \Omega \ , \tag{1.57}$$

which is the right hand side of the Boltzmann equation, is studied. Such solutions exist and are commonly called *equilibrium states*. Based on the definitions of the macroscopic variables (1.37) to (1.40) the equilibrium states can be rewritten in forms known as *Maxwellian distributions*.

As a preliminary result, the following lemma holds

Lemma 1.3. *If* $\ln(f)J(f) \in L^1(\mathbb{R}^d)$ *for any* $t \in I$ *and* $\boldsymbol{r} \in \Omega$ *the so-called Boltzmann's inequality holds for all* f *with* $f > 0$ *in* $I \times \Omega \times \mathbb{R}^d$*:*

$$\int_{\mathbb{R}^d} \ln(f)J(f) \, d\boldsymbol{v} \leq 0 \qquad \textit{in } I \times \Omega \ , \tag{1.58}$$

where the equality sign applies if and only if

$$f' f'_* = f f_* \ \textit{almost everywhere in } I \times \Omega \times \mathbb{R}^d \ .$$

Proof. In [**25**, p. 78] a proof is stated for a more general collision operator J which includes the here chosen one. □

Since $\ln(f)$ is always strictly less than 0, the equivalence sign in (1.58) is valid if and only if (1.57) holds almost everywhere. Therefore, a direct consequence of Boltzmann's inequality is that (1.57) holds almost everywhere if and only if $\ln(f)$ is a collision invariant. If f is given by

$$f(t, \boldsymbol{r}, \boldsymbol{v}) = \exp\left(a(t, \boldsymbol{r}) + \boldsymbol{b}(t, \boldsymbol{r}) \cdot \boldsymbol{v} + c(t, \boldsymbol{r}) \ \boldsymbol{v} \cdot \boldsymbol{v}\right) \qquad \text{in } I \times \Omega \times \mathbb{R}^d$$

with functions $a, c : I \times \Omega \to \mathbb{R}$ and $b : I \times \Omega \to \mathbb{R}^d$, then $\ln(f)$ is a collision invariant due to Theorem 1.2. Thus, f fulfils (1.57), i.e. f is an equilibrium state.

On the other hand, if f is a solution of (1.57) with $f > 0$ in $I \times \Omega \times \mathbb{R}^d$, then also

$$\int_{\mathbb{R}^d} \ln(f)J(f) \, d\boldsymbol{v} = 0 \qquad \text{in } I \times \Omega \tag{1.59}$$

holds and $f' f'_* = f f_*$ almost everywhere in $I \times \Omega \times \mathbb{R}^d$. If f is further assumed to be continuous, condition (1.56) is fulfilled for $\chi = \ln(f)$, i.e. $\ln(f)$ is a basis collision invariant. From Theorem 1.2 it follows that f must be given as in (1.59).

The functions a,c and $\boldsymbol{b}$ can be replaced by meaningful macroscopic quantities. By setting

$$a = \ln\left(\frac{n}{(2\pi RT)^{d/2}}\right) - \frac{\boldsymbol{u}^2}{2RT} \ , \ \boldsymbol{b} = \frac{\boldsymbol{u}}{RT} \ , \ c = \frac{1}{2RT} \qquad \text{in } I \times \Omega \ ,$$

where $R \in \mathbb{R}_{>0}$ is the *universal gas constant* and $T \in \mathbb{R}_{>0}$ the *absolute temperature*, a continuous function for the equilibrium state is given. It is called the *Maxwellian distribution* and can be transformed to

$$f^{eq}(n, \boldsymbol{u}, T) : \begin{cases} I \times \Omega \times \mathbb{R}^d & \to \mathbb{R} \\ (t, \boldsymbol{r}, \boldsymbol{v}) & \mapsto \dfrac{n(t,\boldsymbol{r})}{(2\pi RT)^{d/2}} \exp\left(-\dfrac{(\boldsymbol{v}-\boldsymbol{u}(t,\boldsymbol{r}))^2}{2RT}\right) \end{cases} .$$

It is important to note that with respect to $\boldsymbol{v} \in \mathbb{R}^d$ $f^{eq}(n, \boldsymbol{u}, T)/n$ is a d-dimensional normal distribution with expectation $\boldsymbol{u}$ and covariance matrix $RT\boldsymbol{I}_d$. Therefore, the following three equivalences hold. On the one hand, they show the consistency of the choices of a, $\boldsymbol{b}$ and c. On the other hand, they reveal important properties of f^{eq} on those is referred to later on.

The first one uses the fact that f^{eq}/n is a density function:

$$\rho \overset{(1.38)}{=} m \int_{\mathbb{R}^d} f^{eq} \, d\boldsymbol{v} \ = mn = \rho \qquad \text{in } I \times \Omega \ . \tag{1.60}$$

The second equivalence is based on the expectation of f^{eq}/n. It is

$$\boldsymbol{u} \overset{(1.39)}{=} \frac{1}{n} \int_{\mathbb{R}^d} \boldsymbol{v} f^{eq} \, d\boldsymbol{v} \ = \boldsymbol{u} \qquad \text{in } I \times \Omega \ . \tag{1.61}$$

Finally, the third one relies on the covariance matrix of f^{eq}/n and the *ideal gas law*[3] which is

$$p = nRT \qquad \text{in } I \times \Omega \ . \tag{1.62}$$

Hence, in the case of considering perfect gases one gets

$$p \overset{(1.41)}{=} \frac{1}{d} \int_{\mathbb{R}^d} (\boldsymbol{v}-\boldsymbol{u})(\boldsymbol{v}-\boldsymbol{u}) f^{eq} \, d\boldsymbol{v} = \frac{n}{d} \sum_{i=1}^{d} RT = p \qquad \text{in } I \times \Omega \ .$$

1.3.3. About the H-Theorem. In the following, the so called *homogeneous Boltzmann equation* is considered:

$$\frac{\partial}{\partial t} f = J(f) \qquad \text{in } I \times \Omega \ . \tag{1.63}$$

It is a simplification of the Boltzmann equation (1.54) where the derivatives of f with respect to $\boldsymbol{r}$ vanish and no external forces act. Hence, f simplifies to a function of t and $\boldsymbol{v}$ which shall shorten the notation within this subsection. As it is shown in the last subsection a stationary solution f of (1.57) with $f > 0$ in $I \times \mathbb{R}^d$ is given by a Maxwellian distribution f^{eq}. The H-theorem states that these equilibrium states are the only continuous stationary solutions of (1.57) with $f > 0$ and that the so-called *H-functional*, which is defined by

$$H : \begin{cases} I & \to \mathbb{R} \\ t & \mapsto \displaystyle\int_{\mathbb{R}^d} \ln(f(t,\boldsymbol{v})) \, f(t,\boldsymbol{v}) \, d\boldsymbol{v} \end{cases}$$

never increases in time for any solution f with $f > 0$ of the homogeneous Boltzmann equation (1.63). This statement is rendered precisely in the following theorem:

[3]The ideal gas law is sometimes called *Boyle's law* (cf. for example [**25**]).

Theorem 1.4. *Providing f satisfies* (1.63) *and* $\ln(f)J(f) \in L^1(\mathbb{R}^d)$, *then the following holds*

(1) *H is monotonically decreasing in t.*
(2) *H is only constant in t if f is given as in* (1.59), *particularly if f is a Maxwellian distribution.*

Proof. Multiplying (1.63) with $1+\ln(f)$, integrating over $\boldsymbol{v} \in \mathbb{R}^d$, taking into account that 1 is a collision invariant and that $(1+\ln(f))\frac{\partial}{\partial t}f = \frac{\partial}{\partial t}(\ln(f)\ f)$ results in

$$
\begin{aligned}
\int_{\mathbb{R}^d} (1+\ln(f))\frac{\partial}{\partial t} f \ d\boldsymbol{v} &= \int_{\mathbb{R}^d} (1+\ln(f))J(f)\ d\boldsymbol{v} && \text{in } I\\
\int_{\mathbb{R}^d} \frac{\partial}{\partial t}(\ln(f)\ f)\ d\boldsymbol{v} &= \int_{\mathbb{R}^d} \ln(f)J(f)\ d\boldsymbol{v} && \text{in } I\\
\frac{\partial}{\partial t}H(t) &= \int_{\mathbb{R}^d} \ln(f)J(f)\ d\boldsymbol{v} && \text{in } I\ .
\end{aligned}
$$

Then, according to Boltzmann's inequality (cf. Lemma 1.3) the right hand side is always less than 0 and, as presented in the previous subsection, only equal to 0 if and only if f is given as in (1.59). □

It is to be mentioned that this fundamental property for solutions of the homogeneous Boltzmann equation is also given for solutions of the Boltzmann equation in the case that no external force is applied to the gas and further no heat is transfered at the boundary of the gas (cf. *generalised H-theorem* e.g. in [**25**]).

1.3.4. BGK Approximation of the Collision Operator. When dealing with the Boltzmann equation one of the main shortcomings is the complex structure of the collision operator J. Bhatnagar, Gross and Krook proposed in 1954 an elegant simplification [**19**]. The so-called *BGK collision operator* is defined by

$$
Q(f) := -\frac{1}{\omega}(f - M_f^{eq}) \qquad \text{in } I \times \Omega \times \mathbb{R}^d\ , \tag{1.64}
$$

where $M_f^{eq} := f^{eq}(n_f, \boldsymbol{u_f}, T_f)$ is a particular Maxwellian distribution with the moments of f serving as suitable choices. To clarify the notation, the moments of f are endued with the selfsame index and the absolute temperature T_f is given implicitly by the ideal gas law (1.62), i.e. $p_f = n_f R T_f$. ω denotes the *(expected) mean free time* between two collisions which is qualified in the following in (1.66).

Recapitulating the last two subsections, the Maxwellian distribution f^{eq} quantifies a stationary state by means of the expected distribution of atoms of an ideal gas in a closed volume where no external forces are applied and no heat is transfered at the boundary. For that configuration the *(expected) mean absolute thermal velocity* $\bar{c}$ is defined and transformed by

$$
\begin{aligned}
\bar{c} &:= \frac{1}{n}\int_{\mathbb{R}^3} \|\boldsymbol{v}-\boldsymbol{u}\| f^{eq}\ 3\boldsymbol{v}\\
&= \int_{\mathbb{R}^3} \frac{\|\widetilde{\boldsymbol{v}}\|}{(2\pi RT)^{3/2}} \exp\left(-\frac{(\|\widetilde{\boldsymbol{v}}\|^2)}{2RT}\right)\ 3\widetilde{\boldsymbol{v}}\\
&= 4\pi \int_0^\infty \frac{\|\widetilde{\boldsymbol{v}}\|^3}{(2\pi RT)^{3/2}} \exp\left(-\frac{(\|\widetilde{\boldsymbol{v}}\|^2)}{2RT}\right)\ 3\|\widetilde{\boldsymbol{v}}\|\\
&= 4\pi \frac{1}{(2\pi RT)^{3/2}} \frac{(2RT)^2}{2} = \sqrt{\frac{8}{\pi}RT}\ .
\end{aligned} \tag{1.65}
$$

The *(expected) root mean square thermal velocity* c_0 is defined and transformed similarly to the derivation presented above:

$$c_0 := \sqrt{\frac{1}{n}\int_{\mathbb{R}^3} (\boldsymbol{v}-\boldsymbol{u})^2 f^{eq}\ 3\boldsymbol{v}} = ... = \sqrt{3RT}\ .$$

With the mean free path l_f and mean absolute thermal velocity $\overline{c}$ one obtains the *(expected) mean free time* between two collisions as

$$\omega := \frac{l_f}{\overline{c}}\ . \tag{1.66}$$

The particular choice for Q is motivated by the idea to retain three essential features of the collision operator J. These properties of Q are the subject of the following theorem:

Theorem 1.5. *Let Q be defined as in* (1.64), $\chi(\boldsymbol{v}) := a + \boldsymbol{b}\cdot\boldsymbol{v} + c\ \boldsymbol{v}\cdot\boldsymbol{v}$ *for any* $a, c \in \mathbb{R}$ *and* $\boldsymbol{b} \in \mathbb{R}^d$ *and* $f, \chi f \in L^1(I\times\Omega\times\mathbb{R}^d)$ *with* $f > 0$ *in* $\times\Omega\times\mathbb{R}^d$. *Then, the following holds:*

(1) *χ is a collision invariant ,*
(2) $\int_{\mathbb{R}^d} \ln(f)Q(f)\ d\boldsymbol{v} \leq 0$ *,*
(3) $\int_{\mathbb{R}^d} \ln(f)Q(f)\ d\boldsymbol{v} = 0$ *if and only if f is a Maxwellian distribution .*

Proof. Let χ and f be as demanded. Taking advantage of the properties of M_f^{eq} (1.60), (1.61) and (1.62) yields

$$\begin{aligned}\int_{\mathbb{R}^d} \chi Q(f)\ d\boldsymbol{v} &= -\omega\int_{\mathbb{R}^d} (a + \boldsymbol{b}\cdot\boldsymbol{v} + c\ \boldsymbol{v}^2)(f - M_f^{eq})\ d\boldsymbol{v} \\ &= -\omega\int_{\mathbb{R}^d} (a - 2c{\boldsymbol{u_f}}^2 + (\boldsymbol{b} + 2c\boldsymbol{u_f})\cdot\boldsymbol{v} + c\ (\boldsymbol{v}-\boldsymbol{u_f})^2)(f - M_f^{eq})\ d\boldsymbol{v} \\ &= 0\ ,\end{aligned}$$

in $I\times\Omega$ which proves the first statement. The second proposition is proved by the following manipulations which hold in $I\times\Omega$:

$$\begin{aligned}\int_{\mathbb{R}^d} \ln(f)Q(f)\ d\boldsymbol{v} &= \int_{\mathbb{R}^d} \ln(f/M_f^{eq})Q(f)\ d\boldsymbol{v} + \underbrace{\int_{\mathbb{R}^d} \ln(M_f^{eq})Q(f)\ d\boldsymbol{v}}_{=0} \\ &= -\frac{1}{\omega}\int_{\mathbb{R}^d} \underbrace{M_f^{eq}}_{\geq 0}\underbrace{(1 - f/M_f^{eq})\ln(f/M_f^{eq})}_{\geq 0}\ d\boldsymbol{v} \leq 0\ ,\end{aligned}$$

whereby the second sum is zero because $\ln(M_f^{eq})$ is a collision invariant of Q. This transformation also reveals that the equality sign applies if and only if $f = M_f^{eq}$ which proves the third statement. $\square$

Replacing the collision operator J of the Boltzmann equation by Q leads to the *BGK-Boltzmann equation*:

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla_r} + \frac{\boldsymbol{F}}{m}\cdot\boldsymbol{\nabla_r}\right) f = Q(f) \text{ in } I\times\Omega\times\mathbb{R}^d\ . \tag{1.67}$$

The common properties of the collision operator for rigid spheres and the BGK collision operator lead to two common characteristics of the Boltzmann and BGK-Boltzmann equation. The first one is the relation of solutions of the Boltzmann equation to the conservation laws of mass and momentum. This is the subject of Subsection 1.3.1. There, it is established that the moments of a particle density function f which solves the Boltzmann equation respect the two conservation laws.

Due to the form of the collision invariants of Q, the consistency of the moments of a solution of the BGK-equation with the corresponding macroscopic quantities can be established as well, i.e. the moments also fulfil the governing equations of the conservations. The second common characteristic of the Boltzmann and BGK-Boltzmann equation is the relaxation towards an equilibrium state which is a Maxwellian distribution. This is established by the H-Theorem which also holds for the homogeneous form of the BGK-Boltzmann equation due the properties of the BGK collision operator.

1.3.5. About a Hydrodynamic Limit for the BGK-Boltzmann equation. This subsection aims to establish a connection of the moments obtained from a solution of a BGK-Boltzmann equation with the incompressible Navier-Stokes equation. In the previous subsection it is shown that these moments, namely mass density, velocity and stress tensor, fulfil the conservation laws of mass and momentum. Newton's characterising hypothesis for incompressible Newtonian fluids is still left to be established for the mesoscopic model. It postulates that the stress tensor is of the form

$$\boldsymbol{P} = -p\boldsymbol{I}_d + 2\mu\boldsymbol{D} \ , \tag{1.68}$$

where the dynamic viscosity μ is a material describing constant and $\boldsymbol{D}$ is the strain rate tensor. For this purpose, in the following the BGK-Boltzmann equation is studied in a particular regime of small Knudsen numbers Kn and small Mach numbers Ma. Thereunto, the equation is rewritten as a sequence of higher-order equations leading to a power series. Neglecting the terms of order higher than one, an equation is obtained for which the stress tensor corresponding to a solution satisfies Newton's hypothesis.

To begin with, a *characteristic length* $L \in \mathbb{R}_{>0}$ is introduced which characterises the magnitude of the domain Ω. Further, a *characteristic speed* $U \in \mathbb{R}_{>0}$ is fixed which stands for a typical magnitude of a macroscopic flow velocity. With the mean free path l_f and the mean absolute thermal velocity $\bar{c}$ two typical microscopic magnitudes are given. According to (1.65), the latter is $\bar{c} = \sqrt{\frac{8}{\pi}RT}$ and therefore, the *speed of sound* $c_s = \sqrt{\gamma RT}$ with the *adiabatic exponent* γ (e.g. $\gamma = \frac{5}{3}$ for a monatonic gas) is of the same order of magnitude as $\bar{c}$. Alltogether, the following two dimensionless quantities are defined:

- *Knudsen number*
$$Kn := l_f/L \ , \tag{1.69}$$
- *Mach number*
$$Ma := U/c \ . \tag{1.70}$$

With these two parameters Kn and Ma the terminology of small in the above mentioned context can be quantified. A limiting process where Kn and Ma tend to zero implies that l_f will also tends to zero while c_s or $\bar{c}$, respectively, will tend to infinity since L and U are constants.

Now, the kinematic viscosity ν and dynamic viscosity μ are defined by

$$\nu := \frac{\pi}{8}\bar{c}l_f \tag{1.71}$$

$$\mu := \nu\rho \tag{1.72}$$

Considering viscous fluid flows in a macroscopic framework as presented in Section 1.1 μ is assumed to be a constant which describes the fluid. Hence, in a limiting process l_f and $1/\bar{c}$ must tend to zero with the some order of magnitude.

A third dimensionless parameter which is of interest later on is the *Reynolds number*

$$Re := \frac{\rho\, U\, L}{\mu} = \frac{\pi\, \gamma}{2} \rho \frac{Ma}{Kn} \ . \tag{1.73}$$

In case of considering incompressible fluids ρ is a constant. With (1.73) it becomes clear that the Reynolds number Re is proportional to the ratio of the Mach to the Knudsen number.

Using the Lagrange description, the BGK-Boltzmann equation (1.67) can be transformed to

$$f = M_f^{eq} - \frac{l_f}{\bar{c}} \frac{d}{dt} f \qquad \text{in } I \times \Omega \times \mathbb{R}^d \ . \tag{1.74}$$

Differentiating (1.74) with respect to t yields

$$\frac{d}{dt} f = \frac{d}{dt} M_f^{eq} - \frac{l_f}{\bar{c}} \frac{d^2}{dt^2} f \qquad \text{in } I \times \Omega \times \mathbb{R}^d \ . \tag{1.75}$$

Now, (1.75) serves to substitute $\frac{d}{dt} f$ in (1.74):

$$f = M_f^{eq} - \frac{l_f}{\bar{c}} \frac{d}{dt} M_f^{eq} + \frac{l_f^2}{\bar{c}^2} \frac{d^2}{dt^2} f \qquad \text{in } I \times \Omega \times \mathbb{R}^d \ .$$

This process can be repeated again and again leading to higher order equations. If $h := l_f / \bar{c}$ tends to zero terms of the equation become singular and the question arises how does a solution f_h behave in such a limiting process. The physical meaning of the limit admits that at least the moments of f_h converge. The sequence of higher order equations leads to a power series in h around t which is known as *Chapman-Enskog expansion*:

$$f = \sum_{i=0}^{\infty} \left(-\frac{l_f}{\bar{c}} \right)^i \frac{d^i}{dt^i} M_f^{eq} \qquad \text{in } I \times \Omega \times \mathbb{R}^d \ . \tag{1.76}$$

Note that also M_f^{eq} depends on h which makes a rigorous analysis difficult. For example, questions regarding convergence are controversially discussed in the literature (cf. [**25, 140**]).

To motivate the definition of the viscosity (1.71) which is conform with Stokes' postulates (cf. Section 1.1) an ansatz of the form

$$f = M_f^{eq} - \frac{l_f}{\bar{c}} \frac{d}{dt} M_f^{eq} \qquad \text{in } I \times \Omega \times \mathbb{R}^d \tag{1.77}$$

is chosen. Hereby, it is assumed that higher order terms vanish for $h \to 0$. What follows is similar to the derivation in [**140**] where the absolute temperature T is assumed to be constant. Applying the conservation law of mass (1.8) the first derivative of $M_{f,h}^{eq}$ is given and simplifies as follows:

$$\frac{d}{dt}M_f^{eq} = \left(\frac{1}{\rho}\frac{d}{dt}\rho + \frac{\boldsymbol{c}}{RT}\cdot\frac{d}{dt}\boldsymbol{u} - \frac{\boldsymbol{c}}{RT}\cdot\frac{\boldsymbol{F}}{m}\right)M_f^{eq}$$

$$\frac{d}{dt}M_f^{eq} = \left(\frac{1}{\rho}\left(\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla_r}\right)\rho + \frac{\boldsymbol{c}}{RT}\cdot\left(\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla_r}\right)\boldsymbol{u} - \frac{\boldsymbol{c}}{RT}\cdot\frac{\boldsymbol{F}}{m}\right)M_f^{eq}$$

$$\frac{d}{dt}M_f^{eq} = \left(\frac{1}{\rho}\left(-\boldsymbol{u}\cdot\boldsymbol{\nabla_r}\rho - \rho\boldsymbol{\nabla_r}\cdot\boldsymbol{u} + \boldsymbol{v}\cdot\boldsymbol{\nabla_r}\rho\right)\right)$$
$$+ \left(\frac{\boldsymbol{c}}{RT}\cdot\left(\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla_r}\right)\boldsymbol{u} - \frac{\boldsymbol{c}}{RT}\cdot\frac{\boldsymbol{F}}{m}\right)M_f^{eq}$$

$$\frac{d}{dt}M_f^{eq} = \left(-\underbrace{\boldsymbol{\nabla_r}\cdot\boldsymbol{u}}_{:=a_f} + \underbrace{\frac{\boldsymbol{c}}{\rho}\cdot\boldsymbol{\nabla_r}\rho}_{:=b_f} + \underbrace{\frac{\boldsymbol{c}}{RT}\cdot\frac{\partial}{\partial t}\boldsymbol{u}}_{:=c_f} + \underbrace{\frac{\boldsymbol{c}}{RT}\cdot\left(\boldsymbol{v}\cdot\boldsymbol{\nabla_r}\right)\boldsymbol{u}}_{:=d_f} - \underbrace{\frac{\boldsymbol{c}}{RT}\cdot\frac{\boldsymbol{F}}{m}}_{:=e_f}\right)M_f^{eq}$$

whereas $\boldsymbol{c} := \boldsymbol{v} - \boldsymbol{u}$ and all transformations hold in $I\times\Omega\times\mathbb{R}^d$. Inserting the derivative in (1.77) yields

$$f = M_f^{eq}\left(1 - \frac{l_f}{\bar{c}}\left(-a_f + b_f + c_f + d_f - e_f\right)\right) \qquad \text{in } I\times\Omega\times\mathbb{R}^d\ .$$

To obtain the to this f corresponding stress tensor, designated integrals need to be evaluated. Thereunto, it can be taken advantage of the symmetric properties of the the function M_f^{eq} and further of the fact that M_f^{eq}/n is a normal distribution with covariance matrix $RT\boldsymbol{I}_d$. In $I\times\Omega$ and for any $i,j,k,l\in\{1,2,...,d\}$ the following transformations hold:

$$\int_{\mathbb{R}^d} c_i c_j M_f^{eq}\ d\boldsymbol{v} = \delta_{ij}\ n\ RT\ ,$$
$$\int_{\mathbb{R}^d} c_i c_j a_f M_f^{eq}\ d\boldsymbol{v} = \frac{\partial}{\partial r_k}u_k\int_{\mathbb{R}^d} c_i c_j M_f^{eq}\ d\boldsymbol{v} = \delta_{ij}\ nRT\frac{\partial}{\partial r_k}u_k\ ,$$
$$\int_{\mathbb{R}^d} c_i c_j b_f M_f^{eq}\ d\boldsymbol{v} = \frac{1}{\rho}\frac{\partial}{\partial r_k}\rho\int_{\mathbb{R}^d} c_i c_j c_k M_f^{eq}\ d\boldsymbol{v} = 0\ ,$$
$$\int_{\mathbb{R}^d} c_i c_j c_f M_f^{eq}\ d\boldsymbol{v} = \frac{1}{RT}\frac{\partial}{\partial t}u_k\int_{\mathbb{R}^d} c_i c_j c_k M_f^{eq}\ d\boldsymbol{v} = 0\ ,$$
$$\int_{\mathbb{R}^d} c_i c_j d_f M_f^{eq}\ d\boldsymbol{v} = \frac{1}{RT}\frac{\partial}{\partial r_k}u_l\int_{\mathbb{R}^d} c_i c_j v_k c_l M_f^{eq}\ d\boldsymbol{v}\ ,$$
$$= (\delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})nRT\frac{\partial}{\partial r_k}u_l\ ,$$
$$\int_{\mathbb{R}^d} c_i c_j e_f M_f^{eq}\ d\boldsymbol{v} = \frac{F_k}{mRT}\int_{\mathbb{R}^d} c_i c_j c_k M_f^{eq}\ d\boldsymbol{v} = 0\ .$$

Based on the preparative work, each component p_{ij} $(i,j = 1,2,...,d)$ of the stress tensor $\boldsymbol{P}$ can be stated and further simplified according to

$$p_{ij} = \int_{\mathbb{R}^d}\left(1 - \frac{l_f}{\bar{c}}\left(-a_f + b_f + c_f + d_f - e_f\right)\right)M_f^{eq}\ d\boldsymbol{v} \qquad \text{in } I\times\Omega\times\mathbb{R}^d$$
$$= \delta_{ij}nRT + \frac{nRTl_f}{\bar{c}}\left(\frac{\partial}{\partial r_i}u_j + \frac{\partial}{\partial r_j}u_i\right) \qquad \text{in } I\times\Omega\times\mathbb{R}^d$$
$$= \delta_{ij}p + \mu\left(\frac{\partial}{\partial r_i}u_j + \frac{\partial}{\partial r_j}u_i\right) \qquad \text{in } I\times\Omega\times\mathbb{R}^d\ .$$

This is exactly Newton's hypothesis (1.68).

The assumption of vanishing higher order terms for the limit $h \to 0$ is justified in [**12, 117**] for the special case $\Omega = \mathbb{R}^3$ and initial conditions which are close to an absolute equilibrium $M^{eq}_{abs} := f^{eq}(1, \mathbf{0}, 1)$. In [**12**] Bardos et al. consider fluid dynamic limits of kinetic equations like the Boltzmann and BGK-Boltzmann equation. Among them, one finds the limit towards the incompressible Navier-Stokes equation. Saint-Raymond concentrates especially on the BGK-Boltzmann equation [**117**]. Here, some integrability assumption for f needed for the more general proofs in [**12**] are dropped. In both works, the underlying kinetic equations are considered in a particular scaled form. The force term is neglected and an appropriate choice of Galilean frame and dimensionless units is used such that the BGK-Boltzmann equation reads

$$\left(h\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r}\right) f = -\frac{1}{h\nu}\left(f - M^{eq}_f\right) \qquad \text{in } I \times \Omega \times \mathbb{R}^d\ . \tag{1.78}$$

Solutions of (1.78) are sought in the form

$$f = M^{eq}_{abs}(1 + hg_h) \qquad \text{in } I \times \Omega \times \mathbb{R}^d\ ,$$

where g_h is a perturbation function depending on h. Under appropriate conditions a limiting g can be derived. Then, the explicit form of g comprises a weak solution of the incompressible Navier-Stokes equation.

Many questions concerning the relation of macroscopic and mesoscopic models, which are not addressed in this work, have sill not been answered yet. In this context to be noted is the work of Bardos, Golse and Levermore [**12**] Freireisl [**34**], Freireisl and Novotný [**35**] and Lions [**94**] which dedicate their work *inter alia* into this direction.

CHAPTER 2

Numerical Simulation of Incompressible Newtonian Fluid Flows with Lattice Boltzmann Methods

Historically, lattice Boltzmann methods (LBM) have their seeds in lattice gas cellular automata (LGCA) methods, which were mainly developed in the seventies and eighties of the 20th century to simulate fluid flows, predominantly to solve the Navier-Stokes equation. To be mentioned are the HPP and FHP models, named after the innovators Hardy, Pomeau and Pazzis [**61**] and, respectively, Frisch, Hasslacher and Pomeau [**42**]. LGCA schemes allow to retrace the movement (free flow and collision) of a given finite number of moving particles. Macroscopic quantities like the velocity, the pressure or the stress tensor are obtained from the distribution of the particles. Characteristical for the schemes is the choice of the considered uniform grids, commonly denoted as *lattices*. For example, a two-dimensional position space is approximated by a uniform triangular grid. Each considered particle is equipped with a certain direction, chosen so that it reaches exactly a node of the lattice after a so-called *propagation step* (or *streaming step*). In physical quantities, the direction represents a velocity and the streaming step a constant time interval during which a particle flies freely without colliding with another one. If more than one particle arrive at a node, they collide, i.e. they change their current flight directions according to rules that conserve mass and momentum. This procedure is called the *collision step*. Then, another propagation step follows and so on. The scheme ensures that any of the considered particles is always at one of the lattice nodes moving in one of the dedicated directions. Thus, only a finite number of states of the whole system is possible so that it can be described by means of a vector of Booleans. However, the Boolean nature of LGCA results in a statistical noise. This motivated the transition towards lattice Boltzmann (LB) schemes where the number of moving particles with a dedicated velocity is replaced by an average. This was invented by McNamara and Zanetti [**99**] in 1988. Since the conservation of mass is ensured, these averages form a probability density function which is related to Boltzmann distributions. On account of this found relation, the development accelerates. The explicit collision rules of LGCA were replaced by the BGK collision operator (cf. e.g. Koelman in [**82**]) and later by more complex approximations of the Boltzmann collision operator.

Motivated by the intrinsic relation to the Boltzmann equation, the seemingly rather artificial and unphysical LGCA methods turned to methods (LBM) with a physical meaning. From a mathematical point of view, the interpretation of LBM was and still is an object of many investigations. Here, mainly two questions concerning consistency, stability and consequently convergence arise: The first one concerns the relation of LBM to the Boltzmann equation or its approximations and the second one its relation to certain macroscopic equations like the Euler or Navier-Stokes equation.

Some lattice Boltzmann schemes can be interpreted as explicit finite difference approximations of specially scaled Boltzmann or Boltzmann-like equations, such as the multi relaxation time (MRT) (cf. d'Humieres et al. [**31**] and references therein) or the BGK approximation of the Boltzmann equation. For example, in [**63**] He and Lou present a mostly general procedure to derive certain LBM. In a limiting process, the moments of assumed discrete solutions are related to solutions of macroscopic continuous equations such as the Euler or incompressible Navier-Stokes equation. For the incompressible Newtonian fluid flows, He and Lou [**62**] apply a Chapman-Enskog expansion to an assumed solution of a discrete version of the BGK-Boltzmann equation and show that it approximates a solution of the compressible Navier-Stokes equation for small Mach numbers. In a second limiting process, they relate the solution to that of the incompressible Navier-Stokes equation. However, this *multi-scale* analysis is often criticised with regard to mathematical rigour (cf. e.g. Wolf-Gladrow [**140**]) and references therein). Using a diffusive scaling (cf. [**125**]) and Taylor expansion, in [**76**] Junk et al. link an assumed expanded solution of certain LB schemes directly to the incompressible Navier-Stokes equation and analyse the asymptotic behaviour. In this procedure only one variable is required for the limiting process. The work of Junk and Klar [**75**] is also interesting to note. They succeed in interpreting a particular LB scheme as Chorin-type projection method for the incompressible Navier-Stokes equation.

Yet, only in a few articles the issues of numerical stability and finally convergence of LB schemes are addressed. Despite publications examining accuracy and stability by numerical tests, only very special LBM are considered in a mathematically rigorous manner. Most of them consider one-dimensional cases, e.g. the works of Weiß [**138**] and Rheinländer [**115**]. Weiß proves the consistency and stability of a particular LB model for the heat equation on bounded intervals. Rheinländer dedicates his work in particular to the analysis of numerical phenomena like initial layers, multiple time scales and boundary layers. He studies e.g. the consistency and stability of several LBM to the advection-diffusion equation in one space dimension.

The first section of this chapter is dedicated to derive selected LBM which are consistent to a family of BGK-Boltzmann equations. A framework and a formal procedure to establish the consistency is given in detail. It is stressed that this approach is in contrast to many others, since the derivation does not depend on any argument or assumption related to a macroscopic target equation. The section concludes with a short discussion about the formulation of boundary and initial conditions for LBM. In the second section topics related to the implementation of LB algorithms are addressed. Thereto, dedicated concepts are studied following the realisations in the open source software OpenLB as an example. Two numerical experiments complete this chapter. The first example aims for showing the connection of LBM to the incompressible Navier-Stokes equation. Thereto, an analytical stationary solution of the macroscopic governing equation is given, which is to be reached by applying certain LB schemes. The second numerical experiment is done to solve the famous benchmark problem *lid-driven cavity.* This is realised by modelling the problem both macroscopically and mesoscopically and solving the resulting governing equation by a *finite element method* (FEM) and an LBM, respectively. Both solutions are finally compared with experimental data.

2.1. From BGK-Boltzmann Equations to Lattice Boltzmann Equations

The basic and common idea of all LBM is the coupling of a discretisation parameter $h \in \mathbb{R}_{>0}$ with one or more scaling parameters of the Boltzmann or a Boltzmann-like equation. Hereby, the modality of the connection depends on the regime in which a macroscopic target equation is reached by the mesoscopic equations. The subclass of the here considered LBM leads to a discrete representation $\mathcal{G}$ of that family of BGK-Boltzmann equations $\mathcal{F}$ which is related to the incompressible Navier-Stokes equation. The obtained family $\mathcal{G}$ consists of equations referred to as *lattice Boltzmann equations* which are defined by (2.11).

The particular regime arises from the hydrodynamic relation addressed in Subsection 1.3.5. As it is shown there, the mean free path l_f, the mean absolute thermal velocity $\bar{c}$ and the kinematic viscosity ν are related by (1.71). Therefore, by assuming ν is given as a constant characterising a fluid and by setting the speed of sound

$$c_s = \sqrt{3RT} := \frac{1}{h} \tag{2.1}$$

one finds

$$\bar{c} = \sqrt{\frac{8}{3\pi}}\frac{1}{h}, \ l_f = \sqrt{\frac{24}{\pi}}\nu h, \ \omega = 3\nu h^2 \ . \tag{2.2}$$

With it and after multiplying the BGK-Boltzmann equation (1.67) with h^2 on both sides it becomes

$$h^2 \frac{d}{dt} f = -\frac{1}{3\nu}\left(f - M_f^{eq}\right) \qquad \text{in } I \times \Omega \times \mathbb{R}^d \ , \tag{2.3}$$

where the Maxwellian distribution is

$$M_f^{eq} = \frac{n_f \ h^d}{\left(\frac{2}{3}\pi\right)^{d/2}} \exp\left(-\frac{3}{2}\left(\boldsymbol{v}\ h - \boldsymbol{u_f}\ h\right)^2\right) \qquad \text{in } I \times \Omega \times \mathbb{R}^d \ .$$

With it, one obtains the considered *family of BGK-Boltzmann equations* $\mathcal{F}$ according to

$$\mathcal{F} := \left(h^2 \frac{d}{dt} f + \frac{1}{3\nu}\left(f - M_f^{eq}\right) = 0 \ , \ f \in V(I \times \Omega \times \mathbb{R}^d)\right)_{h>0} \ . \tag{2.4}$$

To cope with the coupling of the parameter h, the frequently used term of *consistency* which connects a family of discrete equations to one single continuous equation needs to be extended. Thereto, the following definition is stated:

Definition 2.1. Let $\mathcal{F} := (F_h(f) = 0, f \in V(X))_{h>0}$ and $\mathcal{G} := (G_h(g) = 0, g \in U_h(X_h) \subseteq V)_{h>0}$ be families of equations. Then, $\mathcal{G}$ is called *consistent of order d* to $\mathcal{F}$ in $U_h(X_h)$, if for the solutions f^h of $F_h(f^h) = 0$ in X there holds

$$G_h(f^h|_{U_h}) \in \mathcal{O}(h^d) \text{ in } X_h \ , \text{ i.e. } \lim_{h \searrow 0} \sup_{x \in X_h} \|\frac{G_h(f^h|_{U_h})(x)}{h^d}\| < \infty \ .$$

The terms $G_h(f^h|_{U_h})$ in X_h are understood as functions of h and called the *truncation errors.*

In the following, the family of lattice Boltzmann equations $\mathcal{G}$ is consistently derived from the family of BGK-Boltzmann equations $\mathcal{F}$ by basically applying chosen LBM, namely the so-called $D2Q9$ and $D3Q19$ model. The derivation is established in three main steps. In the first one, all discrete spaces are defined. Based on this, in the next step the *family of velocity discrete BGK-Boltzmann* *equations* $\mathcal{FG}$ is derived by replacing the velocity space by a discrete space. The position and time space remain to be replaced by their discrete counterparts. This is done in the last step by means of finite difference approximations which finally lead to the wanted family of lattice Boltzmann equations $\mathcal{G}$.

2.1.1. Discrete Time and Phase Space. Let $h \in \mathbb{R}_{>0}$ denote the discretisation parameter. It represents the shortest distance between any two nodes of a uniform grid which approximates the position space Ω. The grid is commonly called the *lattice*. The set containing all inner nodes is denoted by Ω_h, while Γ_h denotes the set that contains all other nodes. In Figure 2.1 an example is given to illustrate the approximation.

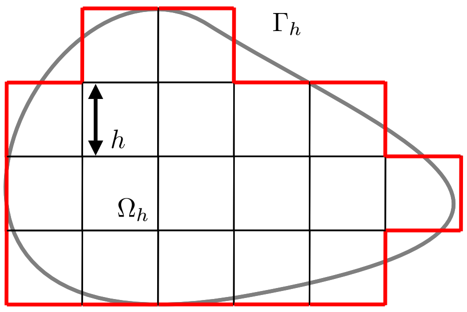


FIGURE 2.1. The considered domain Ω with boundary Γ is approximated by a uniform grid with spacing h which is is commonly called the lattice. The set containing all inner nodes is denoted by Ω_h while Γ_h denotes the set that contains all other nodes.

The velocity space is discretised by a small number $q \in \mathbb{N}$ of chosen velocities v_i, $i = 0, 1, ..., q-1$. The finite space of all velocities is denoted by $Q := \{v_i : i = 0, 1, ..., q-1\}$. They are chosen such that a fictitious particle which is at a knot of the lattice at a fixed time $t = t_0$ will still be at its position or at one of a close neighbouring knot of the lattice at $t = t_0 + h^2$. The discrete time space is then given by $I_h := \big\{t \in I : t = t_0 + h^2 k, k \in \mathbb{N}\big\}$.

If d denotes the space dimension, $DdQq$ commonly denotes the name of the particular LB model. Often used models are the $D1Q3$, $D2Q9$, $D3Q15$, $D3Q19$ and $D3Q27$ model (cf. [**28, 67, 129, 140**]). In the following, two models are considered exemplarily, namely the $D2Q9$ model for which the 9 discrete velocities are given by

$$\begin{aligned}
&\boldsymbol{v}_0 = \begin{pmatrix}0\\0\end{pmatrix}, \ \boldsymbol{v}_1 = \frac{1}{h}\begin{pmatrix}1\\0\end{pmatrix}, \ \boldsymbol{v}_2 = \frac{1}{h}\begin{pmatrix}1\\1\end{pmatrix}, \ \boldsymbol{v}_3 = \frac{1}{h}\begin{pmatrix}0\\1\end{pmatrix}, \ \boldsymbol{v}_4 = \frac{1}{h}\begin{pmatrix}-1\\1\end{pmatrix}, \\
&\boldsymbol{v}_5 = \frac{1}{h}\begin{pmatrix}-1\\0\end{pmatrix}, \ \boldsymbol{v}_6 = \frac{1}{h}\begin{pmatrix}-1\\-1\end{pmatrix}, \ \boldsymbol{v}_7 = \frac{1}{h}\begin{pmatrix}0\\-1\end{pmatrix}, \ \boldsymbol{v}_8 = \frac{1}{h}\begin{pmatrix}1\\-1\end{pmatrix}
\end{aligned}$$

and the $D3Q19$ model where the 19 discrete velocities are

$$\boldsymbol{v}_0 = \begin{pmatrix}0\\0\\0\end{pmatrix}, \ \boldsymbol{v}_1 = \frac{1}{h}\begin{pmatrix}-1\\-1\\0\end{pmatrix}, \ \boldsymbol{v}_2 = \frac{1}{h}\begin{pmatrix}1\\0\\1\end{pmatrix}, \ \boldsymbol{v}_3 = \frac{1}{h}\begin{pmatrix}-1\\0\\0\end{pmatrix}, \ \boldsymbol{v}_4 = \frac{1}{h}\begin{pmatrix}-1\\0\\-1\end{pmatrix},$$

$$\boldsymbol{v}_5 = \frac{1}{h}\begin{pmatrix}-1\\1\\0\end{pmatrix}, \ \boldsymbol{v}_6 = \frac{1}{h}\begin{pmatrix}0\\1\\1\end{pmatrix}, \ \boldsymbol{v}_7 = \frac{1}{h}\begin{pmatrix}0\\-1\\0\end{pmatrix}, \ \boldsymbol{v}_8 = \frac{1}{h}\begin{pmatrix}0\\-1\\1\end{pmatrix}, \ \boldsymbol{v}_9 = \frac{1}{h}\begin{pmatrix}0\\0\\1\end{pmatrix},$$

$$\boldsymbol{v}_{10} = \frac{1}{h}\begin{pmatrix}1\\1\\0\end{pmatrix}, \ \boldsymbol{v}_{11} = \frac{1}{h}\begin{pmatrix}-1\\0\\-1\end{pmatrix}, \ \boldsymbol{v}_{12} = \frac{1}{h}\begin{pmatrix}1\\0\\0\end{pmatrix}, \ \boldsymbol{v}_{13} = \frac{1}{h}\begin{pmatrix}1\\0\\1\end{pmatrix}, \ \boldsymbol{v}_{14} = \frac{1}{h}\begin{pmatrix}1\\-1\\0\end{pmatrix},$$

$$\boldsymbol{v}_{15} = \frac{1}{h}\begin{pmatrix}0\\-1\\-1\end{pmatrix}, \ \boldsymbol{v}_{16} = \frac{1}{h}\begin{pmatrix}0\\1\\0\end{pmatrix}, \ \boldsymbol{v}_{17} = \frac{1}{h}\begin{pmatrix}0\\1\\-1\end{pmatrix}, \ \boldsymbol{v}_{18} = \frac{1}{h}\begin{pmatrix}0\\0\\-1\end{pmatrix}.$$

An extract of 27 knots of a lattice and the discrete velocities of the middle knot is depicted in Figure 2.2. It illustrates the discretised phase space.

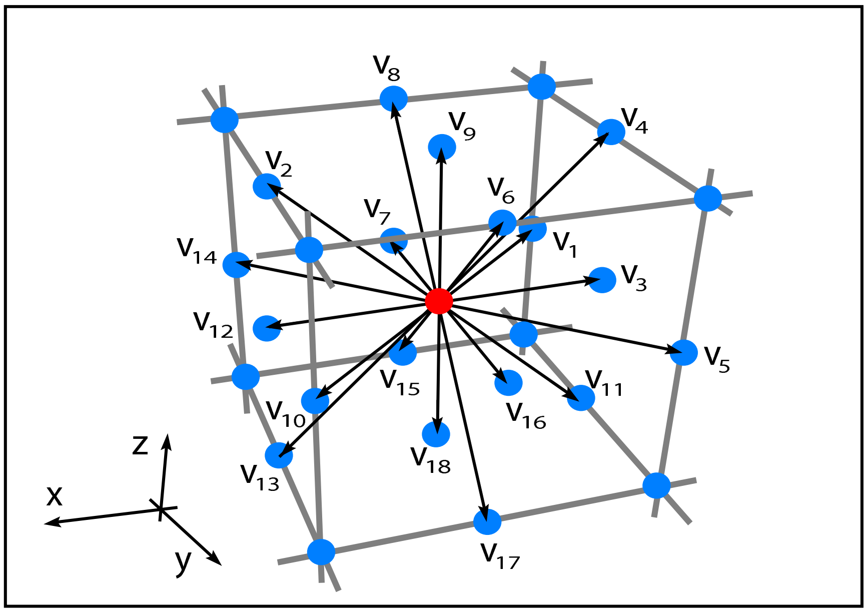


FIGURE 2.2. In the $D3Q19$ model the phase space $\Omega \times \mathbb{R}^3$ is discretised by a uniform grid and 19 velocities v_i, $i = 0, 1, ..., 18$. The 27 red and blue dots symbolise knots of the lattice, while the arrows stand for the discrete velocities. Those are chosen such that a fictive particle which is at the red dot at a fixed time $t = t_0$ will still be at its position or at one of the other knots at $t = t_0 + h^2$.

2.1.2. Velocity Discrete BGK-Boltzmann Equations. Starting point for the derivation to come is the family of BGK-Boltzmann equations $\mathcal{F}$ which is defined according to (2.4). The aim of this subsection is to derive the family of velocity discrete BGK-Boltzmann equations $\mathcal{FG}$ consistently from $\mathcal{F}$. Thereby, the velocity space $\mathbb{R}^d$ is replaced by Q.

In the following, let $h \in \mathbb{R}_{>0}$ be fixed. Due to the definition of $\boldsymbol{v_i}$, $\widetilde{\boldsymbol{v}}_{\boldsymbol{i}} := \boldsymbol{v_i} h$ does not depend on h. Further, it is assumed that there exists a solution f^h of

(2.3) with moments n_{f^h} and $\boldsymbol{u_{f^h}}$. Since f^h depends on h, n_{f^h} and $\boldsymbol{u_{f^h}}$ will also depend on h. Under the condition that n_{f^h}, $\boldsymbol{u_{f^h}} \in \mathcal{O}(1)$, a factor of $M^{eq}_{f^h}$ can be expanded as a Taylor series in h such that the following holds in $I \times \Omega \times Q$:

$$\begin{aligned}
M^{eq}_{f^h} &= \frac{n_{f^h}\ h^d}{\left(\frac{2}{3}\pi\right)^{d/2}} \exp\left(-\frac{3}{2}\widetilde{\boldsymbol{v}}_{\boldsymbol{i}}{}^2\right) \exp\left(3h\ \widetilde{\boldsymbol{v}}_{\boldsymbol{i}} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2}h^2 \boldsymbol{u}^2_{\boldsymbol{f^h}}\right) \\
&= \frac{n_{f^h}\ h^d}{\left(\frac{2}{3}\pi\right)^{d/2}} \exp\left(-\frac{3}{2}\widetilde{\boldsymbol{v}}_{\boldsymbol{i}}{}^2\right) \left(1 + 3h\ \widetilde{\boldsymbol{v}}_{\boldsymbol{i}} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2}h^2 \boldsymbol{u}^2_{\boldsymbol{f^h}} + \frac{9}{2}h^2 \left(\widetilde{\boldsymbol{v}}_{\boldsymbol{i}} \cdot \boldsymbol{u_{f^h}}\right)^2\right) \\
&\quad + R_{1,t,\boldsymbol{r},\boldsymbol{v}}(h) \\
&=: \widetilde{M}^{eq}_{f^h} + R_{1,t,\boldsymbol{r},\boldsymbol{v}}(h)\ ,
\end{aligned}$$

where $R_{1,t,\boldsymbol{r},\boldsymbol{v}} \in \mathcal{O}(h^{3+d})$ is the remainder term for each $(t, \boldsymbol{r}, \boldsymbol{v_i}) \in I \times \Omega \times Q$.

Some properties of the Maxwellian distribution M^{eq}_f are to be preserved also for the approximated Maxwellian distribution $\widetilde{M}^{eq}_f$. Therefore, weights $w_i \in \mathbb{R}_{>0}$ $(i = 0, 1, ..., q-1)$ are sought such that the following holds

$$\begin{aligned}
n_f &= \sum_{i=0}^{q-1} w_i \widetilde{M}^{eq}_f && \text{in } I \times \Omega\ , \\
n_f \boldsymbol{u_f} &= \sum_{i=0}^{q-1} w_i \boldsymbol{v_i} \widetilde{M}^{eq}_f && \text{in } I \times \Omega\ .
\end{aligned}$$

By means of Gauss-Hermite quadrature, He and Luo [**63**] find the weights w_i and the particular set of discrete velocities $\boldsymbol{v_i}$ e.g. for the $D2Q9$ and $D3Q27$ model. For the $D2Q9$ model the weights are

$$\begin{aligned}
w_0 &= \frac{4}{9}w\ , \\
w_1 = w_3 = w_5 = w_7 &= \frac{1}{9}w\ , \\
w_2 = w_4 = w_6 = w_8 &= \frac{1}{36}w\ ,
\end{aligned}$$

where $w := \frac{2}{3}\pi^{d/2} h^{-d} \exp\left(\frac{3}{2}\widetilde{\boldsymbol{v}}_{\boldsymbol{i}}{}^2\right)$. As Wolf-Gladrow in [**140**] shows for the $D3Q19$ model, the conditions demanded above are fulfilled if the weights are given as follows:

$$\begin{aligned}
w_0 &= \frac{1}{3}w\ , \\
w_3 = w_7 = w_9 = w_{12} = w_{16} = w_{18} &= \frac{1}{18}w\ , \\
w_1 = w_2 = w_4 = w_5 = w_6 = w_8 = w_{10} = w_{11} = w_{13} = w_{14} = w_{15} = w_{17} &= \frac{1}{36}w\ .
\end{aligned}$$

With the weights, the moments of f^h can be approximated in $I \times \Omega$ by

$$\begin{aligned}
\int_{\mathbb{R}^d} f^h d\boldsymbol{v} = n_{f^h} = \sum_{i=0}^{q-1} w_i \widetilde{M}^{eq}_{f^h} &= \sum_{i=0}^{q-1} w_i M^{eq}_{f^h} + R_{2,t,\boldsymbol{r}} \\
&= \sum_{i=0}^{q-1} w_i (f^h + 3\nu h^2 \frac{d}{dt} f^h) + R_{2,t,\boldsymbol{r}} \\
&= \sum_{i=0}^{q-1} w_i f^h + R_{3,t,\boldsymbol{r}} \ , \\
\int_{\mathbb{R}^d} \boldsymbol{v} f^h d\boldsymbol{v} = n_{f^h} \boldsymbol{u}_{\boldsymbol{f^h}} = \sum_{i=0}^{q-1} w_i \boldsymbol{v_i} \widetilde{M}^{eq}_{f^h} &= \sum_{i=0}^{q-1} w_i \boldsymbol{v_i} M^{eq}_{f^h} + R_{4,t,\boldsymbol{r}} \\
&= \sum_{i=0}^{q-1} w_i \boldsymbol{v_i} (f^h + 3\nu h^2 \frac{d}{dt} f^h) + R_{4,t,\boldsymbol{r}} \\
&= \sum_{i=0}^{q-1} (w_i \boldsymbol{v_i} f^h) + R_{5,t,\boldsymbol{r}} \ .
\end{aligned}$$

For all $i = 0, 1, ..., q-1$, w_i is understood as a function of h and it holds that $w_i \in \mathcal{O}(h^{-d})$. Further, as it is shown before, $M^{eq}_{f^h}$ is approximated by $\widetilde{M}^{eq}_{f^h}$ with an error in $\mathcal{O}(h^{3+d})$ for all $(t, \boldsymbol{r}, \boldsymbol{v_i}) \in I \times \Omega \times Q$. Therefore, $R_{2,t,\boldsymbol{r}} \in \mathcal{O}(h^3)$ and $R_{4,t,\boldsymbol{r}} \in \mathcal{O}(h^2)$ for all $(t, \boldsymbol{r}) \in I \times \Omega$. Suppose $w_i \frac{d}{dt} f^h(t, \boldsymbol{r}, \boldsymbol{v_i})$ understood as a function of h is in $\mathcal{O}(1)$ for all $(t, \boldsymbol{r}, \boldsymbol{v_i}) \in I \times \Omega \times Q$. Then, one gets $R_{3,t,\boldsymbol{r}} \in \mathcal{O}(h^2)$ and $R_{5,t,\boldsymbol{r}} \in \mathcal{O}(h)$ for all $(t, \boldsymbol{r}, \boldsymbol{v_i}) \in I \times \Omega \times Q$ and finally

$$n_{f^h} - \sum_{i=0}^{q-1} w_i f^h \in \mathcal{O}(h^2) \qquad \text{in } I \times \Omega \ , \tag{2.5}$$

$$\boldsymbol{u}_{\boldsymbol{f^h}} - \frac{\sum_{i=0}^{q-1} w_i \boldsymbol{v_i} f^h}{\sum_{i=0}^{q-1} w_i f^h} \in \mathcal{O}(h^1) \qquad \text{in } I \times \Omega \ . \tag{2.6}$$

In order to shorten the notation, the following terms are defined for $i = 0, 1, ..., q-1$ in $I \times \Omega$:

$$\begin{aligned}
f_i(t, \boldsymbol{r}) &:= w_i f(t, \boldsymbol{r}, \boldsymbol{v_i}) \ , \\
n_{f_i}(t, \boldsymbol{r}) &:= \sum_{i=0}^{q-1} f_i(t, \boldsymbol{r}) \ , \\
\boldsymbol{u}_{\boldsymbol{f_i}}(t, \boldsymbol{r}) &:= \frac{1}{n}(t, \boldsymbol{r}) \sum_{i=0}^{q-1} v_i f_i(t, \boldsymbol{r}) \ , \\
M^{eq}_{f_i}(t, \boldsymbol{r}) &:= \frac{w_i}{w} n_{f_i} \left(1 + 3h^2 \ \boldsymbol{v_i} \cdot \boldsymbol{u}_{\boldsymbol{f_i}} - \frac{3}{2} h^2 \boldsymbol{u}^2_{f_i} + \frac{9}{2} h^4 \left(\boldsymbol{v_i} \cdot \boldsymbol{u}_{\boldsymbol{f_i}}\right)^2\right) \ .
\end{aligned}$$

Multiplying the BGK-Boltzmann equation (2.3) with all weights w_i and using the definitions above, one gets the *velocity discrete BGK-Boltzmann equation.* Depending on the model type, *DdQp* it is given as a set of q equations where $i = 0, 1, ..., q-1$:

$$\frac{d}{dt} f_i = -\frac{1}{3\nu h^2} \left(f_i - M^{eq}_{f_i}\right) \qquad \text{in } I \times \Omega \ . \tag{2.7}$$

With it, one obtains the *family of velocity discrete BGK-Boltzmann equations $\mathcal{FG}$* according to

$$\mathcal{FG} := \left(\frac{d}{dt} f_i + \frac{1}{3\nu h^2} \left(f_i - M^{eq}_{f_i} \right) = 0 \ , \ f_i \in V(I \times \Omega \times Q) \right)_{h>0} \ .$$

The following theorem summarises the results of this subsection:

Theorem 2.2. *Suppose for any $h \in \mathbb{R}_{>0}$ f^h is a solution of the BGK-Boltzmann equation (2.3) with moments n_{f^h} and $\boldsymbol{u_{f^h}}$ and that for $n_{f^h}, \boldsymbol{u_{f^h}}, w_i \frac{d}{dt} f^h(t, \boldsymbol{r}, \boldsymbol{v_i})$ understood as functions of h the following holds:*

$$n_{f^h} \in \mathcal{O}(1) \qquad \text{in } I \times \Omega \ , \tag{2.8}$$

$$\boldsymbol{u_{f^h}} \in \mathcal{O}(1) \qquad \text{in } I \times \Omega \ , \tag{2.9}$$

$$w_i \frac{d}{dt} f^h \in \mathcal{O}(1) \qquad \text{in } I \times \Omega \times Q \ . \tag{2.10}$$

Then, the family of velocity discrete BGK-Boltzmann equations $\mathcal{FG}$ is consistent of order 2 to the family of BGK-Boltzmann equations $\mathcal{F}$, i.e. the truncation error is

$$h^2 \frac{d}{dt} f_i^h + \frac{1}{3\nu} \left(f_i^h - M^{eq}_{f_i^h} \right) \in \mathcal{O}(h^2) \qquad \text{in } I \times \Omega \times Q \ . \tag{2.11}$$

Proof. The statement follows directly from the derivation presented in this subsection. □

2.1.3. Lattice Boltzmann Equations. The aim of this subsection is to derive the family of lattice Boltzmann equations $\mathcal{G}$ which is consistent to the family of BGK-Boltzmann equations $\mathcal{F}$. In the previous subsection, the velocity space $\mathbb{R}^d$ is replaced by a finite set Q of q velocities $\boldsymbol{v_i}$. This led to the family of velocity discrete BGK-Boltzmann equations $\mathcal{FG}$ which is found consistent to the family of BGK-Boltzmann equations $\mathcal{F}$ with a truncation error of order h^2. Tying up to this derivation, the time I and position space Ω are replaced by the discrete spaces I_h and Ω_h according to the definitions established in Subsection 2.1.1. The differential operator $\frac{d}{dt}$ is approximated by a finite difference which finally leads to a second order truncation error.

In the following, the BGK-Boltzmann equation is considered with vanishing external forces, i.e. $\boldsymbol{F} = \boldsymbol{0}$ in $I \times \Omega$. A possible treatment of the force term in LBM is introduced and discussed in detail by Guo et al. in [**57**]. Additionally, the Lagrange description is used to shorten the notation.

Let f^h be a solution of the BGK-Boltzmann equation (2.3) which is three times differentiable. Suppose that for $\frac{d^3}{dt^3} f_i^h$, understood as a function of h, it holds $\frac{d^3}{dt^3} f_i^h \in \mathcal{O}(1)$ for $i = 0, 1, ..., q-1$. Then, Taylor's theorem can be applied leading to a central difference approximation of $\frac{d}{dt} f_i^h$:

$$h^2 \frac{d}{dt} f_i^h(t + h^2/2) = f_i^h(t + h^2) - f_i^h(t) + R_{6,t}(h) \qquad \text{in } I_h \ , \tag{2.12}$$

with a remainder term $R_{6,t} \in \mathcal{O}(h^6)$ for $t \in I_h$. Further, f_i^h is expanded in a Taylor's series and $\frac{d}{dt} f_i^h$ is approximated by a forward difference:

$$\begin{aligned} f_i^h(t + h^2/2) &= f_i^h(t) + \frac{d}{dt} f_i^h(t) h^2/2 + R_{7,t}(h) && \text{in } I_h \\ &= f_i^h(t) + \frac{1}{2} \left(f_i^h(t + h^2) - f_i^h(t) \right) + R_{8,t}(h) && \text{in } I_h \ . \end{aligned} \tag{2.13}$$

Providing $\frac{d^2}{dt^2} f_i^h$ understood as a function of h is in $\mathcal{O}(1)$, then, for the remainder terms it holds $R_{7,t}, R_{8,t} \in \mathcal{O}(h^4)$ in I_h.

Applying Taylor's expansion technique to the velocity discrete Maxwellian $M^{eq}_{f_i}$ leads to the following approximation:

$$M^{eq}_{f^h_i}(t+h^2/2) = M^{eq}_{f^h_i}(t) + R_{9,t}(h) \qquad \text{in } I_h \ . \tag{2.14}$$

Assuming $f^h_i, \frac{d^2}{dt^2} f^h_i \in \mathcal{O}(1)$, then, it follows that the derivative of $M^{eq}_{f^h_i}$ understood as a function of h is in $\mathcal{O}(1)$ as well and with Taylor's theorem the remainder term $R_{9,t}$ is in $\mathcal{O}(h^2)$ in I_h.

Based on the derivations presented in this and the previous subsection the *lattice Boltzmann equation* is formulated as follows

$$f_i(t+h^2) - f_i(t) = -\frac{1}{3\nu+1/2}\left(f_i(t) - M^{eq}_{f_i}(t)\right) \qquad \text{for } t \in I_h,\ i = 0,1,...q-1 \ . \tag{2.15}$$

The *family of lattice Boltzmann equations* $\mathcal{G}$ reads

$$\mathcal{G} := \left(f_i(t+h^2) - f_i(t) + \frac{1}{3\nu+1/2}\left(f_i(t) - M^{eq}_{f_i}(t)\right) = 0 \ , \ f_i \in V(I_h \times Q)\right)_{h>0} \ .$$

Summing up the results of this subsection and of Theorem 2.2, its consistency to the family of BGK-Boltzmann equations is proved in the following theorem:

Theorem 2.3. *Suppose for every $h \in \mathbb{R}_{>0}$ f^h is a solution of the BGK-Boltzmann equation (2.3) with moments n_{f^h} and $\boldsymbol{u}_{\boldsymbol{f}^h}$ and that for n_{f^h}, $\boldsymbol{u}_{\boldsymbol{f}^h}$, f^h_i, $\frac{d}{dt}f^h_i$, $\frac{d^2}{dt^2}f^h_i$, $\frac{d^3}{dt^3}f^h_i$ understood as functions of h the following holds:*

$$n_{f^h} \in \mathcal{O}(1) \qquad \text{in } I_h \times \Omega_h \ , \tag{2.16}$$

$$\boldsymbol{u}_{\boldsymbol{f}^h} \in \mathcal{O}(1) \qquad \text{in } I_h \times \Omega_h \ , \tag{2.17}$$

$$f^h_i, \frac{d}{dt}f^h_i, \frac{d^2}{dt^2}f^h_i, \frac{d^3}{dt^3}f^h_i \in \mathcal{O}(1) \qquad \text{in } I_h \times \Omega_h \text{ for } i = 0,1,...q-1 \ . \tag{2.18}$$

Then, the family of LB equations $\mathcal{G}$ is consistent of order 2 to the family of BGK-Boltzmann equations $\mathcal{F}$ in $I_h \times \Omega_h \times Q$.

Proof. Let f^h be a solution of the BGK-Boltzmann equation (2.3) satisfying all conditions requested for this theorem. Then, Theorem 2.2 holds which leads to a truncation error of

$$\frac{6\nu}{6\nu+1}\left(h^2\frac{d}{dt}f^h_i + \frac{1}{3\nu}\left(f^h_i - M^{eq}_{f^h_i}\right)\right) \in \mathcal{O}(h^2) \qquad \text{in } I \times \Omega \times Q \ .$$

Considering this truncation error at $t+h^2/2$ and substitution according to the finite difference approximations (2.12), (2.13) and (2.14), one finds the error still of the same order:

$$\begin{aligned}
&\frac{6\nu}{6\nu+1}\left(h^2\frac{d}{dt}f^h_i(t+h^2/2) + \frac{1}{3\nu}\left(f^h_i(t+h^2/2) - M^{eq}_{f^h_i}(t+h^2/2)\right)\right)\\
&\approx \frac{6\nu}{6\nu+1}\left(f^h_i(t+h^2) - f^h_i(t) + \frac{1}{3\nu}\left(f^h_i(t) + \frac{1}{2}\left(f^h_i(t+h^2) - f^h_i(t)\right) - M^{eq}_{f^h_i}(t)\right)\right)\\
&= f^h_i(t+h^2) - f^h_i(t) + \frac{1}{3\nu+1/2}\left(f^h_i(t) - M^{eq}_{f^h_i}(t)\right) \in \mathcal{O}(h^2)
\end{aligned}$$

for $i = 0,1,...,q-1$ and for all $t \in I_h$. □

2.1.4. Formulation of Boundary and Initial Conditions. The specification of boundary conditions for LBM is a difficult task. In [**64**] He et al. remark: "[...] the real hydrodynamic boundary conditions have not been fully understood." Especially if one wants to simulate a fluid flow for which the problem configuration is given in macroscopic scales, i.e. conditions at the boundaries or initial conditions are only given for the pressure q and the velocity $\boldsymbol{u}$. Then, one has to provide conversation formulas between the macroscopic quantities and the discrete distribution function $f_i \in Q$. Problems can arise if the number of unknowns for the distribution function exceeds the number of given macroscopic variables. In this case, there is no closed system given to derive conversation formulas. To overcome this difficulty several approaches have been proposed. In many of them the smoothness of higher order moments of f_i is assumed. Then, intra- and extrapolation schemes are employed to close the gap. In many cases however, their physical meaning is not clear. In [**128**] a survey of various boundary conditions for LBM is given. A short extract of frequently used boundary and initial conditions is given in the following.

An exception regarding the difficulties is the treatment of *periodic* and *symmetry boundary conditions* since setting the discrete distribution function f_i such that the periodicity and symmetry, respectively, is ensured automatically leads to periodic and symmetric macroscopic quantities at the boundaries.

For solid walls three kinds of boundary conditions can be distinguished: *no-slip*, *free-slip* and *frictional-slip*. A common macroscopic property is that $\frac{\partial}{\partial \boldsymbol{n}}\boldsymbol{u} = \boldsymbol{0}$. This is realised by different kinds of reflection rules whereby the values of f_i with outward pointing $\boldsymbol{v_i} \in Q^{out}$ are used to determine the values of f_j for inward pointing velocities $\boldsymbol{v_j} \in Q^{in}$.

- The *no-slip* variant approximates a wall which is sufficiently rough to prevent any macroscopic fluid motion, i.e. $\boldsymbol{u} = \boldsymbol{0}$. It is often called *bounce-back boundary condition.* For LBM it is realised by the setting $f_j = f_i$ for $v_i = -v_j$ for all $v_j \in Q^{in}$.
- The *free-slip* boundary conditions ensures $f_j = f_i$ for $v_i \cdot v_j = 0$ for all $v_j \in Q^{in}$ which applies to the case of a smooth surface of the wall with negligible friction exerted upon the fluid.
- The *frictional-slip* boundary condition is a blend of the former mentioned two extremes. It can be realised e.g. by constituting a rule which determines all $f_j \in Q^{in}$ as linear combinations of $f_i \in Q^{out}$.

Initial conditions and more general boundary conditions such as inflow and outflow conditions are more involved as mentioned at the beginning of this subsection and are still an active topic of research. A systematic discussion of using finite difference schemes to formulate boundary and initial conditions for LBM is recorded by Skordos in 1993 [**124**]. A distribution $f_i \in Q$ which is to be set is split into an equilibrium and an off-equilibrium part. With the help of a Chapman-Enskog expansion, the off-equilibrium parts are associated with the velocity gradients. They are kept smooth by considering the gradients of neighbouring nodes. Later, similar approaches are proposed by Inamuro et al. [**73**], He and Zou [**64**] and Latt [**88, 89**]. All four approaches are often used in practice. In [**90**] Latt et al. compare the five approaches of the before mentioned authors intending to adopt Dirichlet boundary conditions at a macroscopic level. All five methods are analysed analytically using the Chapman-Enskop expansion. Various numerical tests are done through benchmarks of two-dimensional and three-dimensional flows revealing non of the methods as superior regarding accuracy and stability.

2.2. Implementation of Lattice Boltzmann Algorithms

This section is devoted to present a generic and efficient implementation strategy dedicated for LBM. Aiming to obtain a straightforward, intuitive and easily extensible implementation with almost no loss of efficiency, it is proposed to take advantage of both the dynamic and static genericity offered by an object-oriented programming language like C++. This allows furthermore to develop a single code that can serve for many purposes. The author have already published parts of the presented work in [**66**].

However, implementation details and performance issues related to LBM have been discussed before in the literature, e.g. in [**45, 84, 87, 128**]; and it has also previously been suggested to use object-oriented techniques for the implementation of LB algorithms [**32**]. In [**68**] Heuveline and Latt propose a strategy for a generic implementation in the framework of the OpenLB project (Appendix A). The there presented ideas are taken up and enhanced towards an advanced realisation of LBM allowing in particular efficient parallelism especially for fluid flow problems with underlying complex geometries.

In the first part of this section, a concept for a data structure design dedicated for LBM is presented. Based on it, the second part addresses the realsation of typical LB algorithms emerging from LB equations like (2.15). The realisation of the proposed concepts are illustrated by means of an example, namely the extension of the open source software OpenLB.

2.2.1. Data Structure Design. An LBM, in its most widely accepted formulation, is executed on a regular, homogeneous lattice Ω_h with equal grid spacing $h \in \mathbb{R}_{>0}$ in all directions. When numerical constraints require that a given problem is solved on an inhomogeneous grid, it is common to adopt a so-called *multi-block* approach: the computational domain is partitioned into sub-grids with different levels of resolution, and the interface between those sub-grids is handled appropriately. This approach appears to respect the spirit of LBM well and leads to implementations that are both elegant and efficient since the execution on a set of regular blocks is relatively fast compared to an unstructured grid representation of the whole geometry. For complex domains a multi-block approach provides another advantage. A given domain can be represented by a certain number of regular blocks which leads to cheap executions times on the one hand and to a sparse memory consumption on the other hand. Furthermore, it encourages a particularly efficient form of *data parallelism*, in which an array is cut into regular pieces and distributed over the nodes of a parallel machine. As a result, LB applications can be run even on large parallel machines with a particularly satisfying gain of speed.

The same spirit is adopted in the OpenLB package, in which the basic datastructure is a `BlockLattice` which represents a regular array of `Cells`. In each `Cell` the q variables for the storage of the discrete velocity distribution functions f_i $(i = 0, 1, ..., q - 1)$ are contiguous in memory (cf. collision optimised data layout in [**139**]). As remarked in the next subsection, required memory is allocated only once since no temporary memory is needed in the applied algorithm. This data structure is encapsulated by a higher level, object-oriented layer. The purpose of this layer is to handle groups of `BlockLattice`, and to build higher level software constructs in a relatively transparent way. Those constructs are called `SuperLattices` and include multi-block, grid refined lattices as well as parallel lattices.

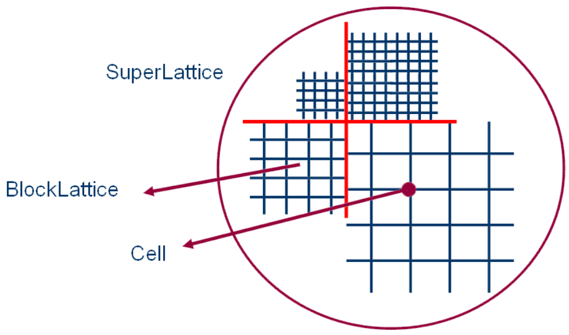


FIGURE 2.3. Data structure in OpenLB: A number of `BlockLattices` build a `SuperLattice` to adopt higher level software constructs like multi-block, grid refined lattices and parallelised lattices

2.2.2. Organisation of the Code. The core of OpenLB consists of a simple and efficient array-like construct called a `BlockLattice`. This object executes an LB algorithm in a very traditional sense as indicated in Algorithm 1, i.e. the lattice Boltzmann equation (2.15) is split in two equation, namely the *collision step*

$$\widetilde{f}_i^h(t,\boldsymbol{r}) = f_i^h(t,\boldsymbol{r}) - \frac{1}{3\nu + 1/2}\Big(f_i^h(t,\boldsymbol{r}) - M^{eq}_{f_i^h}(t,\boldsymbol{r})\Big) \qquad \text{in } I_h \times \Omega_h \times Q \tag{2.19}$$

and the *streaming step (propagation step)*

$$f_i^h(t+h^2, \boldsymbol{r} + h^2\boldsymbol{v_i}) = \widetilde{f}_i^h(t,\boldsymbol{r}) \qquad \text{in } I_h \times \Omega_h \times Q\ . \tag{2.20}$$

All `Cells` of the `BlockLattice` are iteratively parsed, and a local collision step is executed, followed by a non-local streaming step. The streaming step is independent of the choice of lattice Boltzmann dynamics and remains invariant. On the other hand, the collision step determines the physics of the model and can be configured by the user, by attributing a fully configurable dynamics object to each `Cell`. In this way, it is easy to implement inhomogeneous fluids which use a different type of physics from one `Cell` to another. For each time step the collision and streaming step can be executed separately in two loops over all `Cells` or only in one. Both versions are implemented in OpenLB. Yet, for many applications the method fusing the two loops is preferable. With a dedicated swapping technique [**85, 97, 98**] the locality of the stored data is preserved and a costly reloading of data into the cache in a second loop can be avoided.

Algorithm 1 A basic lattice Boltzmann algorithm.

```
1. Reading input
2. Simulation setup
3. Time loop
for $t = t_0$ to $t = t_{max}$ do
  a) Collision
  b) Streaming
  c) Post-processing
  d) Writing output
```

Although this concept of a `BlockLattice` is neat and should please the programmer by being conceptually close to the theory of LBM, it is not sufficiently general to address all possible issues arising in real life. As a case in point, some boundary conditions are non-local and need to access neighbouring nodes. Therefore, their implementation does not fit into the framework of a `BlockLattice` explained previously. The philosophy of OpenLB takes for granted that such situations, although they arise, take place in spatially confined areas only, as for example the domain boundaries. They may therefore be implemented by slightly less efficient means, without spoiling the overall efficiency of the code. Their execution is taken care of by a post-processing step (see Algorithm 1), which, instead of stepping over the whole lattice a second time, parses selected `Cells` only.

Another task that falls under the responsibility of the post-processing step is the interconnection of various `BlockLattice` structures during the implementation of higher level constructs which are described in the previous subsection. In this way, the efficiency of a basic LBM is combined with an access to advanced constructs, including parallel and grid refined codes.

2.3. Numerical Experiment: A 3D Stationary Fluid Flow Problem

In Chapter 1 the connection between a macroscopic and a mesoscopic model both describing incompressible Newtonian fluids is studied. In particular in Subsection 1.3.5 it turned out that a family of BGK-Boltzmann equations is related to the incompressible Navier-Stokes equation in a certain regime of small Mach and Knudsen numbers. As illustrated in this chapter, the discretisation strategy for BGK-Boltzmann equations which leads to LB equations is based on coupling the discretisation parameter with the a term which characterises this regime. The aim of this section is to provide numerical evidence for both, the discretisation approach and the relation of the marcoscopic and mesoscopic model. Beside, the study contributes to validate the realisation of the presented implementation strategy.

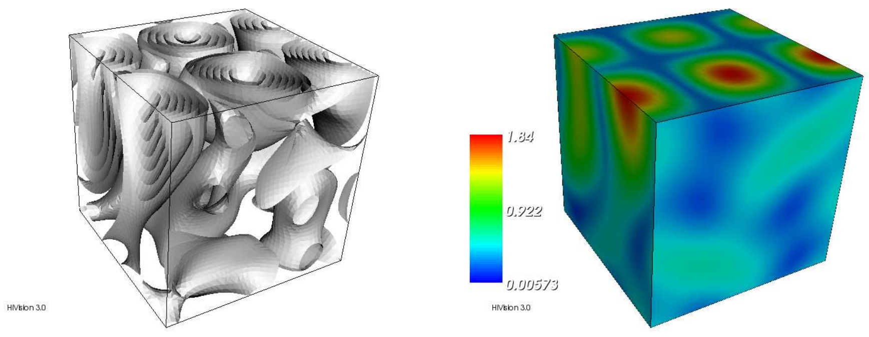


FIGURE 2.4. Velocity distribution $\boldsymbol{u}^*$ in Ω of the analytical solution of the 3D benchmark problem with a Reynolds number of $Re = 10$. The left picture shows isosurfaces of $\|\boldsymbol{u}^*\|_2$ in Ω and the right one the distribution of $\|\boldsymbol{u}^*\|_2$ on the surface of Ω

.

Therefore, in the following the asymptotic behaviour of discrete solutions of LB equations of a particular family is examined. As starting point serves the analytic solution of a stationary 3D fluid flow problem which is formulated by means of the macroscopic model which is governed by the corresponding Navier-Stokes equation. Based on that, here the same fluid flow problem is formulated by means of the respective family of BGK-Boltzmann equations. Then, in Subsection 2.3.2 the LBM of choice is presented and further details e.g. regarding a suitable stop criteria, are addressed. Finally, the obtained results are presented and discussed.

2.3.1. Test Case Description. The subject of interest is the stationary flow of an incompressible Newtonian fluid in a cube. The fluid is characterised by its density $\rho = 1$ and kinetic viscosity $\nu = 0.1$. Defining the characteristic velocity $U = 1$ and the characteristic length $L = 1$ leads to a Reynolds number $Re = 10$. The domain is represented by the unit cube $\Omega = (0,1)^3 \subseteq \mathbb{R}^3$. In the whole domain an external force $\boldsymbol{F}$ is applied to the fluid according to

$$\begin{aligned}
F_1(\boldsymbol{r}) = -\frac{1}{8}\pi\,(&16\pi\nu\,(\cos(2\pi r_2)\cos(2\pi r_1) - \sin(2\pi r_1)\cos(2\pi r_3))\\
&- \cos(2\pi r_2)\cos(2\pi r_3) + 2\cos(2\pi r_2)\cos(2\pi r_1)^2\cos(2\pi r_3)\\
&+ \cos(2\pi r_2)^2\cos(2\pi r_1)\sin(2\pi r_1) - \cos(2\pi r_1)\sin(2\pi r_3)\\
&+ \cos(2\pi r_1)\sin(2\pi r_3)\cos(2\pi r_2)^2 - \sin(2\pi r_2)\cos(2\pi r_1)^2\\
&- \sin(2\pi r_1)\cos(2\pi r_1) - 2\pi r_3\sin(2\pi r_3)\cos(2\pi r_2)\\
&+ 2\pi r_3\sin(2\pi r_3)\cos(2\pi r_2)\cos(2\pi r_1)^2\\
&+ \sin(2\pi r_1)\sin(2\pi r_3)\cos(2\pi r_2)\cos(2\pi r_3) + 16 r_3\sin(2\pi r_1)\sin(2\pi r_2))\\
F_2(\boldsymbol{r}) = -\frac{1}{8}\pi\,(&-8\pi\nu\,(\cos(2\pi r_1) + 2\sin(2\pi r_2)\sin(2\pi r_3)) + \cos(2\pi r_3)\\
&- \cos(2\pi r_3)\cos(2\pi r_1)^2 - \sin(2\pi r_1)\cos(2\pi r_2)\cos(2\pi r_1)\\
&- \cos(2\pi r_2)\sin(2\pi r_2) - \cos(2\pi r_2)\sin(2\pi r_3)\cos(2\pi r_1)\\
&+ \sin(2\pi r_2)\cos(2\pi r_3)\cos(2\pi r_1)\sin(2\pi r_3)\\
&+ 2\pi r_3\sin(2\pi r_2)\cos(2\pi r_3)\cos(2\pi r_2)\sin(2\pi r_1)\\
&- 16 r_3\cos(2\pi r_2)\cos(2\pi r_1))\\
F_3(\boldsymbol{r}) = -\frac{1}{8}\big(&16\pi^2\nu\,(\cos(2\pi r_1)\sin(2\pi r_3) + 2\pi r_3\cos(2\pi r_2)\sin(2\pi r_1)\\
&- \cos(2\pi r_2)\cos(2\pi r_3)) - 2\pi^2 r_3\sin(2\pi r_3)\sin(2\pi r_1)\\
&- 2\pi^2 r_3\sin(2\pi r_2)\cos(2\pi r_1)\sin(2\pi r_1)\\
&+ \pi\sin(2\pi r_2)\cos(2\pi r_1)\cos(2\pi r_3) - \pi\cos(2\pi r_2)\cos(2\pi r_1)\\
&+ 2\pi\cos(2\pi r_2)(\cos(2\pi r_3))^2\cos(2\pi r_1) - 2\pi^2 r_3(\cos(2\pi r_2))^2\\
&+ \pi(\cos(2\pi r_2))^2\cos(2\pi r_3)\sin(2\pi r_1) - 8\cos(2\pi r_1)\sin(2\pi r_2)\big)
\end{aligned} \tag{2.21}$$

Further, the velocity distribution is given at the boundary of Ω:

$$\begin{aligned}
u_1(\boldsymbol{r}) &= \frac{1}{4}\left(\sin(2\pi r_1)\cos(2\pi r_3) - \cos(2\pi r_1)\cos(2\pi r_2)\right)\\
u_2(\boldsymbol{r}) &= \frac{1}{4}\left(\sin(2\pi r_2)\sin(2\pi r_3) + \cos(2\pi r_1)\right)\\
u_3(\boldsymbol{r}) &= -\frac{1}{4}\left(\cos(2\pi r_1)\sin(2\pi r_3) + 2\pi r_3\sin(2\pi r_1)\cos(2\pi r_2)\right.\\
&\quad \left. - \cos(2\pi r_2)\cos(2\pi r_3)\right)
\end{aligned} \tag{2.22}$$

As pointed out in Chapter 1 for such configurations the Navier-Stokes equation (1.34) is the governing equation for the macroscopic model. For this problem $\frac{\partial}{\partial t}\boldsymbol{u} = \boldsymbol{0}$ in Ω since $\boldsymbol{u}$ does not depend on t which leads to the so-called stationary Navier-Stokes equation.

By defining $\boldsymbol{u}^*$ and p^* in Ω as

$$\begin{aligned} u_1^*(\boldsymbol{r}) &= \frac{1}{4}\left(\sin\left(2\pi r_1\right)\cos\left(2\pi r_3\right) - \cos\left(2\pi r_1\right)\cos\left(2\pi r_2\right)\right) \\ u_2^*(\boldsymbol{r}) &= \frac{1}{4}\left(\sin\left(2\pi r_2\right)\sin\left(2\pi r_3\right) + \cos\left(2\pi r_1\right)\right) \\ u_3^*(\boldsymbol{r}) &= -\frac{1}{4}\left(\cos\left(2\pi r_1\right)\sin\left(2\pi r_3\right) + 2\pi r_3 \sin\left(2\pi r_1\right)\cos\left(2\pi r_2\right)\right. \\ &\qquad \left. - \cos\left(2\pi r_2\right)\cos\left(2\pi r_3\right)\right) \\ p^*(\boldsymbol{r}) &= \cos\left(2\pi r_1\right)\sin\left(2\pi r_2\right) r_3 \end{aligned} \tag{2.23}$$

and inserting them into (1.34) one can easily verify that $(\boldsymbol{u}^*, p^*)$ is a solution of the above described problem. The problem setup and the distribution of its solution $\boldsymbol{u}^*$ is illustrated in Figure 2.4.

2.3.2. Discretisation Issues. The stationary problem is solved by considering a suitable instationary problem and solving it numerically by means of an LBM. Therefore, it must be provided that the solution of the instationary problem converges in time towards the solution of the original stationary problem.

Let h denote the discretisation parameter. With it, the considered BGK-Boltzmann equation (2.3) is also specified. Then, the unit cube Ω is disretised by a lattice Ω_h of size $(N+1) \times (N+1) \times (N+1)$ where $N := 1/h$. The $D3Q19$ model is chosen for the numerical experiments. The force term is treated as proposed by Guo et al. [**57**].

Due to the fact that the considered configuration is a closed vessel, the pressure distribution is defined up to a constant. Therefore, subtracting a constant offset from the pressure does not modify the results of the incompressible Navier-Stokes equation. It is observed however, that the LBM experiences numerical instabilities when the pressure, and thus the particle density, is too high. For this reason, the average density in the system is computed at every iteration step of the simulation. At the next iteration, a constant offset is subtracted from the distribution function f_i on every grid node, in order to keep the average density close to a value of 1.

The boundary condition is based on the technique suggested by Skordos in [**124**]. With this type of boundary conditions, velocity gradients on boundary nodes are evaluated using a second-order accurate finite difference scheme. This information is then used to compute the off-equilibrium terms of the particle distribution functions. It is also worth to note that on corner and edge nodes, the value of the pressure is extrapolated from bulk nodes, as the data locally available on the nodes is insufficient to evaluate the exact value of the pressure. The initial condition is set using the Maxwellian distribution with $\boldsymbol{u} = \boldsymbol{0}$ and $\rho = 1$.

The convergence is checked by monitoring the evolution of the *average kinetic energy* $\overline{E_{kin}}$ which is defined as

$$\overline{E_{kin}}(t) := \frac{1}{(N+1)^3} \sum_{\boldsymbol{r} \in \Omega_h} E_{kin}(t, \boldsymbol{r}) \qquad \text{in } I_h$$

where the *kinetic energy* is

$$E_{kin}(t, \boldsymbol{r}) := \frac{1}{2} \boldsymbol{u}_{\boldsymbol{f_i}}(t, \boldsymbol{r}) \cdot \boldsymbol{u}_{\boldsymbol{f_i}}(t, \boldsymbol{r}) \qquad \text{in } I_h \times \Omega_h \ .$$

During the execution of the explicit LB scheme the average kinetic energy $\overline{E_{kin}}$ is recorded over a certain macroscopic time scale $J_h \subseteq I_h$ which represents the latest calculated but not more than $k := 2h^2$ time steps. The recorded average kinetic energies $\overline{E_{kin}}(t)$, $t \in J_h$ are considered as a series of samples. Convergence is claimed if the sample standard deviation of the series is smaller than $\epsilon := 10^{-10}$ times the average of the sample, that is

$$\sqrt{\frac{1}{|J_h|-1}\sum_{t\in J_h}\left(\overline{E_{kin}}(t) - \sum_{t\in J_h}\overline{E_{kin}}(t)\right)^2} < \epsilon\frac{1}{|J_h|}\sum_{t\in J_h}\overline{E_{kin}}(t) \ .$$

2.3.3. Presentation and Discussion of the Numerical Results. Varying the discretisation parameter h, a series of experiments has been performed. The smallest choice of the parameter is $h = 1/400$ which results in a number of unknown variables of approximately $1.22 \cdot 10^9$. With the usage of IEEE double precision floating point arithmetic this results in a storage space of about 9.76 GB in main memory. To keep the execution times short, a hybrid parallel approach, which is presented in Chapter 3, is applied. All experiments have been performed on the *HP XC4000* supercomputer at the Steinbuch Center for Computing at the Universität Karlsruhe (TH) (cf. Section 3.3 for a detailed description).

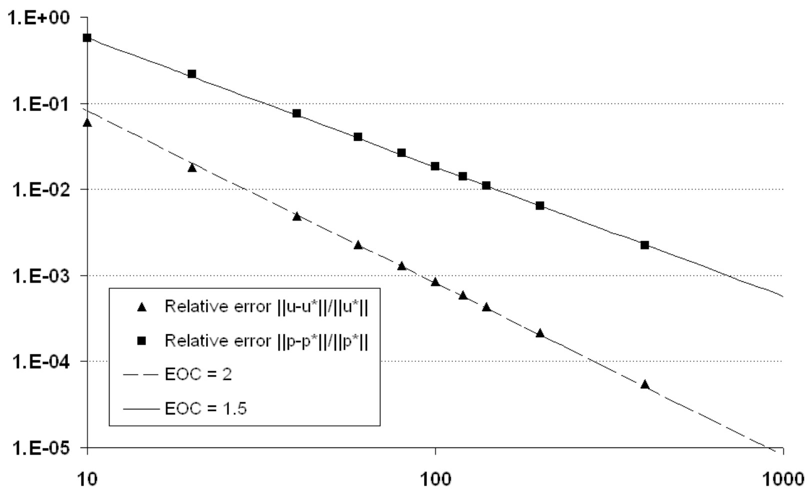


FIGURE 2.5. The development of the relative error for the velocity $\|\boldsymbol{u}_{\boldsymbol{f_i}} - \boldsymbol{u^*}\|_{L^2(\Omega_h)}/\|\boldsymbol{u^*}\|_{L^2(\Omega_h)}$ and pressure $\|p_{f_i} - p^*\|_{L^2(\Omega_h)}/\|p^*\|_{L^2(\Omega_h)}$ is given as a function of the discretisation parameter h. The experimental order of convergence (EOC) for the relative error for the velocity is found to be approximately 2 and that for the pressure 1.5.

For the obtained discrete solution f_i the velocity distribution $\boldsymbol{u}_{\boldsymbol{f_i}}$ and the pressure distribution p_{f_i} are calculated. Then, for several chosen h the relative errors between $\boldsymbol{u}_{\boldsymbol{f_i}}$, p_{f_i} and their counterparts, the analytical solution $\boldsymbol{u^*}$, p^*, are provided according to

$$e_{\boldsymbol{u},h} := \frac{\|\boldsymbol{u}_{\boldsymbol{f_i}} - \boldsymbol{u^*}\|_{L^2(\Omega_h)}}{\|\boldsymbol{u^*}\|_{L^2(\Omega_h)}} \ , \quad e_{p,h} := \frac{\|p_{f_i} - p^*\|_{L^2(\Omega_h)}}{\|p^*\|_{L^2(\Omega_h)}} \ .$$

Both errors are found to be small even for greater discretisation parameters. For $h = 1/40$ e.g. $e_{\boldsymbol{u},1/40} \approx 4.93 \cdot 10^{-3}$ and $e_{p,1/40} \approx 7.57 \cdot 10^{-2}$. Higher accuracy $e_{\boldsymbol{u},1/400} \approx 5.47 \cdot 10^{-5}$ and $e_{p,1/400} \approx 2.22 \cdot 10^{-3}$ is observed for the configuration with $h = 1/400$. Generally it is observed that both errors tend to zero when h

tends to zero. The *experimental order of convergence* (EOC) for $h_1, h_2 \in \mathbb{R}_{>0}$ with $h_1 < h_2$, is defined by

$$EOC_p(h_1, h_2) := \frac{\ln(e_{p,h_1}/e_{p,h_2})}{\ln(h_1/h_2)} , \; EOC_{\boldsymbol{u}}(h_1, h_2) := \frac{\ln(e_{\boldsymbol{u},h_1}/e_{\boldsymbol{u},h_2})}{\ln(h_1/h_2)} .$$

To give an example, in the experiment two typical EOC are $EOC_p(1/200, 1/140) \approx 1.53$ and $EOC_{\boldsymbol{u}}(1/200, 1/140) \approx 1.97$. As Figure 2.5 illustrates, the EOC for the pressure tends to 1.5 and that for the velocity to 2 if h_2 tends to zero.

2.4. Numerical Experiment: A 3D Instationary Fluid Flow Problem

The subject of this section is to study the time evolution of a flow in a cavity driven by a moving lid. Thereby, the main aims are to provide further numerical evidence for the presented mesoscopic approach and to validate its realisation.

In contrast to the problem which is subject of the previous section now the investigated configuration is a more realistic one in the sense that the starting-point is a real observable problem and not one which arises by a given smooth analytical solution. Thus, an additionally error emerging from the choice of the model may be taken into account.

In the beginning of this section, the fluid flow problem configuration is formulated. Afterwards, two approaches to solve the problem numerically are suggested: an FEM and an LBM. Then, the obtained results are benchmarked by comparing them to an experimentally gained solution.

2.4.1. 3D Lid-Driven Cavity Benchmark Problem. The *lid-driven cavity* (LDC) benchmark problem has attracted considerable attention for many years. The reasons are manifold. On the one hand, the geometry of a LDC problem is simple which facilitates experiment set-ups as well as numerical implementations. On the other hand, the flow structure is not simple at all. Especially in the 3D case physical phenomena like boundary layers, eddies of different sizes and various instabilities can appear. An overview of the work done in the last three decades is given in [**27**].

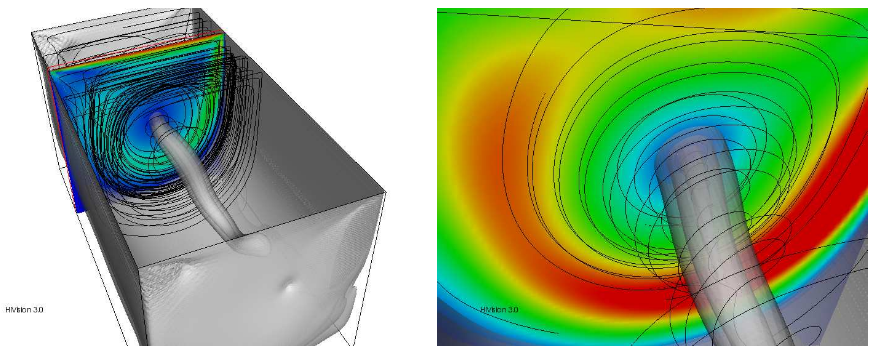


FIGURE 2.6. Lid-driven cavity (cavity ratio 1:1:2): velocity distribution at $t = 12$ for $Re = 1000$. With the help of streamline and isosurfaces the main vortex is accented. The right picture displays a closer look to it and thus reveals its complex structure.

In particular to be mentioned is the work of Ghia et al. [**49**] from 1982 where various configurations of 2D cases are studied. It further provides detailed numerical results. Hence, it serves as benchmark for many other investigations. For the 3D case not many quantitative benchmark results are available. In [**55**] Guermond et al. present experimentally gained results. To compare the numerical results obtained by means of an LBM and an FEM with the experimental data, exactly this simulation set-up is chosen.

An incompressible Newtonian fluid in a rectangular cavity of ratio 1:1:2 is brought into motion by a constantly moving lid. The problem is characterised by the length and speed of the lid, each of them is assumed to be equal to one. In this system of units, the considered duration of the simulation is $\Delta t = 12$ and the Reynolds number is $Re = 1000$. At the beginning the fluid is at rest, i.e. the velocity distribution is given as $\boldsymbol{u} = \mathbf{0}$ in the cavity. Then, the lid abruptly starts to move with the constant unit speed and as a result the fluid starts developing several vortices with one dominant right in the center of the cuboid. The pictures in Figure 2.6 illustrate the described set-up.

Based on this problem specification the fluid flow problem can be described by a macroscopic and a mesoscopic model. Following the explanation and statements of Chapter 1 the governing equations are the incompressible Navier-Stokes equation and the Boltzmann or BGK-Boltzmann equation, respectively. For the macroscopic approach the problem is solved numerically by means of an FEM while for the mesoscopic case an LBM is applied.

2.4.2. Macroscopic Approach with a Finite Element Method. The incompressible Navier-Stokes equation (1.34) is the considered macroscopic governing equation for the LDC problem. The computation domain is defined as $\Omega = \{\boldsymbol{r} \in \mathbb{R}^3 : -0.5 < r_1, r_2, 2r_3 < 0.5\}$ as illustrated in Figure 2.7 and the considered time interval is $I = [0,\ 12] \subseteq \mathbb{R}$.

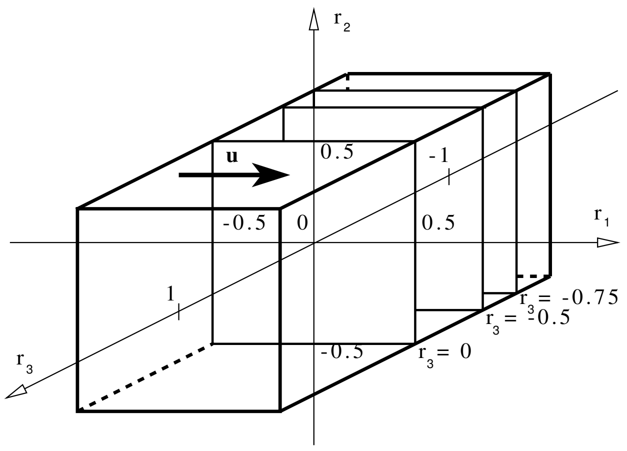


FIGURE 2.7. Simulation setup of the considered 3D lid-driven cavity problem. The upper r_1-r_3-plane ($r_2 = 0.5$) is moving with a constant velocity $\boldsymbol{u}$ while at all other boundary planes of Ω the velocity is set to $\boldsymbol{u} = \mathbf{0}$. Three r_1-r_2-planes at $r_3 = -0.75, r_3 = 0.5$ and $r_3 = 0$ are chosen to compare the results.

The external force is set to $\boldsymbol{F} = \boldsymbol{0}$, the density to $\rho = 1$, the viscosity to $\nu = 10^{-3}$. The initial and boundary conditions are given according to

$$\begin{aligned} &\boldsymbol{u}(0,\boldsymbol{r}) = \boldsymbol{0} && \text{in } \Omega \ , \\ &\boldsymbol{u}(t,\boldsymbol{r}) = \boldsymbol{0} && \text{in } I \times \Gamma \cap \big\{\boldsymbol{r} \in \mathbb{R}^3 : r_2 \neq 0.5\big\} \ , \\ &\boldsymbol{u}(t,\boldsymbol{r}) = (1,0,0)^T && \text{in } I \times \Gamma \cap \big\{\boldsymbol{r} \in \mathbb{R}^3 : r_2 = 0.5\big\} \ . \end{aligned}$$

The time interval I is discretised by $I_d := \big\{t \in I : t = \frac{1}{10}k, k \in \mathbb{Z}\big\}$ and an implicit Euler scheme is applied that leads for each k, $k = 1, 2, ..., 120$, to a system of nonlinear equations:

$$\begin{aligned} 10\left(\boldsymbol{u}^k - \boldsymbol{u}^{k-1}\right) + \left(\boldsymbol{u}^k \cdot \boldsymbol{\nabla_r}\right)\boldsymbol{u}^k &= -\boldsymbol{\nabla_r} p^k + 10^{-3}\boldsymbol{\Delta_r}\boldsymbol{u}^k && \text{in } \Omega \\ \boldsymbol{\nabla_r} \cdot \boldsymbol{u}^k &= 0 && \text{in } \Omega \ , \end{aligned}$$

where $\boldsymbol{u}^k(\boldsymbol{r}) := \boldsymbol{u}(k/10,\boldsymbol{r})$ and $p^k(\boldsymbol{r}) := p(k/10,\boldsymbol{r})$ for $\boldsymbol{r} \in \Omega$.

The position space is discretised by a uniform grid Ω_d which consists of $64 \times 64 \times 128$ cubes of size $(1/64)^3$. Then, a Galerkin FEM is applied with *Taylor-Hood elements* $Q2Q1$. This leads to a nonlinear system with n, ($n \approx 13.4$ million), degrees of freedom for each of the 120 time steps:

$$\boldsymbol{G}^k\left(\begin{pmatrix} \boldsymbol{u}_d^k \\ \boldsymbol{p}_d^k \end{pmatrix}\right) = \boldsymbol{0} \qquad\qquad \text{for } k = 1, 2, ..., 120 \ ,$$

where $(\boldsymbol{u}_d^k, \boldsymbol{p}_d^k)^T \in \mathbb{R}^n$ is the discrete solution vector and

$$\boldsymbol{G^k} : \begin{cases} \mathbb{R}^n & \to \mathbb{R}^n \\ \begin{pmatrix} \boldsymbol{u}_d^k \\ \boldsymbol{p}_d^k \end{pmatrix} & \mapsto \begin{pmatrix} \boldsymbol{A}\boldsymbol{u}_d^k + \boldsymbol{N}(\boldsymbol{u}_d^k) + \boldsymbol{B}^T\boldsymbol{p}_d^k \\ \boldsymbol{B}\boldsymbol{u}_d^k \end{pmatrix} \end{cases}$$

with a nonlinear discrete operator $\boldsymbol{N}$ and matrices $\boldsymbol{A}, \boldsymbol{B}$.

Newton's method is applied in order to linearise these systems. This leads to linear systems which are to be solved until a given tolerance *tol* $:= 10^{-12}$ is reached. Providing start solutions according to

$$(\boldsymbol{u}_{d,0}^1, \boldsymbol{p}_{d,0}^1) := (\boldsymbol{0}, \boldsymbol{0}) \ , \ (\boldsymbol{u}_{d,0}^k, \boldsymbol{p}_{d,0}^k) := (\boldsymbol{u}_d^{k-1}, \boldsymbol{p}_d^{k-1}) \text{ for } k = 2, ..., 120 \ ,$$

one gets for each time step k, $k = 1, ..., 12$, and each Newton step l, $l = 1, 2, ...$, a linear system which is to be solved:

$$\boldsymbol{\nabla} \otimes \boldsymbol{G}_k\left(\begin{pmatrix} \boldsymbol{u}_{d,l-1}^k \\ \boldsymbol{p}_{d,l-1}^k \end{pmatrix}\right)\left(\begin{pmatrix} \boldsymbol{u}_{d,l}^k \\ \boldsymbol{p}_{d,l}^k \end{pmatrix} - \begin{pmatrix} \boldsymbol{u}_{d,l-1}^k \\ \boldsymbol{p}_{d,l-1}^k \end{pmatrix}\right) = -\boldsymbol{G}_k\left(\begin{pmatrix} \boldsymbol{u}_{d,l-1}^k \\ \boldsymbol{p}_{d,l-1}^k \end{pmatrix}\right) \ .$$

Each system is solved with a *generalised minimal residual method* (GMRES) which is an iterative scheme to solve linear systems by means of minimising the norm of the residuum in a specified *Krylov subspaces* (cf. e.g. [**116**]). The tolerance is given similarly to that of the Newton scheme, *tol* $= 10^{-12}$.

A detailed overview of the methods and techniques applied to solve the incompressible Navier-Stokes equation numerically which has been mentioned within this subsection can be found e.g. in [**107**].

2.4.3. Mesoscopic Approach with a Lattice Boltzmann Method. The LDC problem is now formulated mathematically within the mesoscopic model by means of the BGK-Boltzmann equation (2.3) with $h := 1/200$ serving as the governing equation. With it $\delta t = h^2 = 1/40,000$ which yields $480,000$ iteration steps since in the system of units where the LDC problem is formulated the considered time interval is $I = [0,\ 3]$. To reduce the number of time steps needed, in the following the LDC problem is considered after a certain scaling. The results are finally rescaled to the original system of units.

The speed of the moving lid is set to be 4 while the length remains to be 1. With this characteristic quantities the viscosity must be $\widetilde{\nu} = 4 \cdot 10^{-3}$ and the time $\widetilde{I} = [0,\ 3]$ to achieve the same Reynolds number $Re = 1000$ and so describe the same problem. The computation domain is $\Omega = \{\boldsymbol{r} \in \mathbb{R}^3 : -0.5 < r_1, r_2, 2r_3 < 0.5\}$. It is illustrated in Figure 2.7. The $D3Q19$ model is chosen which amounts to a total of about 308 million unknown variables for the lattice of size $201 \times 201 \times 401$.

The boundary conditions are given by

$$\begin{aligned} \boldsymbol{u}(t,\boldsymbol{r}) &= \boldsymbol{0} && \text{in } \widetilde{I} \times \Gamma \backslash \{\boldsymbol{r} \in \mathbb{R}^3 : r_2 = 0.5, -0.5 < r_1, 2r_3 < 0.5\}\ , \\ \boldsymbol{u}(t,\boldsymbol{r}) &= (4,0,0)^T && \text{in } \widetilde{I} \times \Gamma \cap \{\boldsymbol{r} \in \mathbb{R}^3 : r_2 = 0.5, -0.5 < r_1, 2r_3 < 0.5\} \end{aligned}$$

and realised by a technique Skordos [**124**] proposed. The initial condition is set in Ω using the Maxwellian distribution with $\boldsymbol{u} = \boldsymbol{0}$ and $\rho = 1$.

The same technique which is presented in Section 2.3 is applied to ensure the average density to be close to $\rho = 1$. Following the presented reasoning, as well, this can be done since the cavity in the considered configuration is a closed vessel.

2.4.4. Comparison of the Numerical Results with Experimental Data. In the following, the obtained numerical results are compared with experimental data. Hereby, the main goal is to show that the macroscopic and mesoscopic approach lead to numerical results which solve the given problem accurately in the sense that the numerical results are found within the relative measuring accuracy.

The experiments are done by Migeon and published in [**55**]. The purpose of this work is to compare the data with numerical results obtained by an FEM to exhibit high sensitivity to geometrical perturbations. The equipment as well as the visualisation techniques employed for the experiments are described in detail in the mentioned work. Important to note is that the relative accuracy of all measurements is found to be about 3 per cent.

In the following, three planes are selected to compare the results: $r_3 = -\frac{3}{4}, -\frac{1}{2}, 0$ whereby to shorten the notation

$$r_3 = a \text{ denotes } \Omega_a := \Omega \cap \{\boldsymbol{r} \in \mathbb{R}^3 : r_3 = a\} \qquad \text{for } a = -\frac{3}{4}, -\frac{1}{2}, 0\ .$$

The time evolution of two velocity profiles is compared along the middle lines in the three planes:

$$\begin{aligned} &u_1(t,0,r_2,a) && \text{for } a = -\frac{3}{4}, -\frac{1}{2}, 0\ ,\ -0.5 < r_2 < 0.5,\ t = 4,6,8,10,12 \\ &u_2(t,r_1,0,a) && \text{for } a = -\frac{3}{4}, -\frac{1}{2}, 0\ ,\ -0.5 < r_1 < 0.5,\ t = 4,6,8,10,12\ . \end{aligned}$$

The results are found to be very close to each other, especially if one compares the data of the two numerical experiments. For the plane $r_3 = -\frac{3}{4}$ the results are

	Experiment		LBM		FEM	
t	r_1	r_2	r_1	r_2	r_1	r_2
1	0.400000	0.383000	0.387450	0.380843	0.396071	0.379961
2	0.326160	0.320330	0.303963	0.311787	0.307422	0.317039
4	0.216110	0.215280	0.209968	0.211285	0.203598	0.213085
6	0.167590	0.145130	0.163893	0.141055	0.157443	0.143390
8	0.121010	0.089960	0.127640	0.090303	0.124441	0.091127
10	0.080540	0.063900	0.095015	0.064580	0.097394	0.066498
12	0.055000	0.059000	0.076950	0.055980	0.080615	0.054204
6	-0.137747	0.474499	-0.148148	0.469225	-0.177126	0.467614
8	-0.243013	0.439364	-0.257359	0.431507	-0.260520	0.433429
10	-0.317772	0.408334	-0.328105	0.400498	-0.327913	0.406255
12	-0.356548	0.395196	-0.360327	0.390835	-0.357300	0.396686

TABLE 2.1. Position of the centres of the primary eddy for $t = 1, 2, 4, 6, 8, 10, 12$ and of the secondary downstream eddy for $t = 6, 8, 10, 12$ in the symmetry plane $r_3 = 0$.

plotted in Figure 2.8.

In the same three planes the movement of the centres of two major eddies are studied. One is called *primary eddy* and the other *downstream secondary eddy*. In Figure 2.9 their trajectories can be retraced on the basis of the experimentally and numerically gained data. The centres of the eddies are determined by detecting the local minima of $u_1(t, \boldsymbol{r})^2 + u_2(t, \boldsymbol{r})^2$ in the planes $r_3 = 0$, $r_3 = -\frac{1}{2}$ and $r_3 = -\frac{3}{4}$ for several times t. This is done in two steps. At first, the local minima in the discrete space $\Omega_d \cap \Omega_a$ are determined. Then, using the neighbouring nodes of the grid a quadratical interpolation scheme is applied to improve the results. In Tab. 2.1 the positions of the two examined vorticies in the symmetry plane $r_3 = 0$ are stated. This enables a direct comparison of the two applied methods to each other as well as to the experimental data.

The results obtained by the three different approaches are compared with the help of the relative deviation between each two series of positions of the centre of the eddies in an Euklidean norm. Each series consists of the coordinates of the centre of the primary eddy for $t = 1, 2, 4, 6, 8, 10, 12$ and those of the downstream secondary eddy for $t = 6, 8, 10, 12$. The relative deviations are calculated separately for each of the three planes. The results are stated in Tab. 2.2. Comparing the results obtained by means of the LBM with those of the experiments, the relative deviation is found to be less then 4.21 per cent in each plane which is slightly higher than the relative accuracy of measurements of 3 per cent. A similar observation is made comparing the results of the FEM approach with the experimental data. Here, the worst relative deviation of 4.49 per cent is found for the considered plane close to the wall $r_3 = -0.75$. For both numerical approaches holds that the found deviation is the better the closer the considered plane is to the symmetry plane. Further, comparing the results gained by the LBM and the FEM to each other one observes that the relative deviations are smaller than 1.66 per cent.

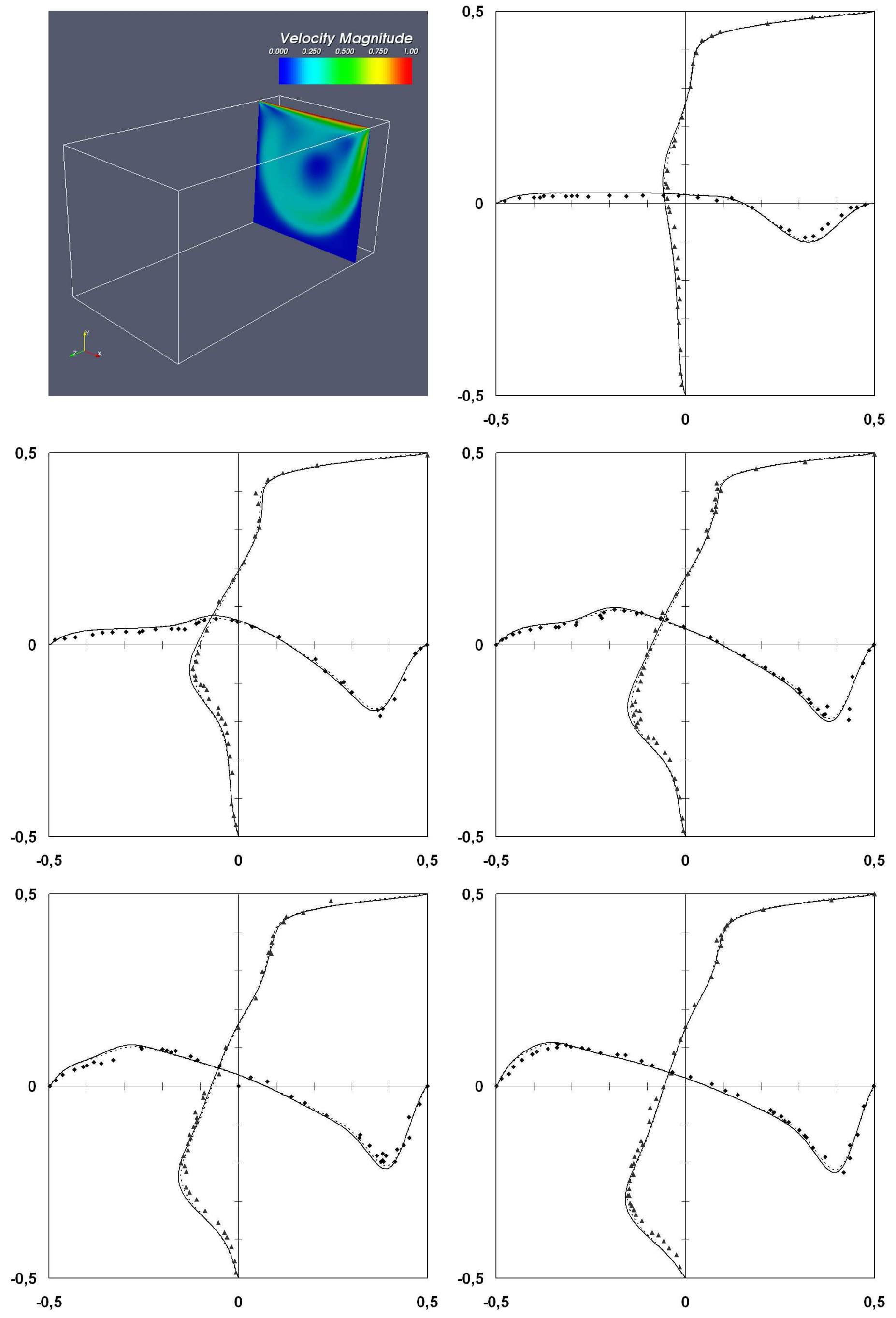


FIGURE 2.8. Velocity profiles in the plane $r_3 = -\frac{3}{4}$ at times $t = 4, 6, 8, 10, 12$ (from left to right and from top to bottom) with $\frac{1}{2}u_1$ as a function of r_2 (abscissa) and $\frac{1}{2}u_2$ as a function of r_1 (ordinate). The symbols ♦ and ▲ stand for the measurements while the lines represent the numerical results obtained by the LBM and the FEM (dashed).

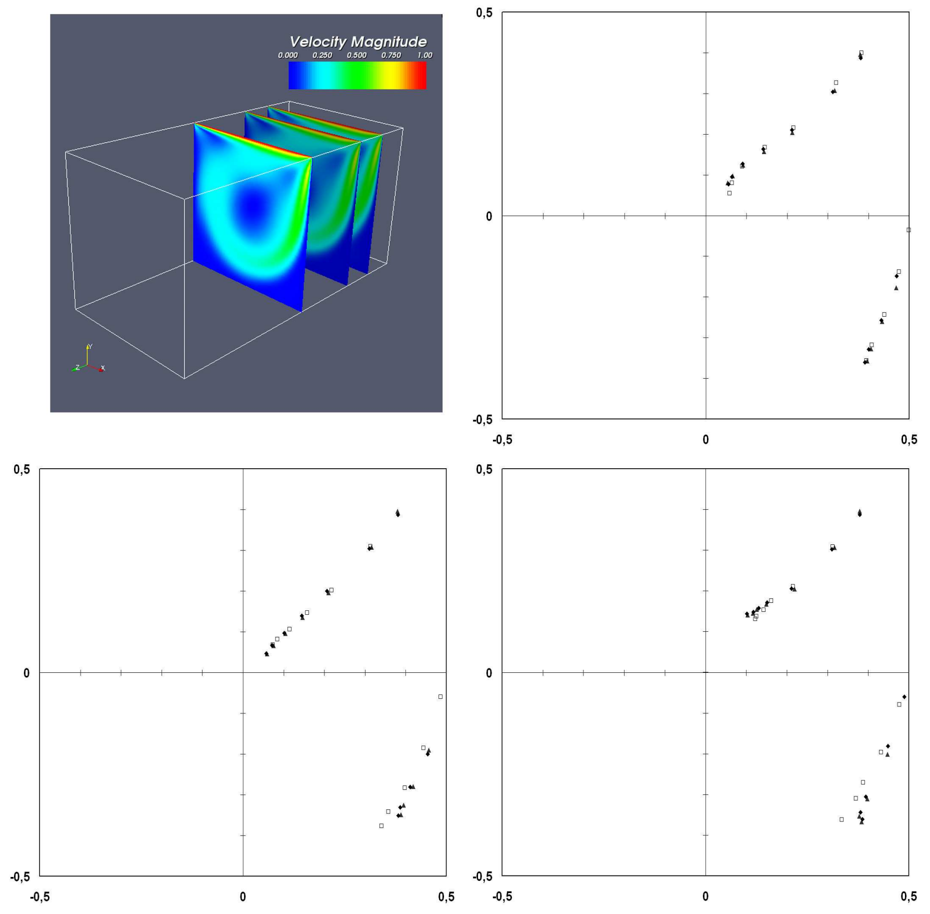


FIGURE 2.9. Position of the centre of the primary eddy and of the downstream secondary eddy at times $t = 1, 2, 4, 6, 8, 10, 12$ in the planes $r_3 = 0$, $r_3 = -\frac{1}{2}$ and $r_3 = -\frac{3}{4}$ (from left to right and from top to bottom). Symbol ♦ stands for the numerical results obtained by the an LBM, ▲ for the results obtained by the an FEM while □ stands for measurements. In the considered planes, the centre of the primary eddy moves towards the centre of the vessel and the centre of the secondary downstream eddy along the right wall towards the lower right corner.

plane	LBM/Experiment	FEM/Experiment	LBM/FEM
$r_3 = 0$	2.27e-2	3.07e-2	1.66e-2
$r_3 = -0.5$	3.76e-2	4.18e-2	1.03e-2
$r_3 = -0.75$	4.21e-2	4.49e-2	1.44e-2

TABLE 2.2. Averaged relative deviation of the positions of the centres of the primary eddy at times $t = 1, 2, 4, 6, 8, 10, 12$ and the secondary downstream eddy at times $t = 6, 8, 10, 12$ in different planes measured in an Euklidean norm.

CHAPTER 3

Hybrid Parallelisation for Lattice Boltzmann Methods

In the last decade LBM have evolved to a mature tool in computational fluid dynamics (CFD) and related topics in the landscape of both commercial and academic software. The simplicity of the core algorithms as well as the locality properties resulting from the underlying kinetic approach lead to methods which are very attractive in the context of parallel and high performance computing (HPC). In that framework it is however a common pitfall to underestimate the complexity of the associated schemes needed to obtain the high performance provided by contemporary computing architectures. New trends in computer architectures such as the multi-core CPUs and coprocessor technologies require specific changes with respect to the mathematical model, algorithm set-up, software design and implementation strategies.

Currently intensive research efforts are undertaken to analyse, develop and adapt adequate LB approaches for dedicated hardware architectures as, e.g., to IBM Cell processors, Graphic Processing Units, Clearspeed accelerator board and multi-core processors from AMD, Intel and others. These new technologies blur the line of separation between architectures with shared and distributed memory. In that context the development of efficient hybrid parallelisation schemes for LBM does not only represent a major challenge in the near future but may become a sine qua non condition to take advantage of the performance both on HPC hardware and on *commodity off the shelf* hardware.

The goal of this chapter is to present a generic hybrid parallelisation concept for LBM allowing coping with platforms sharing both the properties of shared and distributed architecture platforms. The proposed approach relies on a spatial domain decomposition where each sub-domain is represented by a basic block entity which is assigned to a symmetric multiprocessing (SMP) system. For the exchange of data between different block entities a communication-based parallelisation paradigm is employed. The realisation of the proposed concept is illustrated in the framework of the open source software project OpenLB (cf. Appendix A). Besides the model and conceptional aspects, emphasis is placed on the software design and the reworking needed to achieve good performance using OpenMP in that context. It is important to note that currently no standard for an adequate software design exists on multi-core and coprocessor based platforms. This chapter also aims to address peculiarities of the implementation of LBM for such platforms. Parts of the presented work have already been published [**66**] and accepted to be published [**65**]. They are here presented partly in a different context and after a complete revision.

The remainder of this chapter is organised as follows. In Section 3.1 a hybrid parallelisation concept dedicated for LBM is presented. The concept is described by means of presenting its realisation in the OpenLB code. In the following section, issues related to the parallelisation strategies for shared and distributed memory platforms and finally its realisation with OpenMP and MPI, respectively, are addressed. After introducing the chosen high performance computers in Section 3.3, in the last section performance results are discussed by means of considering two three-dimensional test cases. The first one is a problem with an underlying simple geometry, namely the benchmark problem LDC whose corresponding numerical results are presented and discussed in Subsection 2.4.1. The second test case is the fluid flow problem which is subject of Section 6.3, which is roughly spoken dedicated to simulate the respiration in the upper human lungs. In contrast to the simple geometry of the first example, here, the computational domain is complex which poses an additional challenge.

3.1. A Hybrid Parallelisation Concept

The most time demanding steps in LB simulations (cf. Algorithm 1) are the collision and the streaming. Since the collision step is purely local and the streaming step only requires data of the neighbouring nodes, parallelising by domain partitioning leads to low communication costs and is therefore efficient. This has been discussed by many researchers, e.g. in [**69, 96, 103, 112, 143**].

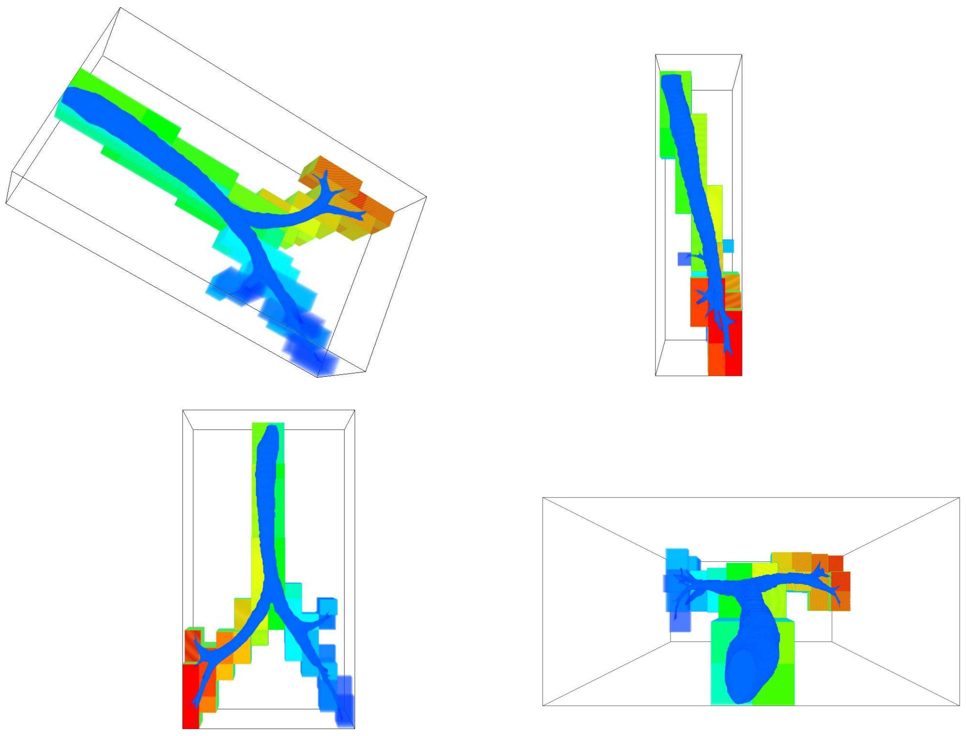


FIGURE 3.1. The partitioning of the lattice Ω_h which represents the geometry of a human lungs. The bounding box pictures the extended lattice $\widetilde{\Omega}_h$ and the coloured cuboids the extended sub-lattices $\widetilde{\Omega}_h^{l_k}$.

The proposed hybrid parallelisation concept also follows the classical approach of partitioning the data according to their geometrical origin. In Subsection 2.2.1

a data structure layout for typical LB implementations is introduced. This concept is now taken up. The strategy relies on partitioning the considered discrete position space Ω_h into $n \in \mathbb{N}$ disjoint, preferably cube-shaped sub-lattices Ω_h^k, $k = 0, 1, ..., n-1$, of as similar sizes as possible.

This becomes feasible by extending Ω_h to a cuboid-shaped lattice $\widetilde{\Omega}_h$ through the introduction of ghost cells. Then, $\widetilde{\Omega}_h$ is split into $m \in \mathbb{N}$ disjoint, always cuboid-shaped extended sub-lattices $\widetilde{\Omega}_h^l$ $(l = 0, 1, ..., m-1)$ of as similar sizes as possible and preferable cube-shaped. Afterwards, all those extended sub-lattices $\widetilde{\Omega}_h^l$ which consist solely of ghost cells are neglected. The number of the remaining extended sub-lattices $\widetilde{\Omega}_h^l$ $(l_0, l_1, ..., l_{n-1})$ defines n. Finally, one gets for each $k \in \{0, 1, ..., n-1\}$ the wanted Ω_h^k as a subspace of $\widetilde{\Omega}_h^{l_k}$ by neglecting the possibly existing ghost cells.

An example is given in Figure 3.1. There, the extended lattice $\widetilde{\Omega}_h$, which is of size $427 \times 213 \times 787$, is split into $m = 1,000$ extended sub-lattices consisting of between $68,796$ and $73,960$ lattice cells. All extended sub-lattices are almost cube-shaped. The worst ratio of highest to lowest number of cells in a arbitrary Cartesian direction of any of the sub-lattices is found to be $\approx 1,103$. Neglecting all those sub-lattices consisting only of ghost cells, 138 sub-lattices remain.

Based on the concept for a data structure for LB scheme, which is presented in Subsection 2.2.1, each extended sub-lattice $\widetilde{\Omega}_h^{l_k}$ is represented by the construct of a `BlockLattice` and the whole set by a `SuperLattice`. Then, each `BlockLattice` is assigned to one SMP unit. There, the parallelisation is based on a paradigm dedicated to shared memory platform which takes advantage of the regular structure of the construct `BlockLattice`. For the exchange of data between the SMP units, i.e. between the `BlockLattices`, a communication-based paradigm is employed. In Figure 3.2 the organisation of the data in the proposed hybrid concept is illustrated. In order to assure an even load balance for in particular complex geometries, the

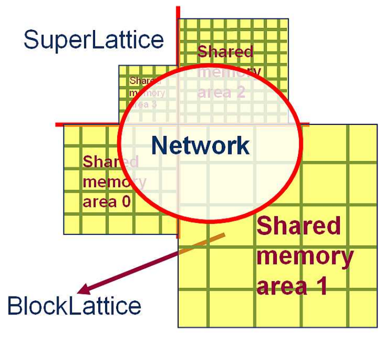


FIGURE 3.2. The underlying data structure layout for LB schemes as proposed in the hybrid parallelisation concept and adapted in OpenLB.

domain is partitioned in a sufficiently large number of disjoint cube-shaped sub-lattices. Then, several `BlockLattices` are assigned to each of the SMP units. This can be done by a sophisticated graph-based partitioning algorithm to find a distribution where the communication costs between the processing units are kept low.

In Section 2.2 the implementation of typical LB algorithms (cf. Algorithm 1) is studied. For the parallel case, it is not categorically necessary to change the basic structure of the sequential algorithm. The usage of pre-processor programming enables to write one single code for the sequential and several parallel modes. The few parts of the program that require a special treatment for the parallel case can be handled by pre-processor directives. An advantage is that one single code simplifies the development of new code. Moreover, there is no loss in efficiency since always either the sequential or a parallel code is compiled. In combination with the usage of a modern object oriented and template based programming language, the adoption of several parallelisation paradigms like OpenMP and MPI and finally their combination is enabled.

3.2. Realisation of the Hybrid Parallelisation Concept

To realise the proposed hybrid parallelisation in OpenLB, MPI is used for communication between the `BlockLattices` within a `SuperLattice` and OpenMP is used for the parallelisation within each single `BlockLattice`. In the framework of OpenLB, pre-processor programming is introduced by setting a variable `PARALLEL_MODE` in the makefile to either `OFF`, `MPI`, `OMP` or `HYBRID`. Then, variables are passed to the compiler so that the pre-processor can prepare the corresponding code. Then, an MPI and/or a OpenMP manager are/is provided to keep the further implementation both, simple and generic. In the code, the managers are realised by means of classes which define a template-generic interface to the corresponding libraries. Thus, necessary methods are provided for common data-types as well as for abstract ones. Further, static settings are fixed through the managers. In parallel mode, instances of the corresponding managers are always provided globally to enable access to the static setting and to the subroutines. In the following section the realisation of the concept is illustrated. Firstly, details regarding the implementation dedicated for shared memory platforms are discussed; and secondly, those dedicated for distributed memory platforms are stated.

3.2.1. Implementation for Shared Memory Platforms with OpenMP. OpenMP (Open Multi-Processing) is an application programming interface for portable shared memory multi-processing programming (see e.g. [**26**] and references therein). The main design goal is to keep the sequential functionality and the source code structure of the program while providing flexible but simple ways to annotate the program that specific parts can be run in parallel. This meets the requirements for the proposed hybrid parallelisation concept as it is introduced in the previous section.

OpenMP is characterised by a rather simple semantic which makes it difficult to capture the complexity of the underlying hardware leading to the following possibly critical issues which need to be considered:

- **Synchronisation overhead:** Frequent change of control flow between one sequential main task and parallel parts can reduce the speedup. Even though there is a huge number of time steps and the execution time for one time step is rather small this is not a severe problem. The EPCC OpenMP Microbenchmarks [**24**] show an overhead per loop construct of about 5 microseconds, with up to 25 microseconds when using a reduction operation. This is only tolerable considering applications of adequate lattice size where the execution of one time step takes many times over these overhead times. For a realistic problem with a lattice size of 100^3, for example, the execution takes 0.53 seconds (1.9 MLUP/s[1]) on an AMD Opteron with clock speed of 2.6 GHz on *HP XC4000* (see Section 3.3 for specifications).

[1]Million fluid-lattice-site updates per second (MLUP/s), here for a *D3Q19* BGK model, is a frequently used measuring unit, e.g. in [**139**].

- **Shared access to common data:** In principle, this is not an issue due to the very regular data structures of the `BlockLattices` whose data are distributed by decomposing the position space. However, excessive shared access to common variables e.g. through statistic methods which collect and reduce data of the whole domain can result in a considerable loss in efficiency.
- **Shared memory bandwidth:** In LB simulations, especially in the three-dimensional case where the number q of discrete velocities $\boldsymbol{v_i}$ $(i = 0, 1, ..., q-1)$ is relatively high, the memory bandwidth can be the limiting restriction (cf. e.g. [**138**]). The possible bottleneck can be reduced or even completely avoided by fusing the two loops for the collision and streaming step together.
- **Consideration of the memory system hierarchy:** The OpenMP programming paradigm supports shared memory parallel programming on many architectures. However, it does not consider the hierarchy of the memory system i.e. it assumes uniform memory access (UMA) where the times to access specific parts of the memory are assumed to be equal. This is in contrast to the considered hardware of non-uniform memory access (NUMA) platforms (cf. Section 3.3). The memory of such NUMA platforms are partitioned into different parts which are local to a specific processor. Unfortunately, these different partitions are not considered in current programming environments, where one cannot specify to which processor the result of a memory allocation should be local. Even worse, because of the shared memory paradigm, one usually has only one memory allocation command per data structure, whose different parts are later accessed by different processors. It is important to note that an allocation command on an operating system like Linux does not actually allocate virtual or even physical memory but address space. The allocation to virtual memory pages takes place when the memory is accessed the first time causing a page fault. Therefore, it is crucial that the first memory access on the data happens in the same pattern the processors will have during the rest of the calculations. The worst case occurs when the initialisation is left sequentially and all the allocated memory is the root processor's local memory.

According to the hybrid parallelisation concept presented in Section 3.1, each of the extended sub-lattices $\widetilde{\Omega}_h^{l_k}$ $(k = 0, 1, ..., n-1)$ is represented by a `BlockLattice` which is assigned to a SMP unit. The parallelisation strategy for shared memory platforms also relies on domain partitioning. This is realised by OpenMP pragmas which split the loops over all `Cells` of the `BlockLattice` in every collision and streaming step. In the following, two different approaches are presented. In contrast to the first one, the second one does respect the memory system hierarchy. In Section 3.4 the performance of the two approaches are compared.

If the collision and streaming step are executed separately in two loops over all `Cells`, the data access is exclusive and the order of access is arbitrary. It is stressed that this also holds for the streaming step due to a swapping technique [**97, 98, 85**] which is implemented in OpenLB. In the first approach the collision (c) and streaming (s) step are parallelised straightforwardly using OpenMP directives for loops. Using an Intel compiler, it turns out that at both testing platforms (cf. Section 3.3) for the collision step a dynamic scheduling with block size one is the best choice, while for the streaming step the default scheduling is to be preferred. The OpenLB source code of the parallelised collision step is shown exemplarily for

```
template<typename T, template<typename U> class Lattice>
void BlockLattice2D<T,Lattice>::
                    collide(int x0, int x1, int y0, int y1)
{
  OLB_PRECONDITION(x0>=0 && x1<nx);
  OLB_PRECONDITION(x1>=x0);
  OLB_PRECONDITION(y0>=0 && y1<ny);
  OLB_PRECONDITION(y1>=y0);

  int iX, iY;
  #ifdef PARALLEL_MODE_OMP
  #pragma omp parallel for private (iY) schedule(dynamic,1)
  #endif
  for (iX=x0; iX<=x1; ++iX) {
    for (iY=y0; iY<=y1; ++iY) {
      grid[iX][iY].collide(getStatistics());
      grid[iX][iY].revert();
    }
  }
}
```

Listing 3.1. The OpenLB source code for a parallelised collision step (c). In the parallel modes `OMP` and `HYBRID` the variable `PARALLEL_MODE_OMP` is defined. Thus, the compiler can parallelise the loop over all `Cells` of each `BlockLattice`.

the presented straightforward approach in Listing 3.1.

It is also possible to process both steps in one single loop (*bulk c/s*), even if the data of the distribution function $f_i(\boldsymbol{r})$ ($i = 1, 2, ..., q-1$ and $\boldsymbol{r} \in \widetilde{\Omega}_h^{l_k}$) are stored only once in one array [**85, 97, 98**]. Unlike the previously described procedure, the access order is not arbitrary. The execution of a `Cell` now depends on its own data and that of some particular `Cells` in the neighbourhood. However, there is no data dependency to `Cells` that are located somewhere in positive r_1-direction [**85**]. Making use of this specificity, the domain of a `BlockLattice` is split along that direction into layers of as equal size as possible and each layer is assigned to a thread. To handle the correlation of the data the parallel *bulk c/s* is done in three steps. Firstly, the lower boundary `Cells` of each layer perform a collision step. Secondly, the layers without the lower boundary as new formed blocks execute a *bulk c/s*. Finally, after synchronising all threads, the lower boundary `Cells` process a streaming step.

In order to improve the efficiency on multi-processor platforms with non-uniform memory access, it is essential to ensure a static memory access. As pointed out before, data should be split into parts whereas each part is assigned to a specific thread and preferably accessed by this particular thread. In the second approach, the domain which is represented by a `BlockLattice` is split into blocks of as equal size as possible exactly as it is done for the *bulk c/s* described before. The OpenMP directive for loops with a static scheduling with the associated block size is used for the initialisation of the data, the collision (c*) and streaming (s*) step. The approach where the collision and streaming is done in one single loop remains unchanged and is now referred to as *bulk c/s**. Threads are bound to specified cores by calling the Linux system function `sched_setaffinity()` to avoid the operating system moving the threads. As mentioned previously in Section 3.1 an OpenMP manager is provided to simplify the implementation. To enable a static assignment of a sub-domain to a specific thread the instance is given for each thread and initialised with the number of dynamic threads set to zero.

In OpenLB a class provides statistic data of a dedicated `BlockLattice` e.g. the average density, average energy and maximum velocity. A method of this class is called while a `Cell` performs a collision, whereby it collects the statistic data and saves them in variables provided by an instance of the class. Since an exclusive access of one thread to an instance of the statistic class blocks the other threads, an approach using the corresponding OpenMP directive is inefficient. Additionally provided instances of this class for each thread followed by a reduction solve that shortcoming satisfyingly. Thereby one has to take care to avoid that any two instances can be loaded into the same cache-line, for this would also prevent access to the data for one thread as long as the other has loaded the line into its cache. This problem is solved by encapsulating each instance through allocating and assigning memory straight before and after an instance is constructed.

3.2.2. Implementation for Distributed Memory Platforms with MPI. The hybrid parallelisation concept presented in Section 3.1 postulates that the data belonging to a `SuperLattice` are distributed in memory of several SMP units. Moreover, it is supposed that to each of the employed SMP units belongs the data of one or more `BlockLattices`. To perform a streaming step, the data of all neighbouring `Cells` are needed. Since generally SMP units do not have direct access to the memory associated to other SMP units, the demanded data needs to be communicated. The realisation in OpenLB used the MPI (message passing interface) standard which was established in 1994 by the MPI forum [**101**].

Algorithm 2 Basic parallel lattice Boltzmann algorithm for the adoption of communication-based parallelisation paradigms like MPI.

```
1. Reading input
2. Simulation setup
3. Time loop
for $t = t_0$ to $t = t_{max}$ do
   a) Collision
   > Blocking
   > Communicating
   > Writing to ghost cells
   b) Streaming
   c) Post-processing
   d) Writing output
```

A common and efficient strategy [**83**] is to introduce a layer of additional `Cells` *(ghost cell layer)* around each extended sub-lattice $\widetilde{\Omega}_h^{l_k}$ $(k = 0, 1, ..., n-1)$. The size of the layer depends on the applied LB model. For the $D2Q9$ and $D3Q19$ model e.g. only the data of direct neighbouring `Cells` is needed. Thus a ghost cell layer of size 1 in each space direction is sufficient. In OpenLB these extended sub-lattices are also treated as `BlockLattices`.

If the collision and streaming step are executed in two loops (*c/s*), each of the `BlockLattices` performs the collision step only on the sub-lattice without the envelope of ghost cells. Afterwards the data required by other processing units are consolidated into blocks and sent to those units where they are finally written to the ghost layer of the sub-lattice. Finally, a generic streaming step over all `Cells` of the sub-lattice without the layer is performed. Thus, the basic structure of the classical sequential LB algorithm (cf. Algorithm 1) is preserved. Due to three additionally introduced steps, namely *blocking*, *communicating* and *writing to ghost cells* the collision and streaming step remain unchanged. A prototypical parallel LB algorithm is given with Algorithm 2.

In case of executing the collision and streaming step in one single loop (*bulk c/s*), all `Cells` at the boundary of the `BlockLattice` without the ghost cell layer perform a collision step, followed by a *bulk c/s* of the other `Cells` of this sub-lattice. It is to be noted that so far all data of the `Cells` from the ghost cell layer are not touched. Then, a *blocking*, *communicating* and *writing to ghost cells* step is performed exactly to those described before. Finally, a streaming step of all `Cells`, belonging to the boundary of the `BlockLattice` without the envelope, concludes the procedure.

3.3. Computer Architecture Based Issues

Three high performance computers are considered to be appropriate platforms to test the hybrid parallelisation approach. All three architectures consist of a number of nodes with one or more multi-core processors which are connected by a network. Hence, they all belong to the class of computers which blur the line of separation between architectures with shared and distributed memory.

Two high performance computers at the Steinbuch Center for Computing at the University of Karlsruhe are considered. The first of them provides shared memory nodes with uniform memory access. There, solely the OpenMP implementation is tested, whereas, on the second computer both the OpenMP-based and MPI-based implementation are tested. The third environment chosen for testing is a supercomputer at the Forschungszentrum in Jülich. Here, the focus is placed on testing the MPI-based implementation for a highly computing-time demanding problem.

The first test environment is an HP Integrity RX8620 16-way node which is partitioned into two 8-way (logical) nodes. Each Itanium-2 processor runs at a speed of 1.6 GHz and has 6 MB of level 3 cache. This (logical) node has 64 GB of main memory and one Quadrics QsNet II adapter and is part of a larger *HP XC6000* installation with another 100 2-way Itanium-2 nodes. Though eight processors share two memory banks, an interleaving mechanism provides uniform memory access where the aggregate memory bandwidth is 12.8 GB/s.

The second high performance computer at the Steinbuch Center for Computing considered to test the approach is the *HP XC4000.* It is equipped with 750 two-way nodes (HP ProLiant DL145 G2), which reach a peak performance of 15.6 TFLOP/s, 12 TB total main memory and an InfiniBand 4X DDR Interconnect. One node provides two AMD Opteron sockets, running at a clock speed of 2.6 GHz, and 16 GB of main memory. Each socket hosts one dual-core processor with 1 MB of level 2 cache. The two cores of one processor share a memory bandwidth of 6.4 GB/s. The memory access is non-uniform, i.e. each socket has faster access to its own memory.

Jülich Research on Petaflop Architectures (*JUROPA*) is the name of the high performance computer at the Forschungszentrum in Jülich. This supercomputer reaches a peak performance of 207 TFLOP/s. It hosts in total 2, 208 nodes. Each one is equipped with two Intel Xeon X5570 (Nehalem-EP) quad-core processors running at a clock speed of 2.93 GHz and 24 GB of main memory. The nodes are connected by Infiniband QDR with non-blocking fat tree topology and a Sun Data Center Switch 648.

3.4. Presentation and Discussion of the Performance Results

In order to test the performance of the proposed parallel approach, two test cases are considered. The examples fundamentally differ by their underlying geometries. The first computational domain is a hexahedron and thence simple. In contrast, the second fluid domain is the upper part of a particular human lungs. The implementation of the proposed hybrid parallelisation strategy is employed to solve, in particular, the two problems which are considered in the previous chapter in Section 2.3 and Section 2.4. Since the numerical results of both tests cases are validated by comparing them with an analytical solution and, respectively, with experimentally gained data as well as with results gained by applying another numerical method, the implementation can be seen as validated. Therefore, this issue is not considered in the following.

3.4.1. Test Case with a Simple Geometry: 3D Lid-Driven Cavity. In the following, the performance results of the hybrid parallel approach are studied in detail for an example with a simple underlying geometry. Thereto, a 3D LDC problem similar to that formulated in Subsection 2.4.1 is considered. The applied LBM is chosen exactly like the one specified in Subsection 2.4.3 except for the considered discretisation parameter $h \in \mathbb{R}_{>0}$ and the number of time steps. The discretisation parameter $h \in \mathbb{R}_{>0}$ and with it the lattice size is kept variable to test the scalability of the parallelisation. In order to keep the total execution times low, the number of time steps is always set to 100. Since the computation and copying operations are identical for every single time step an examination of this number of time steps is sufficient to get scalable results.

All source code is compiled with the Intel compiler using optimisation level 3. The compiler uses two different optimisation settings for performance tuning, namely one for the sequential mode and another for all parallel modes where OpenMP is involved. It is important to note that this optimisation setting involves specific heuristics especially to control the use of memory bandwidth among processors. It turns out that this setting leads to an increase of the overall computing times on a single thread. Each configuration is executed at least three times to resolve random, possibly hardware caused artefacts polluting the presented results, whereby the best one of them is presented in the following. The underlying measure for the performance analysis is the *efficiency* E_{ff} which is defined according to

$$E_{ff} := \frac{t_1}{p\ t_p}\ , \tag{3.1}$$

where t_p is the absolute time in seconds needed to perform 100 time steps with p processing units employed.

In order to enable a clear and meaningful analysis of the hybrid parallelisation approach, the performance tests are put in execution separately for the OpenMP-based and MPI-based codes. For the OpenMP-based tests the domain of the whole cavity is represented by one single `BlockLattice` which is assigned to one SMP

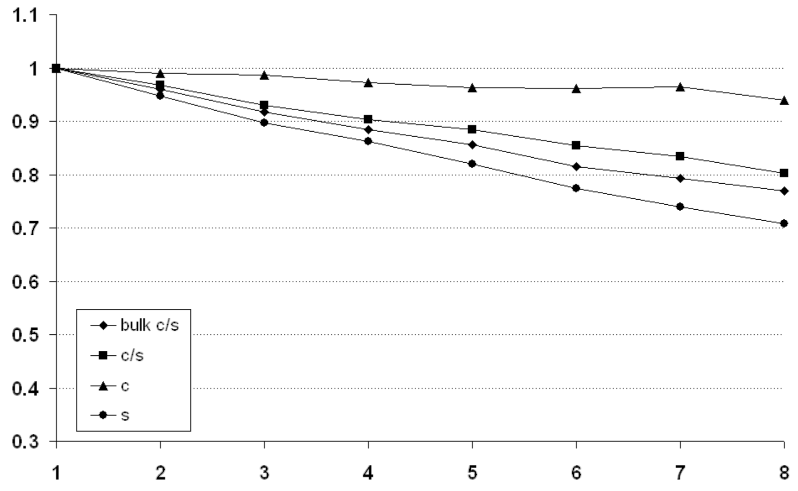


FIGURE 3.3. Efficiency as a function of the number of threads for the 3D LDC problem considered in Section 2.4 on a lattice of size 201 × 201 × 401 obtained on the *HP XC6000*. The abbreviations in the legend are defined in Subsection 3.2.1.

mode	threads	bulk c/s	c/s	c	s	compiler flags
sequential	1	2,405 s	3,006 s	1,359 s	1,628 s	-O3
OpenMP	1	2,642 s	3,132 s	1,459 s	1,635 s	-O3 -openmp
	2	1,377 s	1,619 s	737 s	863 s	
	3	960 s	1,122 s	493 s	608 s	
	4	747 s	867 s	375 s	474 s	
	5	617 s	709 s	303 s	399 s	
	6	540 s	611 s	253 s	352 s	
	7	476 s	537 s	216 s	316 s	
	8	429 s	488 s	194 s	289 s	

TABLE 3.1. Computing times for 100 time steps of the 3D LDC problem considered in Section 2.4 on a lattice of size 201×201×401 measured on the *HP XC6000*. The abbreviations in the legend are defined in Subsection 3.2.1.

unit of the two considered test platforms, namely the *HP XC4000* and *HP XC6000* (cf. Section 3.3). For the analysis of the parallelisation dedicated for distributed memory platforms the domain is represented by a `SuperLattice` which is split into an *a priori* given number of `BlockLattices`. Each of the `BlockLattices` is assigned to one of the employed processing SMP units, whereby for the tests only one core per SMP unit is employed. Thus, the MPI-based approach can be directly compared to the OpenMP-based approach.

The performances related to the first of the OpenMP-based approaches described in Subsection 3.2.1, namely *c/s* and *bulk c/s*, are presented in Table 3.1 and Figure 3.3 for the computations on the platform *HP XC6000* and in Table 3.2 and Figure 3.4, respectively, for the computations on the platform *HP XC4000*. The results obtained on the platform which provides uniform memory access, namely a part of the *HP XC6000*, show a smooth and almost linear decrease of the efficiency

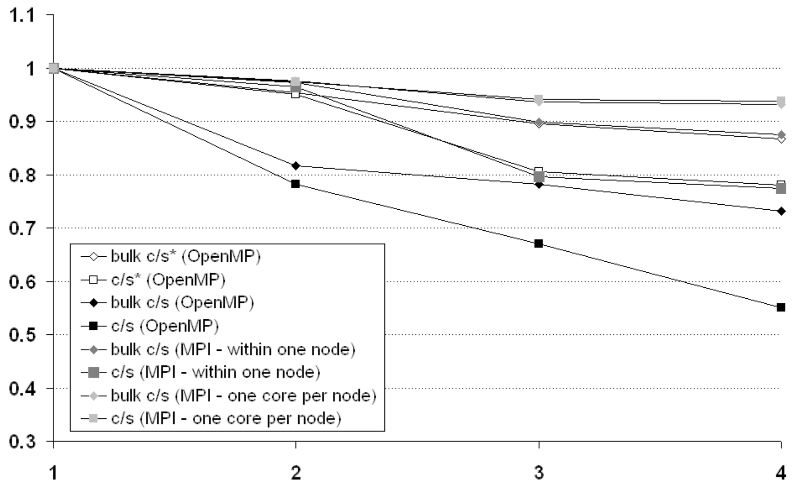


FIGURE 3.4. Efficiency as a function of the number of threads for the 3D LDC problem considered in Section 2.4 on a lattice of size $201 \times 201 \times 401$ obtained on the *HP XC4000* in comparison with the results of an MPI-based parallelisation. The abbreviations in the legend are defined in Subsection 3.2.1.

mode	threads	bulk c/s	c/s	c	s	compiler flags
sequential	1	902 s	947 s	589 s	330 s	-O3
OpenMP	1	1,123 s	1,008 s	614 s	390 s	-O3 -openmp
OpenMP	2	687 s	645 s	332 s	311 s	
	3	479 s	501 s	227 s	275 s	
	4	384 s	458 s	178 s	281 s	
OpenMP (local memory)	2	589 s	530 s	314 s	213 s	
	3	418 s	417 s	211 s	201 s	
	4	324 s	323 s	161 s	158 s	
MPI	1	935 s	960 s	597 s	347 s	-O3
MPI (within one node)	2	481 s	498 s	293 s	195 s	
	3	347 s	402 s	200 s	204 s	
	4	267 s	310 s	150 s	156 s	
MPI (one core per node)	2	479 s	493 s	293 s	192 s	
	3	333 s	340 s	195 s	141 s	
	4	251 s	256 s	145 s	105 s	

TABLE 3.2. Computing times for 100 time steps for the 3D LDC problem considered in Section 2.4 on a lattice of size $201 \times 201 \times 401$ measured on the *HP XC4000*. The abbreviations in the legend are defined in Subsection 3.2.1.

as a function of threads, leading to an efficiency of approximately $E_{ff} \approx 0.80$ for eight threads. While the performance results of the three tests on this platform are found with small variations, the results of the first approach obtained on the platform without uniform memory access (*HP XC4000*) show variations of up to 20 per cent. Moreover, the reached efficiency is much smaller. These results are in accordance with numerical tests obtained on other hardware platforms by means of OpenMP for LBM [**143, 144**].

A comparison of the efficiencies obtained on the platform *HP XC4000* for the approaches taking into account a local memory allocation and a local first touch of data (*c/s**, *bulk c/s**) is depicted in Figure 3.4. The efficiency can be improved significantly and the variation of the test results is found to be less than 5 per cent. The best result for that approach is achieved for the *bulk c/s** using two threads, reaching an efficiency of $E_{ff} = 0.95$, while the first approach only leads to $E_{ff} = 0.82$.

Apart from a few exceptions on both considered platforms it is observed that performing the collision and streaming step in one single loop over all `Cells` (*bulk c/s*) is more efficient, considering the absolute execution times, than in two separated loops (*c/s*). This is pointed out as well for other platforms in [**97, 98**]. It is interesting to note that this holds for both the sequential and the parallel implementation for the presented benchmark problem.

In Figure 3.4 the efficiency of the purely MPI-based approach is presented and compared to the corresponding results of the OpenMP versions. It is stressed that the considered implementation tested on one node uses a direct access to the local shared memory. It is important to notice that in the case of three and four processes the version executed on three respectively four different nodes i.e. involving extranodal communication is more efficient than the version involving intranodal communication. In the case of two processes there is no significant difference to observe. Employing only one core per socket, then, each core can almost saturate the memory-to-socket bandwidth in contrast to the case of three and four cores per node and thus the limitation of the bandwidth becomes visible. Comparing the efficiency based on the overall execution time using the same code executed on one core, the OpenMP version with the local memory assignment and the MPI version are found to be of high conformance. Though, considering the absolute execution times as they are presented in Table 3.2, it is observed that the MPI version is more efficient than both OpenMP based approaches. Moreover, the MPI results clearly show the intricate dependency between the efficiency and job partitioning at the core and nodal level. The motivation of the proposed hybrid parallelisation concept relies on the idea that this kind of local partition at the nodal level should automatically be treated by the considered parallelisation paradigm. Since OpenMP is actually intended to offer such an interface, the results clearly reveal the potential of optimising the considered implementation of OpenMP.

As last part of the performance tests, the MPI-based approach is tested on up to 256 nodes on the *HP XC4000*. In contrast to the other tests, the cavity of the underlying 3D LDC problem is chosen to be cube-shaped. However, the considered LB method as well as the testing strategy and environment remain the same. The results are presented in form of the reached efficiency for various grid sizes in Figure 3.5. It is observed that the efficiency increases if the grid size increases and the number of employed cores is fixed. For a grid size of 401^3 an efficiency of approximately $E_{ff} \approx 0.77$ is reached when 256 cores are employed while for a grid of size 301^3 the corresponding efficiency is about $E_{ff} \approx 0.68$.

Summing up the presented benchmark results, the shared memory approach with OpenMP cannot compete against the MPI version on SMP nodes. This somewhat disappointing result is partly due to heuristics used by the compiler when using OpenMP leading to a loss in efficiency even in the case of sequential execution. To achieve better performance results the memory hierarchy and moreover on platforms with NUMA a binding of threads to specific processors have to be

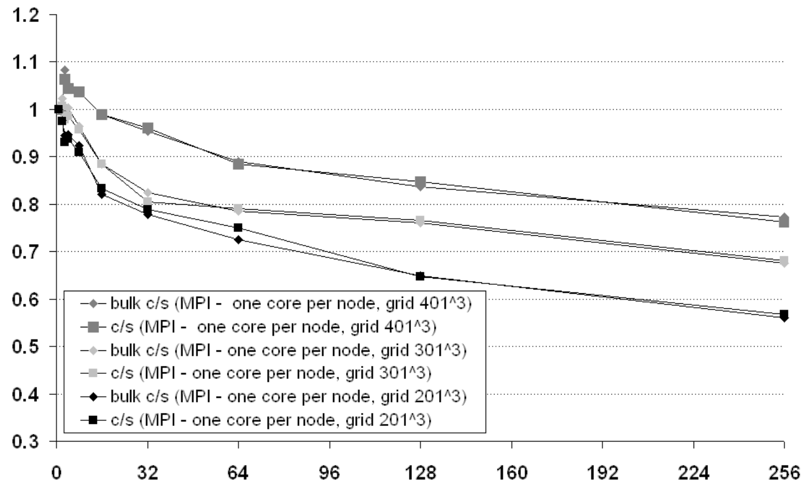


FIGURE 3.5. Efficiency as a function of the number of processes for 100 time steps for the 3D LDC problem considered in Section 2.4 yet on a cube-shaped lattice of various sizes obtained on the *HP XC4000* of the MPI-based parallelisation. The abbreviations in the legend are defined in Subsection 3.2.1.

taken into account. Since OpenMP does not provide any support for memory hierarchies and to control affinity, the programmer is left to use utilities provided by the operating environment.

Multi-core platforms are however a reality which cannot be circumvented, and whose importance is expected to increase strongly in the near future. It can also be expected that the needs of high performance computing are better taken into account in future hardware platforms, as the communication between cores is improved. Hybrid implementations as the one presented in this chapter are therefore of crucial importance. It is believed that the approach described here takes into account the fundamental philosophy of LBM. It combines efficiency with ease of use and could serve as a programming paradigm for LB implementations on current and on future computation platforms.

3.4.2. Test Case with a Complex Geometry: Flow in the Upper Human Lungs. In this subsection the pure MPI-based implementation, which is part of the hybrid parallelisation approach, is tested for a problem with an underlying complex geometry. Thereby, the main aim is to present and discuss the obtained performance. The upper human lungs are considered to be a suitable example for a fluid flow problem with a complex geometry. The same problem is in focus of Section 6.3. There, the problem is formulated, the applied discretisation methods are stated and the obtained numerical results are presented and discussed. The computational domain is extracted from *computer tomography* (CT) data of a patient. Then, the data is prepared as described in Section 6.1 which leads to discrete lattices Ω_h with $h = 0.23$ mm. This and two finer lattices Ω_h with $h = 0.23$ mm$/N$ ($N = 2, 3$) are considered to illustrate the parallelisation strategy and in particular test its scalability. Thereby, the quasi-refinement is based on the approach mentioned in Subsection 6.1.3.

According to the concept presented in Section 3.1, the considered lattices Ω_h with $h = 0.23$ mm$/N$ ($N = 1, 2, 3$) need to be partitioned. Thereto, at first all

refinement level N	dimension of $\widetilde{\Omega}_h$	fluid cells	non-boundary / fluid cells	boundary/ fluid cells
1	$452 \times 234 \times 859$	$2,327,357$	87.8%	12.2%
2	$900 \times 464 \times 1714$	$17,465,152$	93.6%	6.4%
3	$1350 \times 696 \times 2571$	$57,675,848$	95.7%	4.3%

Table 3.3. The considered extended discrete lattices $\widetilde{\Omega}_h$ representing the geometry of human lungs for different discretisation parameters $h = 0.23$ mm/N whereby $N = 1, 2, 3$ is the corresponding refinement level (cf. Subsection 6.1.3).

case name	refinement level N	initial cuboids m	remaining cuboids n	min. cell cuboid	max. cell cuboid
N1m1200	1	$1,200$	152	73,710	77,142
N1m12000	1	$12,000$	744	6,859	8,000
N1m24000	1	$24,000$	$1,347$	3,375	4,096
N1m48000	1	$48,000$	$2,258$	1,728	2,028
N2m1200	2	$1,200$	161	581,256	602,454
N2m12000	2	$12,000$	741	57,798	62,400
N2m24000	2	$24,000$	$1,310$	28,830	30,752
N2m48000	2	$48,000$	$2,254$	14,440	15,625
N3m1200	3	$1,200$	158	2,000,349	2,026,056
N3m12000	3	$12,000$	734	198,476	205,320
N3m24000	3	$24,000$	$1,296$	97,336	103,776
N3m48000	3	$48,000$	$2,210$	47,952	52,022

Table 3.4. The table shows all considered test cases and provides for each of the cases ($h = 0.23$ mm/N with $N = 1, 2, 3$) the initial number m of sub-lattices $\widetilde{\Omega}_h^l$ ($l = 0, 1, ..., m-1$), the remaining number n of sub-lattices $\widetilde{\Omega}_h^{l_k}$ ($k = 0, 1, ..., n-1$) as well as the minimal and maximal number of cells in the respective remaining lattices $\widetilde{\Omega}_h^{l_k}$.

Ω_h ($h = 0.23$ mm/N with $N = 1, 2, 3$) are extended to cuboid-shaped lattices $\widetilde{\Omega}_h$ through the introduction of ghost cells. An overview concerning the dimensions of the different considered extended lattices $\widetilde{\Omega}_h$ as well as its corresponding numbers of fluid, non-boundary and boundary cells is given in Table 3.3. In Figure 3.1 the obtained remaining extended sub-lattices obtained for $m = 1,000$ but also for the geometry of the upper human lungs are visualised.

Then, the three considered extended lattices $\widetilde{\Omega}_h$ ($h = 0.23$ mm/N with $N = 1, 2, 3$) are split into $m = 1,200,\ 12,000,\ 24,000,\ 48,000$ disjoint extended sub-lattices $\widetilde{\Omega}_h^l$ ($l = 0, 1, ..., m-1$). Afterwards, all sub-lattices which just consist of ghost cells are neglected, leaving a certain number $n \leq m$ of extended sub-lattices $\widetilde{\Omega}_h^{l_k}$ ($k = 0, 1, ..., n-1$). This leads to twelve different test cases which are listed in Table 3.4. There, for each of the test cases ($h = 0.23$ mm/N with $N = 1, 2, 3$) the obtained number n of remaining lattices $\widetilde{\Omega}_h^{l_k}$ ($k = 0, 1, ..., n-1$) as well as the minimal and maximal number of cells in the respective remaining lattices $\widetilde{\Omega}_h^{l_k}$ are specified.

According to the presented strategy, each of the remaining extended sub-lattices $\widetilde{\Omega}_h^{l_k}$ $(k = 0, 1, ..., n-1)$ is represented by one single `BlockLattice`, which all together constitute a `SuperLattice`. Then, the `BlockLattices` are distributed among the available SMP nodes by building blocks with respect to the numbering. The underlying LB algorithm is based on a $D3Q19$ model whereby the collision and streaming step in every time step is done in one single loop (*bulk c/s*). Further details concerning the chosen discretisation, in particular about the boundary conditions, can be found in Subsection 6.3.3.

The twelve test cases are executed on the *JUROPA*. Thereby, in every case a different number of cores p, but always with just one core per node, is employed. The C++ source code is compiled with the Intel compiler using optimisation level 3. As for the first in the previous subsection considered example, each configuration is tested three times to resolve random, possibly hardware artefacts. Then, the best measured time of the three is presented in the following. In order to keep the total execution times low, the number of to be performed time steps is always set to 100. The time in seconds measured for these 100 steps obtained employing p processes, respectively nodes, is captured by the variable t_p. For some test cases, more than the available memory on one node of 16 GB is required. Thus, t_1 is not available in those cases. Therefore, a comparison of the performance results based on the efficiency E_{ff} as defined in (3.1) is not feasible. In order to compare performances of LB implementations obtained on different computers or obtained for different implementation techniques, the measuring unit *million fluid-lattice-site updates per second* MLUP/s is frequently introduced, e.g. in [**139**]. In the following, this terminus is extended to also enable comparisons of in parallel executed LB code by introducing the measuring unit MLUP/ps, which stands for *million fluid-lattice-site updates per process and second.* The performance results obtained for the test cases are measured in this unit and captured by the variable P_{LB} which is defined according to

$$P_{LB} := 10^{-4} \frac{N_c}{t_p p}$$

whereby N_c denotes the number of fluid cells.

In Figure 3.6 the measured performances P_{LB} are plotted as a function of the employed number of MPI-processes $p = 2^0, 2^1, ..., 2^8$. Considering the obtained graphs of all test cases, a general observation made is a slight decline of P_{LB} for increasing p. This is in accordance with the obtained performance results for the problem with an underlying simple geometry which is considered in the previous subsection.

For all considered test cases ($m = 1,200,\ 12,000,\ 24,000,\ 48,000$ and $p = 2^0, 2^1, ..., 2^8$) it is observed than the greater the refinement level N is chosen the better are the obtained performances P_{LB}. The best value $P_{LB} \approx 2.14$ MLUP/ps is measured for $m = 12,000$ and $N = 3$ on $p = 8$ nodes. This characteristics can be explained by the fact that the computational costs for a boundary cell is usually higher that for a non-boundary cell and that the ratio of boundary cells to fluid cells increases for smaller problem sizes as stated in Table 3.3).

Further, it is observed that for the cases where the number of initial cuboids m is the smallest, i.e. $m = 1,200$, in general the obtained performances P_{LB} are the smallest. In this cases the number of obtained `BlockLattices` ranges from $n = 152$ to $n = 161$ while for the other cases it is much greater ($n \geq 734$). This leads to a relative bad load balance especially for the cases where $p \geq 64$. However, for a

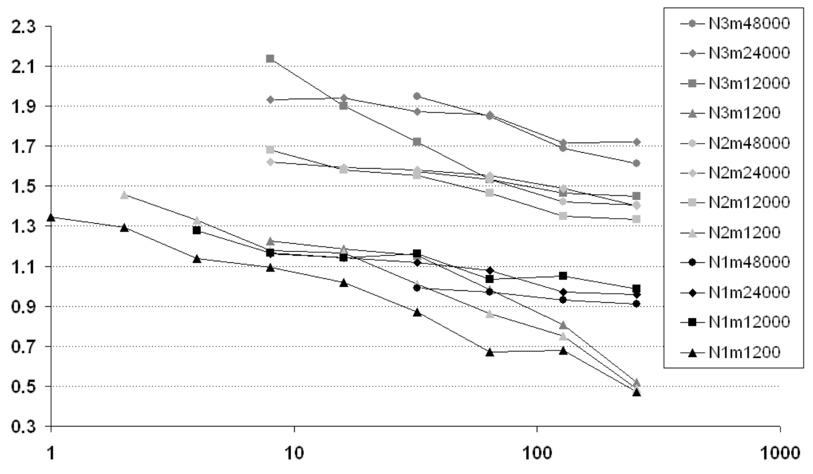


FIGURE 3.6. The graph shows the obtained values for P_{LB} in MLSUP/ps (million fluid-lattice-site updates per process and second) as a function of the number of employed MPI-processes p for 100 times steps of an $D3Q19$ LB algorithm obtained on the *JUROPA*. The underlying computational domain is obtained by CT scans of the upper human lungs. The problem is subject of Section 6.3.

fixed problem size, which is here characterised by N, the overhead due to communication costs will be greater if m and with it n increases. Thus, then increasing m, it is expected that the positive effect of a better load balance is compensated by additional costs due to increasing effort for the three steps in Algorithm 2: *blocking*, *communicating* and *writing to ghost cells*. This is indeed observed then considering the test results. By trend, in the case where $N = 1$ the best result is obtained for $m = 12,000$ and for the cases $N = 2, 3$ for $m = 24,000$. Yet, it is to be noted that the measured performances P_{LB} are relatively close to each other in the majority of the considered cases with $m \geq 12,000$. In particular, comparing the results obtained for $m = 24,000$ to those for $m = 48,000$, no significant drop of the performances P_{LB} is observed.

Summing up, the parallel performance results for the considered complex geometry of a human lungs are found to confirm the efficiency of the underlying approach in terms of that the computing time can almost be halved if the number of employed nodes is doubled. Thereunto, it is important to ensure a good load balance, which is reached by providing a sufficient large number m of `BlockLattices`. The overhead for large values of m is observed to be rather small for the considered cases. It is expected that applying a sophisticated graph-based partitioning algorithm would lead to a reduction of this overhead since relative expensive network communication could be replaced by memory operations.

CHAPTER 4

Fluid Flow Control and Optimisation with Lattice Boltzmann Methods and Automatic Differentiation

In the three previous chapters mesoscopic models to describe fluid flow problems and numerical methods to solve them are presented, discussed and applied. In this framework the considered fluids are always restricted to be incompressible and Newtonian. Now in this and later in the following chapter, the complexity of the problems to be considered is increased. The focus is brought to optimal flow control and flow optimisation problems whereas the underlying models describing the fluid flows are of mesoscopic nature. In the following, such problems are to be considered which can be formulated as constrained optimisation problems in an abstract manner according to

$$\left.\begin{array}{l}\text{find control } \boldsymbol{\alpha} \text{ and state } f \text{ which} \\ \text{minimise } J(f,\boldsymbol{\alpha}) \text{ and fulfill } \boldsymbol{G}(f,\boldsymbol{\alpha}) = \boldsymbol{0} \ .\end{array}\right\} \tag{4.1}$$

Herby, the function f is said to be the *state*, the function $\boldsymbol{\alpha}$ the *control*, the functional J the *objective* or *cost functional* and $\boldsymbol{G}(f,\boldsymbol{\alpha}) = \boldsymbol{0}$ the *constraint* or *side condition*. The side condition couples the control $\boldsymbol{\alpha}$ with the state f. It comprises the governing equation of a mesoscopic model. This is in the following always a BGK-Boltzmann equation (2.3). Thus, with the BGK-Boltzmann equation the function $\boldsymbol{G}$ which establishes the side condition is generally also non-linear in f.

The considered approach of formulating an optimal control or optimisation problem with a mesoscopic model describing the underlying fluid flow is in contrast to the classical approach where the fluid flow is modelled macroscopically. There, the incompressible Navier-Stokes equation serves as governing equation for the fluid flow as part of the side conditions (cf. Gunzburger [**56**], Hinze [**70**] and references therein). Both approaches share the complexity of the side condition which makes an analytical and numerical investigation challenging. However, comparing the efforts made to investigate optimisation problems of the two different formulation approaches, little has been done to investigate the here considered mesoscopic approach. To the knowledge of the author, questions regarding existence and uniqueness of solutions have not been addressed at all.

Generally, if one wants to solve fluid flow optimisation problems numerically two main strategies can be distinguished. They are known as *first-discretise-then-optimise (approach)* and *first-optimise-then-discretise (approach)* hence they vary in the order of deriving the necessary condition for an optimum, called the *optimality system*, before or after the problem is discretised.[1] The focus of this chapter

[1]The two strategies *first-discretise-then-optimise* and *first-optimise-then-discretise* are also known as *first-discretise-then-differentiate* and *first-differentiate-then-discretise*, cf. e.g. Gunzburger [**56**]. This nomenclature is rooted in whether the derivatives needed to formulate an optimality system is derived based on a discrete or continuous optimisation problem formulation.

is brought on the adoption, investigation and application of methods to the first-discretise-then-optimise approach while the next chapter, Chapter 5, follows the strategy of first-optimise-then-discretise.

The first-discretise-then-optimise approach leads to optimal control and optimisation problems each with an LB equation being the part and parcel of the constraints. Such problems were investigated before. Pingen et al. consider topology and design optimisation problems in [**108, 109, 110, 111**] and Tekitek et al. dedicate their work [**131**] to parameter identification problems. The main aim of this chapter is to propose, investigate and finally apply strategies to solve optimal flow control and optimisation problems of the type given in (4.1). Whereby, the proposed framework should allow an implementation which is highly generic in the sense that a wide range of fluid flow optimisation and control problems can be solved without changing the source code. The key to reach that goal is the adoption of automatic differentiation techniques which consequently become the primary part of the proposed overall concept. Mostly, the proposed approach relies on several methods which have been proposed before for similar flow control up to general optimisation problems as they are classified and discussed in the literature, e.g. in Lions [**93**], Tröltzsch [**133**] and Gunzburger [**56**].

The remainder of this chapter is organised as follows. At first, in Section 4.1 the proposed overall strategy to solve the here considered flow optimisation problems numerically is stated. It is substantiated in detail in the following sections whereas the main emphasis is placed on the adoption of automatic differentiation techniques. Thereunto, in Section 4.2 two different strategies are introduced. Afterwards, in Section 4.3 details regarding the implementation of the approach for parallel use are discussed. Then, in the following section the approach is tested. For the purpose of validation the results of two fluid flow optimisation problems are discussed. The first one is a family of parameter identification problems and the second one is a distributed control problem. Finally, the last section of this chapter is dedicated to discuss the obtained performance results for both the sequential and the parallel implementation by means of considering again the two examples.

4.1. From the Optimisation Problem Formulation to the Numerical Solution: A Sensitivity-based Strategy

In the following, an overall strategy to numerically solve a given constraint optimisation problem which is formulated as in (4.1) is proposed. The approach consists of four main steps:

(1) **Discretisation** to obtain a discrete formulation of the considered constrained optimisation problem,
(2) **Reformulation** of the discrete problem as unconstrained optimisation problem,
(3) **Deriving an optimality condition** for an optimum of the reformulated problem,
(4) **Solving the optimisation problem** by means of an iterative gradient-based method using automatic differentiation (AD) to obtain the gradients.

All steps of the strategy are presented in detail within the remainder of this section except for one aspect of the fourth step, namely the application of AD methods to obtain the needed derivatives. Since the application of AD techniques play a major role in its actual realisation, a separate subsection is dedicated to discuss them in more detail. Furthermore, using AD in this context makes the main difference to other, already proposed approaches to solve certain optimisation problems with an underlying mesoscopic-type equation as part of the constraints. In the work of Pingen et al. [**108, 109, 110, 111**] and Tekitek et al. [**131**] the gradients needed to solve the optimality condition are determined by means of an adjoint-based approach. All these proposed approaches have in common that a certain dual problem needs to be determined. Hereby, usually the structure of the problem differs with respect to the considered optimisation problem class. This is in contrast to the here proposed strategy where the usage of AD techniques provides a framework to obtain the derivatives generically.

4.1.1. First Step: Discretisation. Starting with the continuous problem formulation (4.1) in this step the problem is to be rewritten as a discrete constraint flow control or optimisation problem. In Chapter 2 it is shown that BGK-Boltzmann equations can be substituted consistently by LB equations. Thereby, the underlying space $I \times \Omega \times \mathbb{R}^d$ is discretised for a given discretisation parameter $h \in \mathbb{R}_{>0}$ by $I_h \times \Omega_h \times Q$. Since a BGK-Boltzmann equation is part and parcel of the side condition $\boldsymbol{G}(f, \boldsymbol{\alpha}) = \boldsymbol{0}$ and the state f acts on $I \times \Omega \times \mathbb{R}^d$ a discretisation strategy which is an extension of the previous one is advisable and therefore proposed. Thus, an LB equation like (2.15) becomes the discrete governing equation of the underlying fluid flow problem of the considered optimisation problem. The state f is replayed by its discrete pendant f_i. For the control $\boldsymbol{\alpha}$ which is a function in a Hilbert space U a second discretisation parameter $n \in \mathbb{N}$ is introduced. If the control space $U =: U_n$ is already a finite dimensional space n is given by the space dimension of U. Otherwise the control space U is replaced by a n-dimensional Hilbert space U_n. The objective J and the rest of the constraint G are substituted by functions $J_{h,n}$ and $G_{h,n}$ which are chosen appropriately with respect to the discrete space $I_h \times \Omega_h \times Q$ and U_n. This leads to the following abstract formulation of the derived discrete flow control or optimisation problem:

$$\left.\begin{array}{l}\text{find control } \boldsymbol{\alpha_n} \text{ and state } f_i \text{ which} \\ \text{minimise } J_{h,n}(f_i, \boldsymbol{\alpha_n}) \text{ and fulfill } \boldsymbol{G_{h,n}}(f_i, \boldsymbol{\alpha_n}) = \boldsymbol{0} \ .\end{array}\right\} \tag{4.2}$$

4.1.2. Second Step: Reformulation. In this step the discrete problem (4.2) is to be rewritten as an unconstrained optimisation problem. Therefore, in the following the state f_i is understood as a function of $\boldsymbol{\alpha_n}$ which is implicitly defined by the side condition $\boldsymbol{G_{h,n}}(f_i, \boldsymbol{\alpha_n}) = \boldsymbol{0}$. Provided that such a function f_i exists the constrained optimisation problem (4.2) is rewritten equivalently as an unconstrained optimisation problem:

$$\left.\begin{array}{l}\text{find control } \boldsymbol{\alpha_n} \text{ which} \\ \text{minimise } J_{h,n}(f_i(\boldsymbol{\alpha_n}), \boldsymbol{\alpha_n}) \ .\end{array}\right\} \tag{4.3}$$

In the next step a necessary condition for a minimum of this problem will be derived.

4.1.3. Third Step: Deriving an Optimality Condition. Assuming that $J_{h,n}$ is totally differentiable for all $\boldsymbol{\alpha_n} \in U_n$, a necessary condition for a control $\boldsymbol{\alpha^*_n}$) which minimises $J_{h,n}(f_i(\boldsymbol{\alpha^*_n}), \boldsymbol{\alpha^*_n})$ is

$$\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i(\boldsymbol{\alpha^*_n}), \boldsymbol{\alpha^*_n}) = \boldsymbol{0} \ . \tag{4.4}$$

This condition for a minimum of the optimisation problem (4.3) is called a *(discrete) optimality system.* Usually, a closed expression of the system cannot be directly determined. This is due to the high complexity of the LB equation and with it of the constraints so that in general solving the inverse problem to determine f_i analytically becomes an impossible task. Yet, starting with the constrained problem formulation (4.2) and applying the Lagrange formalism leads to an optimality condition in a closed form. This was illustrated by Pingen et al. [**108, 109, 110, 111**] and Tekitek et al. [**131**] for several particular problems, namely for topology and design optimisation and parameter identification problems.

4.1.4. Fourth Step: Solving the Optimisation Problem. As pointed out in the last step the here followed path does not directly lead to an optimality system which is given analytically in a closed form. Thus, methods solving the optimality system directly, known as *one-shot methods*, cannot be applied to solve a considered optimisation problem numerically. Instead, iterative optimisation algorithms which numerically solve the actual optimisation problem (4.2) or just the derived optimality condition (4.4) can be applied. An overview about the variety of optimisation methods can be found in the literature, e.g. in [**104**] or dedicated for unconstrains problems in [**44**]. Among the methods which can be applied to solve non-linear optimisation problems one finds the sub-class of *line search methods.* A prototypical structure of a line search algorithm is given with Algorithm 3.

Algorithm 3 Line search

```
Set $\boldsymbol{\alpha_n^0} = \boldsymbol{\alpha_{n,0}}$ //Initial guess for control variable
Set $k = 0$
**while** Stop condition not fulfilled **do**
  1. Compute the descent direction $\boldsymbol{d^k}$
  2. Compute step length $\delta^k$
  3. Set $\boldsymbol{\alpha_n^{k+1}} = \boldsymbol{\alpha_n^k} + \delta^k \boldsymbol{d^k}$
  4. Set $k = k + 1$
```

Methods of this class can in turn be classified with respect to two main aspects. The first one is how for every single optimisation step $k = 1, 2, ...$ the *descent direction* $\boldsymbol{d^k}$ is computed. And the second distinguishing aspect is the way the *step length* δ^k is determined. Beside these two main aspects the freedom of the choice for an initial guess for the control variable $\boldsymbol{\alpha_n^0} = \boldsymbol{\alpha_{n,0}}$ and an adequate stop criterion are to be considered.

In the following, solely such line search methods are considered which require the evaluation of the goal functional $J_{h,n}$ and its total derivative $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}$ to determine the descent direction $\boldsymbol{d^k}$ and the step length δ^k $(k = 1, 2, ...)$. This restriction is rather due to practical reasons because the evaluation of higher order derivatives incorporates higher computational costs. However, in principle the proposed approach offers strategies to obtain higher derivatives.

The sub-class of line search methods which demand the evaluation of the derivatives is known under the term *gradient-based optimisation methods.* Among them, one finds e.g. the *steepest descent method* where $\boldsymbol{d^k} := -\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})$ and the *BFGS method* with $\boldsymbol{d^k} := -H_k^{-1} \frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})$. The later one is named after its inventors Broyden, Fletcher, Goldfarb and Shanno and is a *quasi-Newton method.* The idea is to approximate the inverse of the Hessian of $J_{h,n}$ in every single optimisation step k $(k = 1, 2, ...)$ by H_k^{-1} which is obtained iteratively requiring H_{k-1}^{-1}, $\boldsymbol{\alpha_n^k}$, $\boldsymbol{\alpha_n^{k-1}}$, $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})$ and $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i(\boldsymbol{\alpha_n^{k-1}}), \boldsymbol{\alpha_n^{k-1}})$. As examples for suitable step length strategies, the *Armijo rule* and the *Wolfe-Powell rule* are to be mentioned (cf. e.g. [**44**]).

The evaluation of the goal functional $J_{h,n}$ for given $\boldsymbol{\alpha_n^k}$ requires at first solving the constraints $\boldsymbol{G_{h,n}}(f_i, \boldsymbol{\alpha_n}) = \mathbf{0}$ to obtain $f_i(\boldsymbol{\alpha_n^k})$) which is substantially solving an LB equation in every optimisation step $k = 1, 2, ...$ This can be done as illustrated in Chapter 1. Then, it is an easy task to compute $J_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})$.

The last question which remains to be answered is how can the total evaluated derivatives $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})$ be obtained. Regarding this question the proposed approach differs to that of Pingen et al. [**108, 109, 110, 111**] and Tekitek et al. [**131**]. They derive the adjoint equation dedicated for each of their considered problems and solve the obtained linear systems by e.g. in [**109**] a Schur-complement method. Here however, the application of techniques provided in the framework of AD is proposed. The following section is dedicated to consider in detail selected AD methods which can be applied to obtain the needed evaluated gradients.

4.2. The Gradient of the Goal Functional by Automatic Differentiation

Automatic differentiation (AD), also referred to as *alogrithmic differentiation* (cf. [**53**]), is a family of techniques to obtain numerically evaluated derivatives of a given function which is specified in form of a computer program. The underlying basic idea is to consider the evaluation process of a function as a sequence of elementary operations. An AD tool provides, e.g. in a table, for each different elementary operation its corresponding analytical derivative. The wanted evaluated derivative of the considered function is elementary operation by elementary operation composed by combining the intermediate data of the function evaluation process in accordance with the differentiation rule for the actual elementary operation provided by AD and also in accordance with the chain rule.

AD techniques are conceptually different to those of the fields symbolic differentiation and finite differences. The different scope of AD and symbolic differentiation techniques results in the adoption of different concepts. The main objective of AD is to obtain evaluated derivatives. This is in contrast to symbolic differentiation which aims to derive an as simple as possible closed analytical expression for a derivative of a target function. With finite differences approaches the derivative is approximated and evaluated while with AD techniques always the exact analytical derivative is evaluated. Considering the total numerical error, for both, AD and finite differences techniques, rounding errors for the evaluation of the function must be considered. However, an additional approximation error has to be taken into account only for finite differences approaches. The naturally arising question which of the three methods is preferable for the here considered optimisation problems is not directly in the focus of this work. Yet, later in this chapter numerical and performance results which are obtained by applying an AD technique are presented. With it a first step is done for a practical comparison.

The remainder of this section is dedicated to give a formal description of the problem of automatically obtaining an evaluated Jacobian of a function which is given by a computer program. Based on that, two strategies are derived which will lead to the AD techniques called the forward and backward mode. Finally, the two strategies are discussed with particular regard to their application to fluid flow control and optimisation problems with LBM. The presented framework solely considers first-order derivatives. Yet, the strategies can be applied in turn to derivatives to obtain higher-order derivatives.

4.2.1. From a Formal Description of Automatic Differentiation Problems to Solution Strategies. The following formal derivation is an extension of the one stated by Bendtsen et al. in [**18**] in terms of genericity and detailedness. The function which is to be differentiated is defined by

$$\boldsymbol{g} : \begin{cases} A \subseteq \mathbb{R}^n & \to \mathbb{R}^m \\ x & \mapsto \boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \ . \end{cases}$$

It is assumed that $\boldsymbol{g}$ is given as a composition of $l+m+n \in \mathbb{N}$ elementary operations $\tau_i \in \mathcal{T}$ $(i = 1, ..., l+m+n)$ which are differentiable. For any fixed $\boldsymbol{x} \in A$ the decomposition reads as follows:

$$\left.\begin{aligned} z_1 &:= \tau_1 :\equiv x_1 \\ z_2 &:= \tau_2 :\equiv x_2 \\ z_3 &:= \tau_3 :\equiv x_3 \\ &\vdots \\ z_n &:= \tau_n :\equiv x_n \end{aligned}\right\}$$
$$\left.\begin{aligned} z_{n+1} &:= \tau_{n+1}(z_1, z_2, ..., z_n) \\ z_{n+2} &:= \tau_{n+2}(z_1, z_2, ..., z_{n+1}) \\ &\vdots \\ z_{n+l} &:= \tau_{n+l}(z_1, z_2, ..., z_{n+l-1}) \end{aligned}\right\}$$
$$\left.\begin{aligned} y_1 = g_1(\boldsymbol{x}) \ = z_{n+l+1} &:= \tau_{n+l+1}(z_1, z_2, ..., z_{n+l}) \\ y_2 = g_2(\boldsymbol{x}) \ = z_{n+l+2} &:= \tau_{n+l+2}(z_1, z_2, ..., z_{n+l+1}) \\ &\vdots \\ y_m = g_m(\boldsymbol{x}) = z_{n+l+m} &:= \tau_{n+l+m}(z_1, z_2, ..., z_{n+1+m-1}) \end{aligned}\right\}$$

The first n elementary operations are assignments. They are introduced fictitiously to simplify the derivation to come. The other next $m+l$ operations are given in a fixed order by e.g. a computer program. The vector $\boldsymbol{z} \in \mathbb{R}^{l+m+n}$ aggregates the input data $\boldsymbol{x}$ at the first n positions, afterwards the intermediate results and at the last m position the output data $\boldsymbol{y}$. The *elementary operations* τ_i $(i = 1, ..., l+m+n)$ are stated with a different number of arguments. Although, usually only one or two arguments are active, i.e. $\tau_i(z_1, z_2, ..., z_{i-1}) = \tau_i(a, b)$ or $\tau_i(z_1, z_2, ..., z_{i-1}) = \tau_i(a)$ holds for $i = n+1, n+2, ..., l+m+n$ where $a, b \in \{z_1, z_2, ..., z_{i-1}\}$. If one argument is active the elementary operation is said to be a *unary* and if two are active a *binary elementary operation*. Depending on the programming language which is used the available set of elementary operations $\mathcal{T} = \mathcal{T}_{binary} \cup \mathcal{T}_{unary}$ differs. An incomplete example for typical operations is given as follows

- Set of binary elementary operations:
  $\mathcal{T}_{binary} = \{\tau(a, b) = a + b, a * b, a - b, a/b, ...\}$
- Set of unary elementary operations:
  $\mathcal{T}_{unary} = \{\tau(a) = a, -a, 1/a, a + c, a * c, sin(a), cos(a), exp(a), ... \ : c \in \mathbb{R}\}$

The chain rule applied to the composition of elementary operations yields

$$\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1}) = \begin{cases} 0 & : \text{ if } i < j \\ 1 & : \text{ if } i = j \\ \displaystyle\sum_{k=j}^{i-1} \frac{\partial \tau_i}{\partial z_k}(z_1, z_2, ..., z_{i-1}) \frac{d\tau_k}{dz_j}(z_1, z_2, ..., z_{k-1}) & : \text{ if } i > j \ . \end{cases} \tag{4.5}$$

Then, with the definitions

$$\boldsymbol{A} = (a_{ij})_{1 \le i,j \le l+m+n} \ , \ a_{ij} := \begin{cases} 0 & : \text{ if } i < j \\ 1 & : \text{ if } i = j \\ \dfrac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1}) & : \text{ if } i > j \ , \end{cases}$$

$$\boldsymbol{B} = (b_{ij})_{1 \le i,j \le l+m+n} \ , \ b_{ij} := \begin{cases} 0 & : \text{ if } i < j \\ 0 & : \text{ if } i = j \\ \dfrac{\partial\tau_i}{\partial z_j}(z_1, z_2, ..., z_{i-1}) & : \text{ if } i > j \end{cases}$$

the relation (4.5) is rewritten equivalently in matrix notation

$$\boldsymbol{A} = \boldsymbol{I}_{l+m+n} + \boldsymbol{B}\boldsymbol{A} \ . \tag{4.6}$$

The intermediate results $\boldsymbol{z}$ can be obtained easily by evaluating the function g. Just as facile one gets the partial derivatives of the elementary operations τ_i ($i = 1, 2, ..., l+m+n$) by e.g. looking them up in a table. Finally, gathering both information, the evaluated derivatives $\frac{\partial\tau_i}{\partial z_j}$ $(z_1, z_2, ..., z_{i-1})$ can be obtained. Thus, isolating the matrix $\boldsymbol{A}$ in the linear system (4.6) and solving the transformed system leads to numerically feasible strategies to obtain the wanted derivatives. Therefore, at first the following equivalent transformations are performed to isolate $\boldsymbol{A}$:

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{I}_{l+m+n} + \boldsymbol{B}\boldsymbol{A} & \Leftrightarrow \\ (\boldsymbol{I}_{l+m+n} - \boldsymbol{B})\,\boldsymbol{A} &= \boldsymbol{I}_{l+m+n} & \Leftrightarrow \qquad (4.7) \\ \boldsymbol{A} &= (\boldsymbol{I}_{l+m+n} - \boldsymbol{B})^{-1} & \Leftrightarrow \\ \boldsymbol{A}\,(\boldsymbol{I}_{l+m+n} - \boldsymbol{B}) &= \boldsymbol{I}_{l+m+n} & \Leftrightarrow \\ (\boldsymbol{I}_{l+m+n} - \boldsymbol{B})^T\,\boldsymbol{A}^T &= \boldsymbol{I}_{l+m+n} & \qquad (4.8) \end{aligned}$$

Two linear systems (4.7) and (4.8) are obtained where the later one is the adjoint equation of the prior one. The system (4.7) can be solved by forward substitution as stated in Algorithm 4. With backward substitution the dual system (4.8) can

Algorithm 4 Automatic differentiation: Simple forward mode

```
for $i = 1$ to $i = l + m + n$ do
    Compute $z_i = \tau_i(z_1, z_2, ..., z_{i-1}) = \tau_i(z_{k_1}, z_{k_2})$
    Compute $t_1 = \frac{\partial\tau_i}{\partial z_{k_1}}(z_{k_1}, z_{k_2})$ by e.g. look up table
    Compute $t_2 = \frac{\partial\tau_i}{\partial z_{k_2}}(z_{k_1}, z_{k_2})$ by e.g. look up table
    Set $\frac{d\tau_i}{dz_i} \equiv 1$
    for $j = 1$ to $j = i - 1$ do
        Set $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1}) = t_1 \frac{d\tau_{k_1}}{dz_j}(z_1, z_2, ..., z_{k_1-1}) + t_2 \frac{d\tau_{k_2}}{dz_j}(z_1, z_2, ..., z_{k_2-1})$
```

be solved. This leads to a scheme alike the one presented in Algorithm 5.

4.2.2. Forward and Reverse Accumulation. Both schemes, Algorithm 4 and Algorithm 5, are not efficient in terms of memory consumption. And further, not all $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$ $(1 \le j < i \le l+m+n)$ are needed to determine the derivatives of the actually considered function g. Taking these issues into account both schemes can be improved. For the forward substitution strategy this optimisation leads to the AD techniques which are known as *forward modes*. Respectively, for the backward substitution strategy the AD techniques called *reverse* or *backward modes* are obtained. For both alternatives an example is given, one in Algorithm 6

Algorithm 5 Automatic differentiation: Simple reverse mode

```
for $i = 1$ to $i = l + m + n$ do
  Compute $z_i = \tau_i(z_1, z_2, ..., z_{i-1})$
  for $j = 1$ to $j = i - 1$ do
    Compute and store $\frac{\partial \tau_i}{\partial z_j}(z_1, z_2, ..., z_{i-1})$ by e.g. look up table
for $j = l + m + n$ to $j = 1$ do
  Set $\frac{d\tau_j}{dz_j} \equiv 1$
  for $i = l + m + n$ to $i = j + 1$ do
    Set $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1}) = \sum_{k=j+1}^{i} \frac{\partial \tau_k}{\partial z_j}(z_1, z_2, ..., z_{k-1}) \frac{d\tau_i}{dz_k}(z_1, z_2, ..., z_{i-1})$
```

for a forward mode scheme and another in Algorithm 7 for a reverse mode scheme.

Algorithm 6 Automatic differentiation: Forward mode

```
for $i = 1$ to $i = n$ do
  Set $z_i = x_i$
  for $j = 1$ to $j = n$ do
    if $i = j$ then
      Set $\frac{d\tau_i}{dz_i} \equiv 1$
    else
      Set $\frac{d\tau_i}{dz_j} \equiv 0$
for $i = n + 1$ to $i = l + n + m$ do
  Compute $z_i = \tau_i(z_1, z_2, ..., z_{i-1}) = \tau_i(z_{k_1}, z_{k_2})$
  Compute $t_1 = \frac{\partial \tau_i}{\partial z_{k_1}}(z_{k_1}, z_{k_2})$ by e.g. look up table
  Compute $t_2 = \frac{\partial \tau_i}{\partial z_{k_2}}(z_{k_1}, z_{k_2})$ by e.g. look up table
  for $j = 1$ to $j = n$ do
    Set $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1}) = t_1 \frac{d\tau_{k_1}}{dz_j}(z_1, z_2, ..., z_{k_1-1}) + t_2 \frac{d\tau_{k_2}}{dz_j}(z_1, z_2, ..., z_{k_2-1})$
    if $i > n + l$ then
      Set $\frac{dg_{i-l-n}}{dx_j}(x_1, x_2, ..., x_n) = \frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$
  if $i > n + l$ then
    Set $y_{i-n-l} = z_i$
```

The here presented forward mode algorithm, namely Algorithm 6, is rooted in the forward substitution scheme which is stated in Algorithm 4. An essential difference is that the loop variable j runs to n instead of $i - 1$. With it, not all $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$ $(1 \le j < i \le l + m + n)$ are determined but only the ones which are needed to obtain the Jacobian of g. Further, interfaces are added to initialise the *dependent variables* $\boldsymbol{x}$ at the beginning and to obtain the Jacobian of g at the end. It is to be noted that the partial derivatives $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$ $(n < i \le l + m + n, 1 \le j \le n)$ do not all have to be stored. Instead, for every $i = n + 1, n + 2, ..., l + m + n$ only two, namely $\frac{\partial \tau_i}{\partial z_{k_1}}(z_{k_1}, z_{k_2})$ and $\frac{\partial \tau_i}{\partial z_{k_1}}(z_{k_1}, z_{k_2})$ $(1 \le k_1, k_2 \le n)$, are determined and stored just temporarily for the actual step i. In a considered program storage space is provided in form of various variables where the intermediate results z_i $(i = 1, 2, ..., l + m + n)$ can be stored. Usually, most of the intermediate results are just stored temporarily because they are only needed for a few of the following elementary operations. Thus, many variables are reused or destructed, i.e. the dedicated memory is used to store other results. This fact can be exploited to save memory. Any time the memory which is dedicated for an intermediate result z_i is freed, the memory dedicated for $\frac{df_{i-l-n}}{dx_j}(x_1, x_2, ..., x_n)$

for all $j = 1, 2, ..., n$ can be freed as well.

In Algorithm 7 an example for a reverse mode scheme is given. The presented algorithm is similar to that which is labelled in [**53**] as *non-incremental adjoint recursion approach*. Beside this approach, others exist (cf. e.g. [**53**]) which share many of the characteristics of the one presented here. In the given example it is illustrated that for each elementary operation $\tau_i(z_1, z_2, ..., z_{i-1}) = \tau_i(z_{k_{i,1}}, z_{k_{i,2}})$ $i = n+1, n+2, ..., l+n+m$ the up to two indices $k_{i,1}, k_{i,2}$ of the active variables as well as the corresponding evaluated partial derivatives $\frac{\partial \tau_i}{\partial z_{k_{i,1}}}(z_{k_{i,1}}, z_{k_{i,2}})$ and $\frac{\partial \tau_i}{\partial z_{k_{i,1}}}(z_{k_{i,1}}, z_{k_{i,1}})$ needed to be stored. This is due to the reverse order of the loop in which the total derivatives are computed afterwards. As in the forward mode scheme not all $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$ $(1 \leq j < i \leq l+m+n)$ are needed to be determined. Here the loop variable i can run from $l+m+n$ down to $l+n+1$ instead to $j+1$ to get the Jacobian of g.

Algorithm 7 Automatic differentiation: Reverse mode

```
for $i = 1$ to $i = n$ do
  Set $z_i = x_i$
for $i = n + 1$ to $i = l + n + m$ do
  Compute $z_i = \tau_i(z_1, z_2, ..., z_{i-1}) = \tau_i(z_{k_{i,1}}, z_{k_{i,2}})$
  Store $k_{i,1}, k_{i,2}$
  Compute and store $\frac{\partial \tau_i}{\partial z_{k_{i,1}}}(z_{k_{i,1}}, z_{k_{i,2}})$ by e.g. look up table
  Compute and store $\frac{\partial \tau_i}{\partial z_{k_{i,1}}}(z_{k_{i,1}}, z_{k_{i,1}})$ by e.g. look up table
  if $i > n + l$ then
    Set $y_{i-n-l} = z_i$
for $j = l + m + n$ to $j = 1$ do
  Set $\frac{d\tau_j}{dz_j} \equiv 1$
  for $i = l + m + n$ to $i = l + n + 1$ do
    Set $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1}) = 0$
    for $k = j + 1$ to $k = i$ do
      if $j = k_{k,1}$ or $j = k_{k,2}$ then
        Set $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1}) += \frac{\partial \tau_k}{\partial z_j}(z_1, z_2, ..., z_{k-1}) \frac{d\tau_i}{dz_k}(z_1, z_2, ..., z_{i-1})$
    if $i > n + l$ and $j \leq n$ then
      Set $\frac{dg_{i-l-n}}{dx_j}(x_1, x_2, ..., x_n) = \frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$
```

4.2.3. Forward or Reverse Mode for Lattice Boltzmann Fluid Flow Optimisation Problems: A Discussion. Which of the two modes are preferable depends on the considered function $\boldsymbol{g}$ itself. If n is much greater than m the computational costs to obtain the Jacobian with a forward scheme will be greater than those of a backward scheme and vice versa. This is due to the fact that one of the two loops in a backward mode has m iterations and in a forward mode n. A second aspect which is important to consider is the number of elementary operations $l+m+n$ of the function $\boldsymbol{g}$. Considering backward mode schemes, the massive need of memory is a critical issue for problems with a high number of elementary operations $l+m+n$ because many intermediate results are needed to be stored. This is in contrast to forward mode schemes where most of the intermediate results are stored temporarily and memory is reused. *Checkpointing strategies* are proposed in e.g. [**52**] to reduce the required storage space on cost of additional computational costs.

For optimisation problems in general, the considered function $\boldsymbol{g}$ which is to be differentiated is the goal functional and hence, m is always equal to 1. And n is given as the dimension of the finite-dimensional control space. Depending on the problem, n can but does not need to be much greater than 1. Considering fluid flow control and optimisation problems, usually and especially for three dimensional problems, the number of elementary operations $l + m + n$ needed to evaluate g is high. With LB schemes being the part and parcel of the constraints this is even more true. The explicit character of LB schemes imply an iteration over several thousand time steps and therefore billions elementary operations. To give an example, an efficient implemented $D3Q19$ model requires roughly 200 operations at each cell of the lattice in every single time step (cf. [**139**]). Considering a fluid flow problem with 100^3 cells and $100,000$ time steps one gets about $2 \cdot 10^{13}$ elementary operations just to solve the side condition. Neglecting all other computational costs and assuming that 24 bytes of memory are needed for any operation one gets in total roughly half a petabyte of required storage space then applying a reverse mode scheme. Even if the demanded memory is available the time required to access the data becomes significantly high so that backward mode schemes seem not to be attractive in this framework. Yet, *Parallel I/O* in combination with *checkpointing strategies*, as they are investigated by Bockelmann et al. in [**21**] to overcome similar mass storage problems, might be a possible realistic solution also for AD backward mode schemes. Research is required to give a clear answer to the question which of the two AD strategies is preferable for LB-based fluid flow optimisation problems. As a start in the following, forward mode schemes are investigated in detail.

4.3. Implementation of a Generic Parallel Lattice Boltzmann Fluid Flow Optimisation Problem Solver

This section is devoted to consider the realisation of strategies to numerically solve fluid flow control and optimisation problems as given as in (4.1) both in general and specifically for the strategy which is proposed in the previous two sections. Thus, this section is split into two parts. In the first part, an implementation concept is introduced which is general enough to cope with a wide range of approaches to solve optimisation problems. Among them, one finds the two approaches which are presented in the current and in the following chapter. The realisation of the proposed concept is illustrated by means of an example, namely the extension of the LB-based fluid flow simulation software OpenLB (cf. Appendix A) towards a tool to solve also fluid flow control and optimisation problems. The second part of this section is directly connected to the first one. It considers as a specialisation the realisation of the strategy which is presented in this chapter. Whereby, the main emphasis is placed on details concerning the realisation of the AD forward mode approach. Thereby, special care is taken to obtain an efficient parallel implementation.

4.3.1. A General Optimisation Software Framework. In Appendix A the open source package OpenLB is introduced. It is indicated that the C++ code OpenLB is programed in an object-oriented and template-based style. Taking advantage of both the dynamic and static genericity a parallel solver for various fluid flow control and optimisation problems can be obtained. In Figure 4.1 the proposed software design is pictured as it is realised in OpenLB.

The linchpin of the concept proposed in the previous two sections and also of many other strategies is an iterative optimisation scheme such as the line search optimisation algorithm stated in Algorithm 3. With it, the outermost loop of the

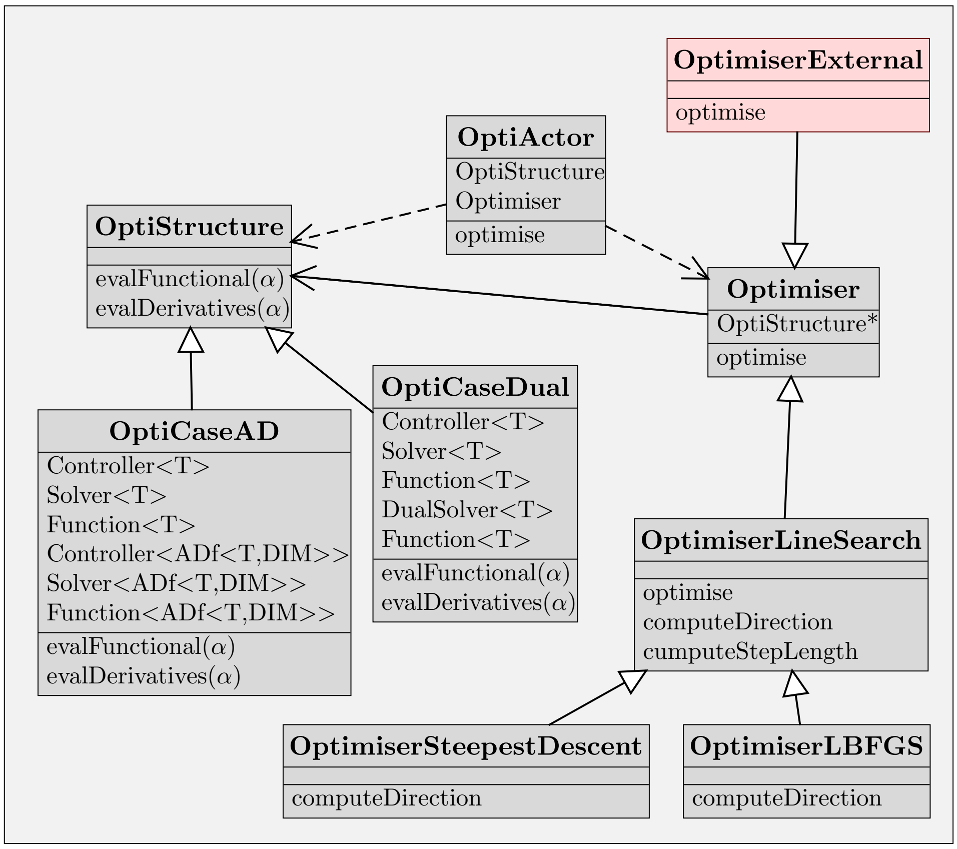


FIGURE 4.1. Software design for a generic fluid flow optimisation problem solver illustrated by the class hierarchy in OpenLB. The arrow $\longrightarrow\!\!\triangleright$ indicates a connection of two classes by inheritance with the head pointing to the base class. $- - >$ points towards a class which holds an instant of the connected class and likewise $-\longrightarrow$ which points towards one class holding a pointer of another.

overall solution process is given. Which data are needed in every optimisation step $k = 1, 2, ...$ depends on the applied optimisation method. For line search algorithms usually the evaluated goal functional $J_{h,n}(f_i, \boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}})$ and its total derivative $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i, \boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}})$ are required. In general also higher-order derivatives of $J_{h,n}$ may be required. In OpenLB the abstract class called `OptiStructures` provides this functionality. It serves as an interface to subclasses where different strategies are realised to obtain the required data. For instance, in the subclass `OptiCaseDual` the strategy which is described in the following chapter is realised. Further, in the subclass `OptiCaseAD` the approach proposed in the previous two sections is implemented. Also by inheritance the implementation of various optimisation algorithms is enabled. The base class `Optimiser` serves as well as an interface. It offers to make use of external software package for that the wrapper subclass `OptimiserExternal` is provided. Further, in another subclass of `Optimiser` called `OptimiserLineSearch` the outermost loop of Algorithm 3 is implemented including a stop criterion and the Wolfe-Powell rule to determine the step length. Two specialisations, namely the class `OptimiserLBFGS` for an LBFGS scheme and `OptimiserSteepestDescent` for an steepest descent scheme complete the hierarchy of inheritance. On top of all other classes the class `OptiActor` organises the composition of two compatible specialisation that is one of `OptiStructures` and the other of `Optimiser`.

4.3.2. Realisation of a Forward Mode Scheme for Parallel Use. In the first part of this section a general framework to realise various strategies to numerically solve optimisation problems is introduced. Now, a particular strategy, namely the one proposed within this chapter, is considered. The aim of this part is to address details regarding its implementation which enables an efficient parallel execution. Hereby, the key issue is the realisation of the forward mode scheme presented in Algorithm 6 which enables to obtain the needed evaluated derivatives. As well as before the realisation is illustrated by an example that is the implementation in the open source library OpenLB.

In the terminology introduced in the first part of this section the considered strategy is realised in form of a derived class called `OptiCaseAD` as specialisation of `OptiStructure`. The functionalities which must be provided by `OptiCaseDual` are the evaluation of the functional $J_{h,n}$ and of its total derivative $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}$ for a given $\boldsymbol{\alpha_n^k}$ which is determined in every optimisation step $k = 1, 2, ...$ by a line search optimisation scheme.

The first task, evaluating $J_{h,n}$ for a given $\boldsymbol{\alpha_n^k}$, is split into two main steps. At first the side condition $\boldsymbol{G_{h,n}}(f_i, \boldsymbol{\alpha_n^k}) = \mathbf{0}$ is solved numerically for $\boldsymbol{\alpha_n^k}$ to obtain the state f_i. Then, with the result f_i and the given control $\boldsymbol{\alpha_n^k}$ the goal functional is evaluated. For many applications solving the side condition means basically solving a problem with an LB equation as governing equation where $\boldsymbol{\alpha_n^k}$ just enters as a parameter. In OpenLB this is realised by introducing a class called `Controller<T>` in which the controlled variables $(\alpha_n^k)_j$ $(j = 1, 2, ..., n)$ are specifies and their current values are saved. Hereby, it is important to note that `Controller<T>` is a template class with template `T` which is now set to a floating-point data type like `double` or `float`. Then, depending on the control variables $\boldsymbol{\alpha_n^k}$ an LB problem is set-up and solved. This is done by a template class as well which is called `Solver<T>` which depends on `Controller<T>`. Finally, the goal functional $J_{h,n}$ is to be evaluated. It is specified in form of a member function of another template class with the name `Function<T>` which again depends on `Controller<T>`.

The second functionality which needs to be provided by the class `OptiCaseDual` is the computation of the evaluated total derivative of $J_{h,n}$ for a given control $\boldsymbol{\alpha_n^k}$. As discussed in Section 4.2 AD provides forward mode schemes to realise this task. In principle two main strategies can be distinguished to implement forward mode schemes like the one stated in Algorithm 6. In the literature, e.g. in [**53**], they are referred to as *program transformation* and *(operator) overloading approach.* The basic idea of the first one is to use pre-processor directives to generate an extended code which evaluates the wanted derivatives. Alternatively, the overloading approach makes use of polymorphism which is available in many object-oriented programming languages such as C++. A newly created data type substitutes the basic floating-point data type of a considered program which evaluates a functional while the actual program itself remains unchanged. However, all elementary operations for the new data type are defined just that beside the evaluated functional also the wanted evaluated derivatives are obtained. In OpenLB the basic floating-point data type is templatised, i.e. it can be substituted easily. Thus, the overloading approach seems very attractive concerning ease of realisation especially in the context of parallelism. It has been carried out in various libraries for example in FAD by Aubert et al. [**6**]), FADBAD by Bendtsen et al. [**18**] or ADOL-C by Griewank

```
template<typename T, unsigned DIM>
class ADf {

  private:
     /// value
     T _v;
     /// derivatives
     T _d[DIM];

  public:
    /// constructors
    ADf();
    ADf(const T& v);
    ADf(const ADf& a);
    /// operators
    ADf& operator *= (const ADf& a);
    ADf& operator *= (const T& v);
      .
      .
    /// initialises the iD-th dependent variable
    void setDiffVariable(unsigned iD);
    /// returns the value _v
    T& v();
    /// returns the derivative _d[i]
    T& d(unsigned i);
};
```

LISTING 4.1. The class `ADf<T,DIM>` in OpenLB enables to obtain `DIM` evaluated derivatives by operator overloading. With it, the AD forward mode scheme presented in Algorithm 6 is realised.

et al. [**54**]. Inspired by the mentioned three implementations in OpenLB a lean template-based overloading approach which enables parallelism is realised. It will be explained in detail in the following.

In Listing 4.1 the declaration of the class `ADf<T,DIM>` is stated in excerpts. It defines a new data type to realise the AD forward mode from Algorithm 6. Whereby, `DIM` represents the dimension n of the control space U_n and `T` the template for the basic floating-point data type. The variable `_v` corresponds to an intermediate result z_i, and in `_d` the total derivatives $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$ $(j = 1, 2, ..., n)$ are stored. With the member function `setDiffVariable` the control variables $(\alpha_n^k)_j$ $j = 1, 2, ..., n$ can be initialised according to the first loop of Algorithm 6. Every differentiable elementary operation which is defined for `T` also needs to be defined for `ADf<T,DIM>` to realise the scheme presented in Algorithm 6. As an example the source code of the implementation of the multiplication operators are stated is Listing 4.2. Due to the implicit type conversion in C++ [**127**] it is sufficient to implement the multiplication of `ADf<T,DIM>` with `ADf<T,DIM>`, `ADf<T,DIM>` with `T` and `T` with `ADf<T,DIM>`. With it, also the multiplication of `ADf<T,DIM>` with integral or other floating-point data types is well defined. The member functions `v` and `d` of the class `ADf<T,DIM>` allow the access to all intermediate and the final results. Now, with this newly defined data type `ADf<T,DIM>` the wanted evaluated derivative can be computed. Essentially, this is done as the evaluated functional itself is computed which is described above. Yet, `ADf<T,DIM>` instead of `T` is set as template parameter for the involved classes `Controller`, `Solver` and `Function`.

```
template <typename T, unsigned DIM>
ADf<T,DIM>& ADf<T,DIM>::operator *= (const ADf<T,DIM>& a)
{
  for (unsigned i=0;i<DIM;++i) {
    _d[i]*=a._v;
    _d[i]+=_v*a._d[i];
  }
  _v*=a._v;
  return *this;
}

template <typename T, unsigned DIM>
ADf<T,DIM>& ADf<T,DIM>::operator *= (const T& v)
{
  _v*=v;
  for (unsigned i=0;i<DIM;++i)
    _d[i]*=v;
  return *this;
}

template <typename T, unsigned DIM>
ADf<T,DIM> operator* (const T& a, const ADf<T,DIM>& b)
{
  return ADf<T,DIM>(b)*=a;
}

template <typename T, unsigned DIM>
ADf<T,DIM> operator* (const ADf<T,DIM>& a, const T& b)
{
  return ADf<T,DIM>(a)*=b;
}

template <typename T, unsigned DIM>
ADf<T,DIM> operator* (const ADf<T,DIM>& a, const ADf<T,DIM>& b)
{
  return ADf<T,DIM>(a)*=b;
}
```

LISTING 4.2. The implementation of the multiplication operators for the data type `ADf<T,DIM>` in OpenLB.

Further, an additional tagging of the control variables $(\alpha_n^k)_j$ $(j = 1, 2, ..., n)$ is necessary before the derivative is evaluated. This is realised by the `Controller<T>` which calls the member function `setDiffVariable` of the class `ADf<T,DIM>` every time the values of the controls are changed.

In template-based C++ programs it is commonplace to use the default constructor to initialise a templatised variable of fundamental data type with the value zero (cf. Stroustrup [**127**]). When implementing the default constructor of `ADf<T,DIM>` it is crucial to ensure that the default constructor of `T` is called to initialise _v correctly. This is necessary because the compiler-generated default constructor does not initialise member variables of fundamental types (cf. Stroustrup [**127**]). This issue is stressed because it is not taken care of in all above mentioned AD libraries.

Another important difference of the here proposed AD implementation to many others is that the memory for `_v` is allocated statically. The drawback of this approach is that the size `DIM` of `_v` must be known at compile time can be alleviated by providing the code for certain different `DIM` by e.g. a factory function. On the other hand, the approach promises to have also considerable advantages. Then allocating a vector of the type `ADf<T,DIM>` data locality in terms of `_v` to its corresponding `_d` is ensured. Depending on the underlying data structure of the considered problem this can result in higher performances because less data needs to be reloaded. Further, copying a vector of the type `ADf<T,DIM>` is facilitated. This is especially true when it comes to implement an AD scheme which enables parallel execution. In the next paragraph this issue is addressed in detail.

Basically, an efficient parallel computation of an evaluated derivative takes place as an efficient parallel computation of the evaluated function itself. This holds in general due to the structure of AD forward mode schemes because the data dependencies of the derivatives $\frac{d\tau_i}{dz_j}(z_1, z_2, ..., z_{i-1})$ $(j = 1, 2, ..., n)$ are similar to those of the intermediate results z_i (cf. Algorithm 6 and Griewank [**53**]). In the above proposed realisation this fact is taken care of in form of integrating the corresponding data within one variable of the created data type `ADf<T,DIM>`. For both tasks, the computation of the evaluated functional $J_{h,n}(f_i, \boldsymbol{\alpha}^{\boldsymbol{k}}_{\boldsymbol{n}})$ and its total derivative $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i, \boldsymbol{\alpha}^{\boldsymbol{k}}_{\boldsymbol{n}})$, usually it holds that solving the side condition is the most time demanding part of the whole process. Therefore, solely the parallelisation of this program part is considered in the following. In the here considered framework the first task consists of basically solving an LB problem. For such problems in Chapter 3 an efficient hybrid parallelisation strategy and its realisation are proposed, tested and discussed. Hence, solely the second task remains to be considered. Yet, it differs form the first one just in the underlying data type. However, the data dependencies remain the same. Thus, the same parallelisation concept which is presented in Chapter 3 can be applied. Solely the implementation needs to be extended because the involved parallelisation paradigm MPI in general and some implementations of OpenMP do not support the use of class data types like `ADf<T,DIM>`. The latter problem and possible solutions are discussed by Terboven et al. in [**132**]. Therefore, in OpenLB critical compiler directives like `#pragma omp threadprivate` for variables of class data types are avoided. For the MPI-based implementation templatised wrapper functions are introduced. They allow a specialised implementation for each employed data type and hence also for `ADf<T,DIM>`. Due to the static memory allocation of `_v` in `ADf<T,DIM>` a variable as well as a vector of variables of this type can be communicated in one single contiguous row of bytes. If the memory for `_v` would have been allocated dynamically a blocking of the data or several communication operations would be necessary. The blocking requires copying all data of all variables which are to be communicated to a contiguous space in the memory. Both loopholes would complicate the implementation and, even worse, would result in a loss of efficiency.

4.4. Numerical Experiments: A Parameter Identification and a Distributed Control Problem

In the following, two fluid flow optimisation problems are studied. In both examples the applied force $\boldsymbol{F}$ is to be controlled. The first one is a parameter identification problem, i.e. the control space is a finite dimensional space. This is in contrast to the second test case where the control space is not finite dimensional. However, after discretisation the control $\boldsymbol{\alpha}$ becomes discrete and hence the discrete problem formulations are akin each other. Both considered optimisation problems are related to the 3D stationary fluid flow problem discussed in Section 2.3. There, the distribution of the macroscopic velocity $\boldsymbol{u}$ is to be determined in Ω for a given force $\boldsymbol{F}$. Thereto, in Subsection 2.3.1 the fluid flow problem is formulated by means of a Navier-Stokes equation as governing equation. Hereby, the force $\boldsymbol{F}$ is given analytically according to (2.21). Then, as stated in Subsection 2.3.2, the problem is solved numerically by applying an LBM. This leads to a series of solutions f_i^h for different discretisation parameters $h \in \mathbb{R}_{>0}$ and, with it, to macroscopic velocities $\boldsymbol{u}_{\boldsymbol{f_i^h}}$ which approximate $\boldsymbol{u}$. Now, with the formulation of the two optimisation problems the questioning of this fluid flow problem is considered inversely, i.e. $\boldsymbol{u}$ or $\boldsymbol{u}_{\boldsymbol{f_i^h}}$, respectively, is assumed to be known and a suitable force $\boldsymbol{F}$ is to be determined.

The numerical experiments aim to, firstly, illustrate the before introduced concept and, secondly, test its realisation especially with respect to its performance. With it, a comparison of the proposed approach with other approaches is enabled. Interesting to note is that in Section 5.6 the approach which is studied in the following chapter is tested for exactly the same configuration. Further, a comparison of a similar approach, yet, with the Navier-Stokes equation instead the BGK-Boltzmann equation as governing equation becomes feasible.

4.4.1. Test Case Formulations. In the following, a set of four optimisation problems is considered, namely three parameter identification problems and one distributed control problem. All problems are formulated as restricted optimisation problems like (4.1). The three parameter identification problems consist of one with one control variable $\boldsymbol{\alpha} \in U_1 := \mathbb{R}$, another with three parameters $\boldsymbol{\alpha} \in U_3 := \mathbb{R}^3$ and finally a third one with eleven parameters $\boldsymbol{\alpha} \in U_{11} := \mathbb{R}^{11}$. For the considered distributed control problem the control $\boldsymbol{\alpha} \in U_\infty := \left(L^2(\Omega)\right)^3$ is a function. The corresponding four goal functionals J_n $(n = 1, 3, 11, \infty)$ are defined as

$$J_n : \begin{cases} H^1(\Omega \times \mathbb{R}^3) \times U_n & \to \mathbb{R} \\ (f, \boldsymbol{\alpha}) & \mapsto \dfrac{1}{2} \displaystyle\int_\Omega (\boldsymbol{u_f} - \boldsymbol{u}^*)^2 \, d\boldsymbol{r} + \dfrac{\alpha_{reg}}{2} \displaystyle\int_\Omega (\boldsymbol{B_n \alpha})^2 \, d\boldsymbol{r} \ . \end{cases} \tag{4.9}$$

Hereby, $\boldsymbol{u}^*$ is the target velocity distribution which is defined as in (2.23) and $\alpha_{reg} \in \mathbb{R}_{\geq 0}$ is a regularisation parameter. The linear continuous operators $\boldsymbol{B_n} \in \mathcal{L}(U_n, L^2(\Omega)^3)$ $(n = 1, 3, 11, \infty)$ denote control extension operators which will be defined explicitly later on. Applying these operators to the controls $\boldsymbol{\alpha} \in U_n$ one get the terms $\boldsymbol{B_n \alpha}$ $(n = 1, 3, 11, \infty)$. They constitute the force term $\boldsymbol{F}$ in the BGK-Boltzmann equations which are parts of the side conditions:

$$\boldsymbol{G_n^h} : \begin{cases} H^1(\Omega \times \mathbb{R}^3) \times U_n & \to H^1(\Omega \times \mathbb{R}^3) \times \left(L^2(\Gamma)\right)^3 \\ (f, \boldsymbol{\alpha}) & \mapsto \begin{pmatrix} h^2 \left(\boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{\boldsymbol{B_n \alpha}}{m} \cdot \boldsymbol{\nabla_v}\right) f - \frac{1}{3\nu} \left(f - M_f^{eq}\right) \\ \boldsymbol{u_f} - \boldsymbol{b} \end{pmatrix} \end{cases} , \tag{4.10}$$

where $\boldsymbol{b}$ is given by the velocity distribution function stated in (2.22).

Lets suppose that $\boldsymbol{F}^*$ is defined as in (2.21) and $\boldsymbol{\alpha}^* \in U_n$ satisfies $\boldsymbol{B_n \alpha}^*$ for one of the cases $n \in \{1, 3, 11, \infty\}$. Then, the corresponding side conditions (4.10) are nothing else than the fluid flow problem formulated and solved in Section 2.3. Thus, choosing the regularisation parameter $\alpha_{reg} = 0$, $J_n(f, \boldsymbol{\alpha}^*)$ is half the squared $L^2(\Omega)$-norm of the residuum $\boldsymbol{u_f} - \boldsymbol{u}^*$. Interesting to note is that $\boldsymbol{u_f}$ is the solution of the fluid flow problem from Section 2.3 formulated mesoscopically by a BGK-Boltzmann equation and $\boldsymbol{u}^*$ the solution of the same problem, yet, formulated macroscopically by the Navier-Stokes equation. In Subsection 2.3.3 it is observed that the residuum tends to zero if the discretisation parameter h tends to zero. From this property advantage will be taken later on when it comes to analysis the problems.

Left to be specified are the control extension operators. For the distributed control problem $\boldsymbol{B_\infty}$ is the identity map. For the parameter identification problems they constitute ansatz functions which are defined as

$$\boldsymbol{B_1}\alpha := \alpha\boldsymbol{F}\ ,\ \boldsymbol{B_3\alpha} := \begin{pmatrix}\alpha_1 F_1\\ \alpha_2 F_2\\ \alpha_3 F_3\end{pmatrix}$$

for any $\alpha \in U_1$ and $\boldsymbol{\alpha} \in U_3$, respectively. Further for any $\boldsymbol{\alpha} \in U_{11}$ and any $\boldsymbol{r} \in \Omega$ the ansatz function reads

$$\begin{aligned}
(B_{11}\alpha)_1(\boldsymbol{r}) := -\frac{1}{8}\pi\,(&\alpha_1 16\pi\nu\,(\cos(2\pi r_2)\cos(2\pi r_1) - \sin(2\pi r_1)\cos(2\pi r_3))\\
&-\alpha_1\cos(2\pi r_2)\cos(2\pi r_3) + 2\alpha_2\cos(2\pi r_2)\cos(2\pi r_1)^2\cos(2\pi r_3)\\
&+\alpha_2\cos(2\pi r_2)^2\cos(2\pi r_1)\sin(2\pi r_1) - \alpha_2\cos(2\pi r_1)\sin(2\pi r_3)\\
&+\alpha_3\cos(2\pi r_1)\sin(2\pi r_3)\cos(2\pi r_2)^2 - \alpha_3\sin(2\pi r_2)\cos(2\pi r_1)^2\\
&-\alpha_3\sin(2\pi r_1)\cos(2\pi r_1) - \alpha_4 2\pi r_3\sin(2\pi r_3)\cos(2\pi r_2)\\
&+\alpha_4 2\pi r_3\sin(2\pi r_3)\cos(2\pi r_2)\cos(2\pi r_1)^2\\
&+\alpha_4\sin(2\pi r_1)\sin(2\pi r_3)\cos(2\pi r_2)\cos(2\pi r_3)\\
&+\alpha_5 16 r_3\sin(2\pi r_1)\sin(2\pi r_2))\\
(B_{11}\alpha)_2(\boldsymbol{r}) := -\frac{1}{8}\pi\,(&-\alpha_5 8\pi\nu\,(\cos(2\pi r_1) + 2\sin(2\pi r_2)\sin(2\pi r_3)) + \alpha_6\cos(2\pi r_3)\\
&-\alpha_6\cos(2\pi r_3)\cos(2\pi r_1)^2 - \alpha_6\sin(2\pi r_1)\cos(2\pi r_2)\cos(2\pi r_1)\\
&-\alpha_7\cos(2\pi r_2)\sin(2\pi r_2) - \alpha_7\cos(2\pi r_2)\sin(2\pi r_3)\cos(2\pi r_1)\\
&+\alpha_7\sin(2\pi r_2)\cos(2\pi r_3)\cos(2\pi r_1)\sin(2\pi r_3)\\
&+\alpha_8 2\pi r_3\sin(2\pi r_2)\cos(2\pi r_3)\cos(2\pi r_2)\sin(2\pi r_1)\\
&-\alpha_8 16 r_3\cos(2\pi r_2)\cos(2\pi r_1))\\
(B_{11}\alpha)_3(\boldsymbol{r}) := -\frac{1}{8}\big(&16\pi^2\nu\,(\alpha_8\cos(2\pi r_1)\sin(2\pi r_3) + \alpha_9 2\pi r_3\cos(2\pi r_2)\sin(2\pi r_1)\\
&-\alpha_9\cos(2\pi r_2)\cos(2\pi r_3)) - \alpha_9 2\pi^2 r_3\sin(2\pi r_3)\sin(2\pi r_1)\\
&-\alpha_{10}2\pi^2 r_3\sin(2\pi r_2)\cos(2\pi r_1)\sin(2\pi r_1)\\
&+\alpha_{10}\pi\sin(2\pi r_2)\cos(2\pi r_1)\cos(2\pi r_3) - \alpha_{10}\pi\cos(2\pi r_2)\cos(2\pi r_1)\\
&+\alpha_{11}2\pi\cos(2\pi r_2)(\cos(2\pi r_3))^2\cos(2\pi r_1) - \alpha_{11}2\pi^2 r_3(\cos(2\pi r_2))^2\\
&+\alpha_{11}\pi(\cos(2\pi r_2))^2\cos(2\pi r_3)\sin(2\pi r_1) - \alpha_{11}8\cos(2\pi r_1)\sin(2\pi r_2)\big)\ .
\end{aligned}$$

4.4.2. Numerical Realisation: Discretisation Issues and Optimisation Method of Choice. As it is illustrated within this chapter the key to solve the considered optimisation problems numerically is to evaluate a series of objective functionals and its derivatives. Using AD in a way how it is described in the previous section the desired evaluated derivatives can be computed by means of an algorithm which has the same basic structure as the one needed to obtain the evaluated functionals itself. Further, as it has been remarked before, the most challenging and computationally costly tasks are solving the side conditions. This in turn is, in the here considered cases, equipollent to solving fluid flow problems which are very similar to that presented in Subsection 2.3.1. To be precise, they only differ in the choices of the force terms. This suggests to base the discretisation strategy for the optimisation problems on that of the fluid flow problem which is stated in detail in Subsection 2.3.2. Following the same strategy the stationary problems are solved numerically by considering them as instationary problems and applying an LB scheme with a stop criterion. Thereto, the choices of the LB model, the boundary conditions and the stop criterion remain as proposed there. The obtained discrete spaces are Ω_h for the underlying position space Ω and Q for the velocity space $\mathbb{R}^3$ for arbitrary discretisation parameters $h \in \mathbb{R}_{>0}$.

The space left to be replaced by a discrete one is the control space U_∞ of the distributed control problem. The control spaces U_1, U_3 and U_{11} for the parameter identification problems are by definition finite dimensional. To discretise U_∞ real trigonometric polynomials of several degrees, namely $N = 1, 2, 3$, are chosen. The underlying Fourier ansatz reads for $\boldsymbol{r} \in \Omega_h$, for each space dimension $i = 1, 2, 3$ and for each considered polynomial degree $N = 1, 2, 3$

$$\left(B_{h,n}(\alpha_n)\right)_i(\boldsymbol{r}) := \prod_{j=1}^{3}\left(a_{i,j,1} + \sum_{k=1}^{N}\left(a_{i,j,2k}\sin\left(2k\pi r_j\right) + a_{i,j,2k+1}\cos\left(2k\pi r_j\right)\right)\right)$$

where $(\alpha_n)_{i,k_1,k_2,k_3} := a_{i,1,k_1}a_{i,2,k_2}a_{i,3,k_3} \in \mathbb{R}$ for any combination of $i \in \{1, 2, 3\}$ and $k_1, k_2, k_3 \in \{1, 2, ..., 2N + 1\}$. The dimension of the obtained control spaces U_n are $n = 3(2N + 1)^3$ which can be identified with $\mathbb{R}^{3(2N+1)^3}$. This leads to three different considered discrete optimisation problems with space dimensions 81, 375 and 1029.

For the three parameter identification problems ($n = 1, 3, 11$) the finite control extension operators are given by their restriction to Ω_h, namely by setting $\boldsymbol{B_{h,n}} := \boldsymbol{B_n}|_{\boldsymbol{\Omega_h}}$. With the choice of the corresponding LB model, consequently, six side conditions $\boldsymbol{G_{h,n}}$ ($n = 1, 3, 11, 81, 375, 1029$) are defined for an arbitrary discretisation parameter $h \in \mathbb{R}_{>0}$.

Finally, the goal functionals J_n ($n = 1, 3, 11, 81, 375, 1029$) need to be replaced by suitable functionals $J_{h,n}$ for $h \in \mathbb{R}_{>0}$. Numerical integration by midpoint rule is used to replace the integrals by sums which leads to

$$J_{h,n}(f_i^h, \boldsymbol{\alpha_n}) := \frac{1}{2}h^3 \sum_{\boldsymbol{r}\in\Omega_h}\Big(\boldsymbol{u}_{\boldsymbol{f_i^h}}(\boldsymbol{r}) - \boldsymbol{u}^*(\boldsymbol{r})\Big)^2 + \frac{\alpha_{reg}}{2}h^3 \sum_{\boldsymbol{r}\in\Omega_h}\left(\left(\boldsymbol{B_{h,n}\alpha_n}\right)(\boldsymbol{r})\right)^2 . \tag{4.11}$$

The discrete optimisation problems are solved iteratively by a line search algorithm. The optimisation method of choice is a BFGS scheme combined with the Wolfe-Powell rule to determine the step lengths. As initial value for the optimisation scheme $\boldsymbol{\alpha_n^0} := \boldsymbol{0}$ for all considered n. The regularisation parameter is set to $\alpha_{reg} = 0$.

4.4.3. Presentation and Discussion of the Numerical Results. In the following, the numerical results of the considered test cases are presented and discussed in detail. Hereby, one main aim is to validate the realised approach. Another one is to obtain results which can be compared with ones gained by other approaches as those presented in the next chapter. Thereto, at first the three parameter identification problems are considered. Here, the goal functionals are slightly modified such that their discrete solutions $\boldsymbol{\alpha_n^*}$ are known in advance. For that configuration numerical results are presented for several discretisation parameters h. Then, the evolution of the iteratively obtained controls $\boldsymbol{\alpha_n^k}$ $(k = 1, 2, ...)$ towards their corresponding discrete solutions $\boldsymbol{\alpha_n^*}$ is monitored. Furthermore, estimates are stated that link the modified to the original problem formulations. Afterwards, for the distributed control problems the evolutions of both the obtained evaluated goal functionals $J_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})$ and the norm of the evaluated gradients $\frac{d}{d\boldsymbol{\alpha}} J_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})$ are studied for $k = 1, 2, ...$ as well as before for differently chosen h.

The analysis aiming to validate the approach can take advantage of the relation between the considered optimisation problems to the fluid flow problem which is discussed in Section 2.3. As it has been remarked in Subsection 4.4.1, for the continuous problem formulations $(n = 1, 3, 11, \infty)$ it holds $2J_n(f(\boldsymbol{\alpha}^*), \boldsymbol{\alpha}^*) = \|\boldsymbol{u}_{\boldsymbol{f}(\boldsymbol{\alpha}^*)} - \boldsymbol{u}^*\|^2_{L^2(\Omega)}$ if $\alpha_{reg} = 0$ and if $\boldsymbol{\alpha}^*$ is chosen such that $\boldsymbol{B_n}\boldsymbol{\alpha}^* = \boldsymbol{F}^*$. Tying up on this relation, a similar statement can be derived for the considered parameter identification problems on a discrete base. Again, the regularisation parameter α_{reg} is set to zero. The controls are set to $\boldsymbol{\alpha_n^*} := (1\ 1\ \cdots\ 1)^T$ for $n = 1, 3, 11$. Then, from the definitions of $\boldsymbol{B_{h,n}}$ directly follows $\boldsymbol{F}^*|_{\Omega_h} = \boldsymbol{B_{h,n}}\boldsymbol{\alpha_n^*}$ and therefore $2J_{h,n}(f_i(\boldsymbol{\alpha_n^*}), \boldsymbol{\alpha_n^*}) = \|\boldsymbol{u}_{\boldsymbol{f_i}(\boldsymbol{\alpha_n^*})} - \boldsymbol{u}^*\|^2_{L^2(\Omega_h)}$. This norm in turn is nothing else but the discretisation and model error which is analysed in Subsection 2.3.3. There, it is observed that $\|\boldsymbol{u}_{\boldsymbol{f_i}(\boldsymbol{\alpha_n^*})} - \boldsymbol{u}^*\|^2_{L^2(\Omega_h)}$ decreases with an order of $EOC \approx 2$. Consequently, these choices of $\boldsymbol{\alpha_n^*}$ $(n = 1, 3, 11)$ are not necessarily solutions of the parameter identification problems. With the triangle inequality an upper bound for the goal functionals is given by

$$\begin{aligned} J_{h,n}(f_i(\boldsymbol{\alpha_n}), \boldsymbol{\alpha_n}) &= \frac{1}{2}\|\boldsymbol{u}_{\boldsymbol{f_i}(\boldsymbol{\alpha_n})} - \boldsymbol{u}^*\|^2_{L^2(\Omega_h)} + \frac{\alpha_{reg}}{2}\|\boldsymbol{B_{h,n}}\boldsymbol{\alpha_n}\|^2_{L^2(\Omega_h)} \\ &\leq \underbrace{\frac{1}{2}\|\boldsymbol{u}_{\boldsymbol{f_i}(\boldsymbol{\alpha_n})} - \boldsymbol{u}_{\boldsymbol{f_i}(\boldsymbol{\alpha_n^*})}\|^2_{L^2(\Omega_h)} + \frac{\alpha_{reg}}{2}\|\boldsymbol{B_{h,n}}\boldsymbol{\alpha_n}\|^2_{L^2(\Omega_h)}}_{=:\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n}),\boldsymbol{\alpha_n})} \\ &\quad + \frac{1}{2}\|\boldsymbol{u}_{\boldsymbol{f_i}(\boldsymbol{\alpha_n^*})} - \boldsymbol{u}^*\|^2_{L^2(\Omega_h)} \end{aligned}$$

which holds for all $h \in \mathbb{R}_{>0}$ and all considered $\boldsymbol{\alpha_n} \in U_n$ $(n = 1, 3, 11)$. If h tends to zero $\widetilde{J}_{h,n}$ will tend to $J_{h,n}$ for $n = 1, 3, 11$ as a consequence of the before observed property of $\|\boldsymbol{u}_{\boldsymbol{f_i}(\boldsymbol{\alpha_n^*})} - \boldsymbol{u}^*\|^2_{L^2(\Omega_h)}$. On the other hand for a given $h \in \mathbb{R}_{>0}$ and $\alpha_{reg} = 0$, $\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n}), \boldsymbol{\alpha_n})$ will tend, of course, to zero if $\boldsymbol{\alpha_n}$ tends to $\boldsymbol{\alpha_n^*}$. This can indeed be observed in all the numerical experiments. In Figure 4.2 the normed errors $\|\boldsymbol{\alpha_n^k} - \boldsymbol{\alpha_n^*}\|_2/\|\boldsymbol{\alpha_n^1} - \boldsymbol{\alpha_n^*}\|_2$ are plotted as a function of iteration steps $k = 1, 2, ...$. The results clearly affirm the theoretically expected asymptotic behaviour. In Figure 4.3 the corresponding evolutions of the normed goal functionals $\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})/\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^1}), \boldsymbol{\alpha_n^1})$ and in Figure 4.4 of their normed gradients $\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})\|_2/\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^1}), \boldsymbol{\alpha_n^1})\|_2$ are depicted.

For the parameter identification problem with $n = 3$ the approach is tested for four different discretisation parameters, namely $h = 1/10, 1/20, 1/40, 1/80$. The numerically observed convergence characteristics are found to be similar. However, if one compares the convergence characteristics between the three parameter identification problems $n = 1, 3, 11$ one clearly finds that the higher the dimension n of the control space U_n the more optimisation steps are necessary to reach a certain value of improvement. For example for an improvement of 10^{-10} of the normed goal functional $\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k})/\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^1}), \boldsymbol{\alpha_n^1})$ for $n = 1$ one step, for $n = 3$

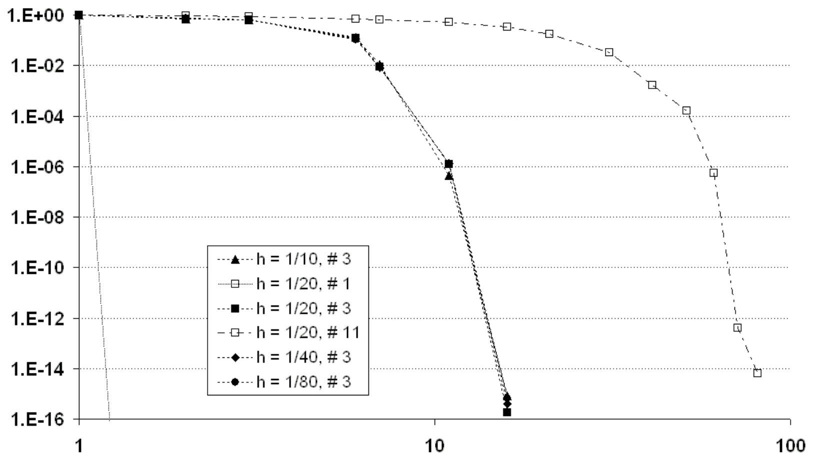


FIGURE 4.2. The evolution of the normalised normed errors $\|\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}} - \boldsymbol{\alpha}_{\boldsymbol{n}}^{*}\|_2 / \|\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{1}} - \boldsymbol{\alpha}_{\boldsymbol{n}}^{*}\|_2$ for the three considered modified parameter identification problems ($n = 1, 3, 11$) are plotted as functions of optimisation steps $k = 1, 2, ...$ for different discretisation parameters h.

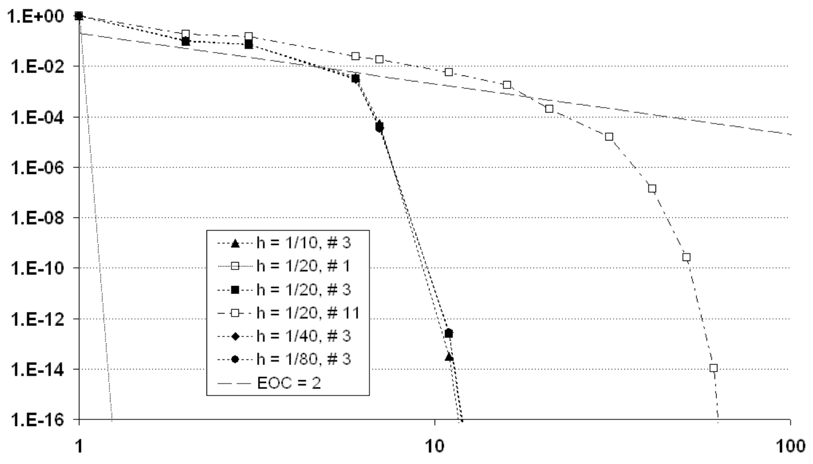


FIGURE 4.3. The evolution of the normalised goal functionals $\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}}), \boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}}) / \widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{1}}), \boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{1}})$ for the three considered modified parameter identification problems ($n = 1, 3, 11$) is given as a function of the number of optimisation steps k for different discretisation parameters h.

nine steps and for $n = 11$ fifty-two steps are needed.

A different strategy is followed to analyse the approach tested on the distributed control problem. In contrast to the parameter identification problem the goal functional remains unchanged. One reason is that the emerging discretisation error cannot be clearly separated. With the chosen ansätze $\boldsymbol{B_{h,n}\alpha_n}$ ($n = 81, 375, 1029$) the force $\boldsymbol{F}^*|_{\boldsymbol{\Omega}_h}$ is only approximated, i.e. $\boldsymbol{F}^*|_{\boldsymbol{\Omega}_h}$ cannot be constructed by any $\boldsymbol{\alpha_n} \in \mathbb{R}^n$. Therefore, in the case where $\alpha_{reg} = 0$ it cannot be guaranteed that there exists an $\boldsymbol{\alpha_n} \in \mathbb{R}^n$ such that $J_{h,n}(f_i(\boldsymbol{\alpha_n}), \boldsymbol{\alpha_n}) = 0$ or $\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n}), \boldsymbol{\alpha_n}) = 0$.

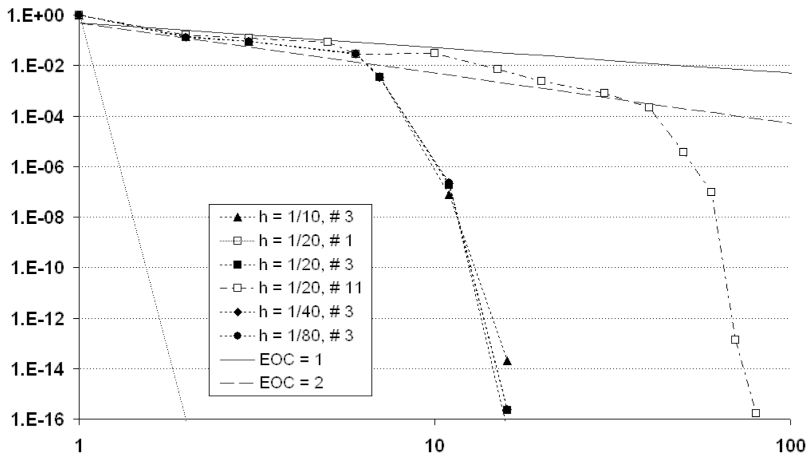


FIGURE 4.4. The evolution of the normalised normed gradient $\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}}),\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}})\|_2/\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{1}}),\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{1}})\|_2$ for the three considered modified parameter identification problems ($n = 1, 3, 11$) is given as a function of the number of optimisation steps k for different discretisation parameters h.

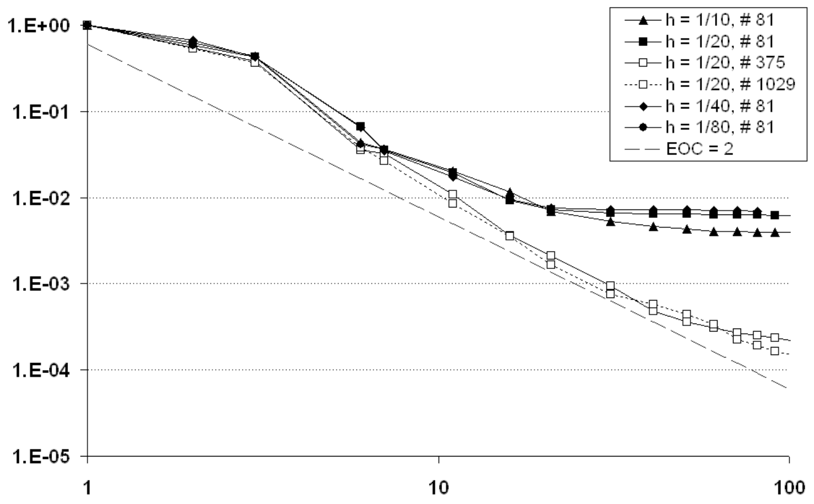


FIGURE 4.5. The evolution of the normalised goal functionals $J_{h,n}(f_i(\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}}),\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{k}})/J_{h,n}(f_i(\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{1}}),\boldsymbol{\alpha}_{\boldsymbol{n}}^{\boldsymbol{1}})$ is given as a function of the number of optimisation steps k for the distributed control problem for $n = 81, 375, 1029$ and different discretisation parameters h.

Another reason is that the results are comparable to that presented in Section 5.6.

In Figure 4.5 the evolution of the normalised goal functional $J_{h,n}(f^k(\boldsymbol{\alpha}_h^k),\boldsymbol{\alpha}_h^k)/J_{h,n}(f^1(\boldsymbol{\alpha}_h^1),\boldsymbol{\alpha}_h^1)$ is stated as a function of optimisation steps $k = 1, 2, ...$ for chosen $n = 81, 375, 1039$ and $h = 1/10, 1/20, 1/40, 1/80$. Thereunto in Figure 4.6 the evolutions of the corresponding normalised normed gradients $\|\frac{d}{d\boldsymbol{\alpha}}J_{h,n}(f^k(\boldsymbol{\alpha}_h^k),\boldsymbol{\alpha}_h^k)\|_2/\|\frac{d}{d\boldsymbol{\alpha}}J_{h,n}(f^1(\boldsymbol{\alpha}_h^1),\boldsymbol{\alpha}_h^1)\|_2$ are plotted.

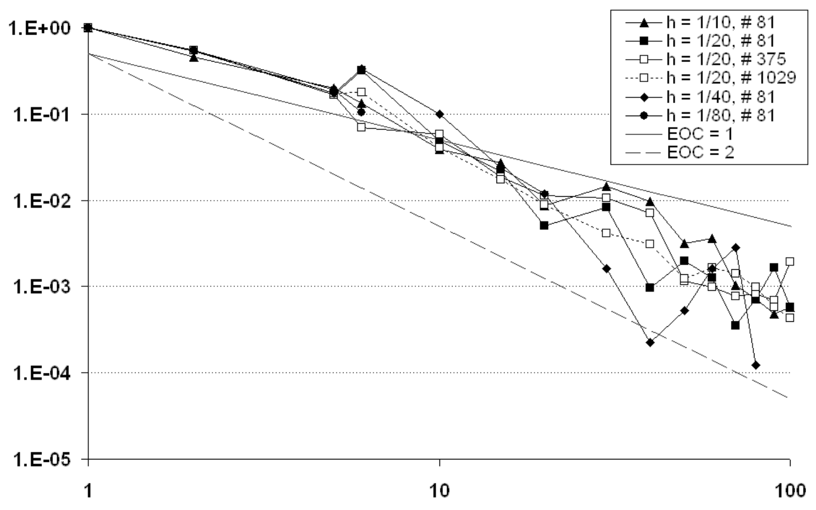


FIGURE 4.6. The evolution of the normalised normed gradients $\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^k}),\boldsymbol{\alpha_n^k})\|_2/\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^1}),\boldsymbol{\alpha_n^1})\|_2$ is given as a function of the number of optimisation steps k for the distributed control problem for $n = 81, 375, 1029$ and different discretisation parameters h.

In contrast to the results of the parameter identification problems it is not observed that all goal functionals $J_{h,n}(f_i(\boldsymbol{\alpha_n^k}),\boldsymbol{\alpha_n^k})$ $(n = 81, 375, 1029)$ of the distributed control problem tend to zero if k increases. For the first order Fourier approximation $(n = 81)$ $J_{h,81}(f_i(\boldsymbol{\alpha_{81}^k}),\boldsymbol{\alpha_{81}^k})/J_{h,81}(f_i(\boldsymbol{\alpha_{81}^1}),\boldsymbol{\alpha_{81}^1})$ tends to approximately $8\cdot 10^{-3}$ for all four considered discretisation parameters h. For the first 100 optimisation steps a similar convergence is not observed for the second $(n = 375)$ and third order Fourier approximation $(n = 1029)$. The found independence of the results from h in the case $n = 81$ together with the observed improvement for a fixed $h = 1/20$ affirm the conclusion that the choice of h cannot be assumed to be the dominant restriction. Thence, it seems plausible that this observed convergence behaviour reflects the approximation error of $\boldsymbol{F^*}|_{\boldsymbol{\Omega_h}}$. The best improvement of $J_{h,1029}(f_i(\boldsymbol{\alpha_{1029}^k}),\boldsymbol{\alpha_{1029}^k})/J_{h,1029}(f_i(\boldsymbol{\alpha_{1029}^1}),\boldsymbol{\alpha_{81}^1}) \approx 1.49\cdot 10^{-04}$ is obtained in the case $n = 1029$ after 100 optimisation steps which is just slightly better than the corresponding result, namely about $2.22\cdot 10^{-04}$, in the case $n = 1029$.

For all considered n and h the trends of the normalised normed gradients $\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^k}),\boldsymbol{\alpha_n^k})\|_2/\|\frac{d}{d\boldsymbol{\alpha}}\widetilde{J}_{h,n}(f_i(\boldsymbol{\alpha_n^1}),\boldsymbol{\alpha_n^1})\|_2$ are found to be to decreasing while the number of optimisation steps increase. Yet, non of the evolutions is observed to decrease steadily. Instead, oscillations are monitored. The amplitudes of the oscillations are found to be the smaller the higher the dimension n of the considered control space U_n is. Interesting to note is that as raw trend lines could serve a linear $(EOC = 1)$ and a quadratical $(EOC = 2)$ decreasing function in k. Similar results are obtained by testing another approach which is presented in the next chapter. Thereunto, in Section 5.6 the observed analogy is discussed.

4.4.4. Presentation and Discussion of the Performance Results. The last part of this subsection is dedicated to present and discuss the performance results which are obtained by both sequential and parallel execution. As test environment serves in either case the *HP XC4000* supercomputer at the Steinbuch Centre for Computing at the Universität Karlsruhe (TH). A detailed description

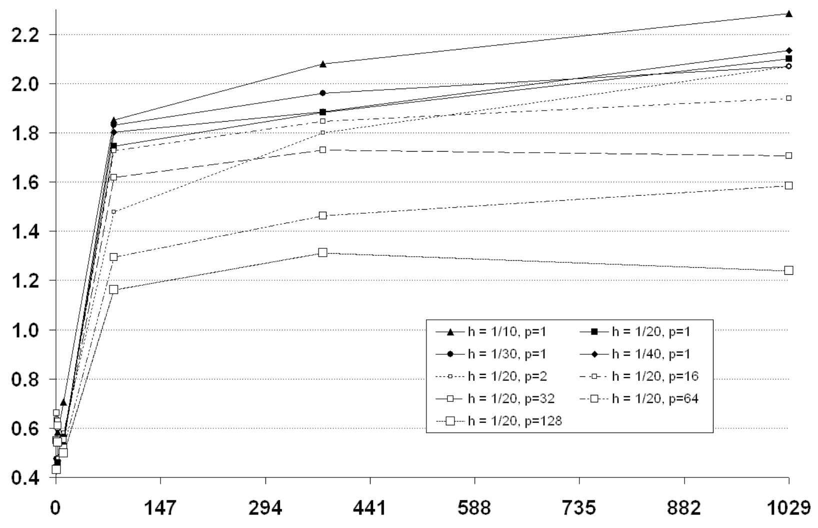


FIGURE 4.7. The averages of the measured number of additional operations $C_{grad}(p, h, n)$ are depicted as functions of the dimension n of the gradient of the goal functional for various discretisation parameters h executed on p processors.

can be found in Section 3.3. The source code is compiled with the Intel' compiler (version 10.1.022) using the optimisation option option $-O3$ and linking Hewlett Packard's MPI library (version 2.3.1). To avoid random hardware-caused outliers all tests are performed three times. In the following, all referred quantities are averages of the corresponding three test results.

The parameter identification problems and the distributed control problem specified in the beginning of this section are considered as test cases. In their discrete formulations they can be distinguished by their respective dimension n ($n = 1, 3, 11, 81, 375, 1029$) of the control space U_n which is in turn the number of derivatives needed to be evaluated. For all tests always only one core and, with it, only one processor per node is employed. The number of processors involved in a particular test is denoted by p.

For all considered test configurations $t_p(h, n)$ denotes the observed time in seconds needed to obtain the corresponding n evaluated derivatives employing p processors. Due to the affinity of the explored test cases (n=1,3,11,81,375,1029) it is expected that all measured elapsed times needed for the evaluation of the corresponding goal functionals $J_{h,n}$ are found to differ only slightly as long h is kept fixed. In fact, the measured quantities affirm the expectation. For example, in the case where $h = 30$ the maximum relative deviation is observed to be approximately 1.5 per cent. Therefore, a comparison of $t_p(h, n)$ based on the average $t_p(h, 0)$ of all six observed elapsed times seems to be promising and justifies the definition of $t_p(h, 0)$.

An interesting question arises concerning the extent of computational costs needed to obtain the evaluated derivatives of the goal functionals $J_{h,n}$ with respect to the dimension n of the gradient and the discretisation parameter h. Thereto,

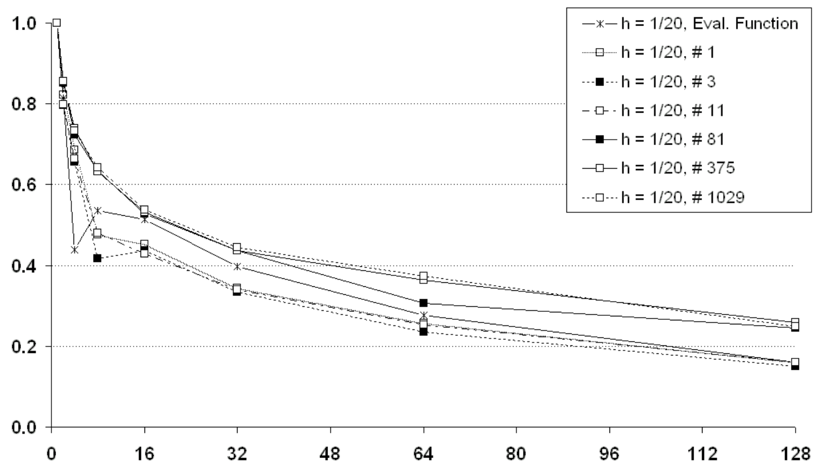


FIGURE 4.8. The evolution of the efficiencies $E_{ff}(p, h, n)$ are depicted as functions of the number of processors p for the fixed discretisation parameter $h = 1/20$ and all considered optimisation problems which are charaterised by their control space dimension n.

the following ansatz is found to be useful

$$t_p(h, n) = t_p(h, 0) + n\ C_{grad}(p, h, n)\ t_p(h, 0) \tag{4.12}$$

where C_{grad} is a function which is implicitly defined by the observations t_p. This ansatz reflects the structure of the AD forward algorithm because of two reasons. Firstly, the computation of an evaluated derivative always requires to perform all the steps which are needed to evaluate the actual function. This is taken into account by the first summand $t_p(h, 0)$. Secondly, the chain rule implies for each elementary operation j and every singe derivative $i = 1, 2, ..., n$ a certain number of additional operations $C^j_{grad}(p, h, n)$. The costs for all elementary operation are represented by $t_p(h, 0)$. Thus, the second summand of (4.12) is meaningful with $C_{grad}(p, h, n)$ presenting the average of the measured number of additional operations $C^j_{grad}(p, h, n)$. In Figure 4.7 C_{grad} is plotted as a functions of n for different fixed discretisation parameters h executed on various numbers of processors p.

At first, the results of the sequentially executed program are discussed. It is observed that in the cases where the dimension of the gradient is rather small, that is $n = 1, 3, 11$, $C_{grad}(1, h, n)$ is found to be small as well, namely always less than 0.75 for all considered $h = 1/10, 1/20, 1/30, 1/40$. Further, the coefficients measured increase if n increases. This also holds for the problems with relatively large gradient dimensions $n = 81, 375, 1029$. However, the growth rate is considerably smaller, e.g $C_{grad}(1, 1/20, 81) \approx 1.84$ grows to $C_{grad}(1, 1/20, 81) \approx 2.12$. Varying the discretisation parameter h the observed differences are not found to be considerable except in the case $h = 1/10$. This can be explained by the fact that the computations for all inner nodes of Ω_h do not differ from each other but differ from those of for the boundary nodes. Due to the problem configuration the computational costs for the boundaries grow less fast than those for the inner nodes if h increases. With it, the uniform costs related to inner nodes dominate and therefore the problem scales almost linearly in h as long as h is small.

Before the quantity C_{grad} is studied for the parallel case the efficiency E_{ff} of the parallel approach is examined. The coefficient E_{ff} is defined similarly to (3.1) by

$$E_{ff}(p,h,n) := \frac{t_1(h,n)}{p\ t_p(h,n)}\ .$$

The discretisation parameter is chosen to be fixed to $h = 1/20$. In Figure 4.8 the obtained efficiencies E_{ff} are depicted as functions of employed processors p. Any of the functions represents one of the six considered problems which is characterised by the dimension $n = 1, 3, 11, 81, 375, 1029$ of the corresponding control space. In the cases with relatively high dimensions, namely $n = 81, 375, 1029$, it turns out that for all considered p the observed efficiencies $E_{ff}(p, 1/20, n)$ are found greater than the efficiencies $E_{ff}(p, 1/20, 0)$ which represents the cases where solely the the goal functions are evaluated. This result is not surprising since, as noted before, the total computational costs grow almost linearly in n with a growth rate of about 2. However, the communication costs grow linearly in n with a growth rate of about 1.

The observed improvement of the efficiencies $E_{ff}(p, 1/20, n)$ for large control space dimension $n = 75, 375, 1029$ is reflected in terms of a smaller measured coefficient $C_{grad}(p, 1/20, n)$ for an increasing number of employed processors p. This is visualised in Figure 4.7.

In summary, the tests show that the needed time to perform one optimisation step roughly grows linearly with the dimension of the control space for both sequential and parallel execution. However, for the parallel case the growth rate can be reduced. Further, the observations also clearly show that the proposed parallelisation strategy for the AD forward mode approach is well suited in terms of scalability.

CHAPTER 5

Adjoint-based Fluid Flow Control and Optimisation based on the BGK-Boltzmann Equation

Like the previous chapter, the actual chapter considers fluid flow control and optimisation problems whose underlying fluid flow models are of mesoscopic nature. In the following, such problems are of interest which can be formulated as constrained optimisation problems in an abstract manner according to

$$\left.\begin{array}{l}\text{find control } \boldsymbol{\alpha} \text{ and state } f \text{ which}\\ \text{minimise } J(f,\boldsymbol{\alpha}) \text{ and fulfil } \boldsymbol{G}(f,\boldsymbol{\alpha}) = \mathbf{0}\end{array}\right\} \tag{5.1}$$

with the same terminology as it is introduced in Chapter 4. It is to be stressed that the governing equation of the side condition is again always a BGK-Boltzmann-like equation similar to (1.67). Consequently, the class of problems, which is considered in the following, is exactly the same as in Chapter 4.

However, the proposed strategy to numerically solve problems of this class fundamentally differs to the one discussed in Chapter 4. In the following, the first-optimise-then-discretise strategy is applied (cf. Gunzburger [**56**]). Thus, at first a necessary condition for an optimum of the continuous problem (5.1) is derived by applying Lagrange's formalism. Then, in order to solve the system, a gradient-based optimisation algorithm is applied. This leads to a series of continuous problems. Each problem comprises the underlying fluid flow problem itself and a second so-called *adjoint problem*. It turns out that the governing equation of the adjoint problem is similarly structured as the BGK-Boltzmann equation. Therefore, dedicated discretisation methods akin LBM can be applied. This leads to discrete equations which have similar properties to those of LB equations. Thus, it is possible to carry over efficient implementation strategies and other concepts such as the hybrid parallelism concept introduced in Chapter 3. With it, the adjoint system and, finally, the whole optimisation problem can be solved highly efficiently in parallel.

The presented solution strategy is also in contrast to the other proposed approaches suggested by Pingen et al. [**108, 109, 110, 111**] and Tekitek et al. [**131**]. Though they also use optimisation algorithms to solve the optimality system, they all derive the adjoint equations for dedicated LB equations. Since this equations are of discrete nature their approaches follow the first-discretise-then-optimise strategy. Further, the approaches differ because the discrete adjoint equations are solved by e.g. a Schur-complement method [**109**]. Thus, there is not taken advantage of the structure of the adjoint problems as it is done in the here presented approach.

This chapter aims to introduce, discuss and apply the mentioned solution strategy. Thereby, emphasis is placed in presenting the framework in a widely general manner. However, in order to illustrate the feasibility and efficiency of the approach, one example is chosen to be considered, namely a 3D distributed control problem. This example also serves as a prototype every time it is required to leave the general-presentation principle. The remainder of this chapter is organised as follows. It starts with the presentation of the overall strategy in Section 5.1. Then, in the following three sections, the proposed approach is substantiated in detail with respect to three main aspects. The first one, presented in Section 5.2, concerns the derivation of a first-order necessary optimality system on the basis of the mentioned prototypical example. Section 5.3 is dedicated to derive a general form of an *adjoint BGK-Boltzmann equation.* Then, in Section 5.4 a discretisation strategy for this equation is proposed which is inspired by LBM as they are studied in Chapter 2. A numerical experiments is considered in the last section which is Section 5.6. There, a discussion is held about the numerical results but also about performance issues. Further, both the numerical and the performance results are compared to those results presented in Chapter 4.

5.1. From the Optimisation Problem Formulation to the Numerical Solution: An Adjoint-based Strategy

Likewise to the overall strategy presented in the previous chapter in Section 4.1 in the following, a strategy is proposed which enables solving numerically fluid flow control and optimisation problems which are formulated as in (5.1). Yet, the introduced approach differs form the previous one in two aspects. The first one is that the actual approach follows the first-optimise-then-discretise strategy while the former one follows the first-discretise-then-optimise-strategy. The second difference is the way the evaluated derivatives are obtained. The actual discussed strategy uses an adjoint equation to determine them, while the previous one uses AD techniques. The aim of this section is to introduce the overall approach step by step, stressing similarities and differences to the formerly discussed approach and, moreover, to ones proposed by others.

The following four steps are to be taken before an optimisation problem which is formulated as in (5.1) can be solved numerically:

(1) **Deriving an optimality system** with Lagrange's formalism which sets up a condition for an optimum,
(2) **Applying an iterative gradient-based optimisation algorithm** leading to a series of problems which are to be solved in every optimisation step $k = 1, 2, ...$ for a particular explicitly given control variable $\boldsymbol{\alpha^k}$,
(3) **Deriving the prototypical adjoint equation** to obtain the needed evaluated gradients,
(4) **Discretising**, in particular, the primal and the adjoint equation.

5.1.1. First Step: Deriving an Optimality System.

In the first step a necessary condition for an optimum $(f^*, \boldsymbol{\alpha^*})$ of an optimisation problem which is given according to (5.1) is to be derived. This is done on a continuous base using Lagrange's formalism. As it is illustrated in the following section, a non-linear coupled system called *optimality system* is obtained. In (5.27) a prototypical optimality system is stated. It is derived for an example which is a 3D distributed control problem. In general, such systems usually consist of three equations. The first one is given by the side condition $\boldsymbol{G}(f^*, \boldsymbol{\alpha^*}) = \boldsymbol{0}$ itself. In this context it is said to be the *primal problem.* The second and third equation set up conditions for the optimum $(f^*, \boldsymbol{\alpha^*})$ and an optimal value $\boldsymbol{\lambda^*}$ for another variable $\boldsymbol{\lambda}$ which is referred to as *Lagrange multiplier.* The second condition is an affine-linear equation which is known as *adjoint problem.* As the second equation, the third one usually couples all three variables $f^*, \boldsymbol{\alpha^*}$ and $\boldsymbol{\lambda^*}$. It is called the *optimality condition.*

5.1.2. Second Step: Applying an Iterative Gradient-based Optimisation Algorithm. Due to the in general complex structure of the obtained optimality system, which is in particular non-linear, an iterative method is applied to obtain a solution. Similarly to the sensitivity-based strategy discussed in Section 4.1 the application of a gradient-based optimisation algorithm is proposed. Here, however, it is applied to a continuous problem. Yet, the procedure remains the same. At first, the problem (5.1) is rewritten in form of an unconstrained optimisation problem. Providing that f can be understood as a function of $\boldsymbol{\alpha}$, i.e. that $f(\boldsymbol{\alpha})$ exists as a function which is implicitly defined by the side condition for any considered $\boldsymbol{\alpha}$, then, the optimisation problem (5.1) equivalently reads

$$\left.\begin{array}{l}\text{find control } \boldsymbol{\alpha} \text{ which}\\ \text{minimises } J(f(\boldsymbol{\alpha}),\boldsymbol{\alpha})\end{array}\right\} . \tag{5.2}$$

Assuming that J is totally differentiable a necessary condition for a minimum $(f(\boldsymbol{\alpha^*}),\boldsymbol{\alpha^*})$ of the reformulated, unconstrained problem (5.2), and with it also of (5.1), is given by

$$\frac{d}{d\boldsymbol{\alpha}} J(f(\boldsymbol{\alpha^*}),\boldsymbol{\alpha^*}) = \mathbf{0} . \tag{5.3}$$

In order to solve the continuous unconstrained optimisation problem (5.2) a line search algorithm just like the one presented in Algorithm 3 is now applied. This leads to a series of descent directions $\boldsymbol{d^k}$ and step lengths δ^k which are to be determined in every optimisation step $k = 1, 2, ...$ for an explicit given $\boldsymbol{\alpha^k}$. The sub-class of gradient-based optimisation methods provides techniques which allow to compute $\boldsymbol{d^k}$ and δ^k from merely evaluated goal functionals J and its total derivatives $\frac{d}{d\boldsymbol{\alpha}}J$. In the literature, e.g. in [**44, 104**], one finds as representative of this class among others the steepest descent method or quasi-Newton methods like the BFGS method in combination with the Armijo rule or the Wolfe-Powell rule.

5.1.3. Third Step: Deriving the Prototypical Adjoint Equation. In this step a strategy to obtain evaluated derivatives $\frac{d}{d\boldsymbol{\alpha}}J(f(\boldsymbol{\alpha}),\boldsymbol{\alpha})$ for given arbitrary $\boldsymbol{\alpha}$ is illustrated. The approach relies on defining an adjoint equation whose solution yields an explicit expression for the wanted evaluated derivative.

For the derivation to come U denotes the control space. It is assumed to be a Hilbert space. For any given $\boldsymbol{\alpha} \in U$ the equation, defined in the following, is said to be the *adjoint equation (for the optimisation problem* (5.1)*)*:

$$\frac{\partial}{\partial f(\boldsymbol{\alpha})} J(f(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = -\boldsymbol{\lambda} \cdot \frac{\partial}{\partial f(\boldsymbol{\alpha})} \boldsymbol{G}(f(\boldsymbol{\alpha}),\boldsymbol{\alpha})) . \tag{5.4}$$

Thereby, its solution $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\boldsymbol{\alpha})$ is known under the terms *dual solution*, *adjoint solution* or shorter the *adjoint.*

Differentiation of the state equation $\boldsymbol{G}(f(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = \mathbf{0}$ $(\boldsymbol{\alpha} \in U)$ on both sides with respect to $\boldsymbol{\alpha}$ by applying the chain rule on the right hand side leads to the so-called *sensitivity equation*:

$$\frac{\partial}{\partial f(\boldsymbol{\alpha})} \boldsymbol{G}(f(\boldsymbol{\alpha}),\boldsymbol{\alpha}) \otimes \boldsymbol{\nabla_\alpha} f(\boldsymbol{\alpha}) = -\boldsymbol{\nabla_\alpha} \otimes \boldsymbol{G}(f(\boldsymbol{\alpha}),\boldsymbol{\alpha}) \qquad (\boldsymbol{\alpha} \in U) . \tag{5.5}$$

With (5.4) and (5.5) for an arbitrary, but fixed $\boldsymbol{\alpha} \in U$ the gradient of J can be transformed according to

$$\begin{aligned}\frac{d}{d\boldsymbol{\alpha}} J(f(\boldsymbol{\alpha}), \boldsymbol{\alpha}) &= \frac{\partial}{\partial f(\boldsymbol{\alpha})} J(f(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \ \boldsymbol{\nabla_\alpha} f(\boldsymbol{\alpha}) + \boldsymbol{\nabla_\alpha} J(f(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\ &= -\boldsymbol{\lambda} \cdot \frac{\partial}{\partial f(\boldsymbol{\alpha})} \boldsymbol{G}(f(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \ \boldsymbol{\nabla_\alpha} f(\boldsymbol{\alpha}) + \boldsymbol{\nabla_\alpha} J(f(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\ &= \boldsymbol{\lambda} \cdot (\boldsymbol{\nabla_\alpha} \otimes \boldsymbol{G}(f(\boldsymbol{\alpha}), \boldsymbol{\alpha})) + \boldsymbol{\nabla_\alpha} J(f(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \ .\end{aligned} \tag{5.6}$$

It is to be noted that the explicit expression for the adjoint equation (5.4) depends on the definition of the goal functional J and side condition function $\boldsymbol{G}$ and therefore on the actual considered problem. Consequently, if one wants to solve an optimisation problem by applying this adjoint-based strategy, one needs to derive an analytic expression for the adjoint equation (5.4) separately for every single problem considered which differs with respect to the structure of J or $\boldsymbol{G}$. However, as indicated in the introduction of this chapter, in the here considered framework the part and parcel of a side condition $\boldsymbol{G}$ is always a BGK-Boltzmann-like equation. Therefore, it is to be expected that the emerging adjoint equation is similarly structured. This issue is addressed in more detail in Section 5.3. There, a governing equation of a prototypical adjoint equation is derived. In the following, this obtained equation is said to be the *adjoint BGK-Boltzmann equation.*

5.1.4. Forth Step: Discretising. The class of the considered fluid flow control and optimisation problems is characterised by the governing equation of the side condition which is a BGK-Boltzmann-like equation similar to (1.67). In Chapter 2 consistent discretisation strategies are discussed which lead to LB equations. There, for a given $h \in \mathbb{R}_{>0}$ the underlying continuous space $I \times \Omega \times \mathbb{R}^d$, where the particle distribution function f is acting on, is replaced by $I_h \times \Omega_h \times Q$. It is proposed to rely the discretisation procedure for the optimisation problem on this strategy as well. Thus, the state f is replaced by its discrete pendant f_i acting on $I_h \times \Omega_h \times Q$.

For the control $\boldsymbol{\alpha}$, which is a function in a Hilbert space U, a second discretisation parameter $n \in \mathbb{N}$ is introduced. In the case where the control space $U =: U_n$ is already a finite dimensional space, n can simply be set to be the space dimension of U. Otherwise, the control space U is replaced by an n-dimensional space U_n. The objective J and its derivative $\frac{d}{d\boldsymbol{\alpha}} J$ are substituted by functions $J_{h,n}$ and $gJ_{h,n}$ which are chosen appropriately with respect to the discrete space $I_h \times \Omega_h \times Q$ and U_n.

With the so far introduced procedure the functional $J_{h,n}$ can be evaluated numerically for any given $\boldsymbol{\alpha_n} \in U_n$. To solve a considered optimisation problem it remains to provide a discretisation strategy for the adjoint equation with respect to (5.1) so that the needed evaluated derivatives can be obtained according to (5.6). Since an adjoint BGK-Boltzmann equation is the primary part of the adjoint equation (5.4) (cf. Subsection 5.2.4) and since it is similarly structured as a BGK-Boltzmann equation, it is suggested to apply methods akin LBM for the discretisation. In the following, these methods are referred to as *adjoint lattice Boltzmann methods* (ALBM). They are introduced in Section 5.4. The underlying unknown function of an adjoint BGK-Boltzmann equation is the *adjoint particle distribution function* φ which acts, as the state f, on $I \times \Omega \times \mathbb{R}^d$. ALBM lead to *adjoint lattice Boltzmann equations* which set up conditions for discrete adjoint particle distribution functions φ_j. As for LBM, the resulting discrete space also is $I_h \times \Omega_h \times Q$ with the same discretisation parameter h.

5.2. First-order Necessary Optimality System

In this section a first-order necessary optimality condition for problems of a prototypical class of optimisation problems with the BGK-Boltzmann equation being part and parcel of the side conditions is derived. The chosen class consists of distributed control problems whereby the force $\boldsymbol{F}$ is to be controlled in the whole domain Ω during I. The presented approach relies on Lagrange's formalism, which leads to a system of three coupled equations. The first equation is governed by the BGK-Boltzmann equation (1.67) and the second one by the adjoint BGK-Boltzmann equation. In general, it is to be noted that as long as the side condition does not differ structurally, the structure of these two equations will not differ as well. Hence, their form is essentially independent of the actual considered class of optimisation problems. This is in contrast to the structure of the third equation. It strongly depends on the variable which is to be controlled and thus on the actual considered problem. However, for many problems the presented approach can easily be adapted. Since the basic structure of the first two equations does not change, the in the following obtained first-order necessary optimality condition can be seen as a prototypical condition for (5.1) - the greater class of constraint optimisation problems at which a BGK-Boltzmann equation constitutes the main part of the side conditions.

In the following, at first the considered class of distributed control problems is formulated. Then, Lagrange's formalism is applied. This sets up several conditions which are finally transformed and condensed to a system of three equations which govern the first-order necessary optimality system.

The considered class consists of such distributed control problems which can be formulated as constrained optimisation problems akin (5.1). Thereby, a BGK-Boltzmann equation together with appropriate initial and boundary conditions serve as subsidiary condition. The force $\boldsymbol{F}$ is to be controlled and hence is replaced by $B\boldsymbol{\alpha}$. Like in the previous section, the Hilbert space U denotes the control space and $B \in \mathcal{L}(U, L^2(I\times\Omega)^d)$ a control extension operator, so that the controlled force $\boldsymbol{F}$ constitutes to

$$\boldsymbol{F} : \begin{cases} I\times\Omega & \to \mathbb{R}^d \\ (t,\boldsymbol{r}) & \mapsto (B\boldsymbol{\alpha})(t,\boldsymbol{r}) \ . \end{cases}$$

The side condition $G(f,\boldsymbol{\alpha}) = 0$ is defined according to

$$\boldsymbol{G} : \begin{cases} H^1(I\times\Omega\times\mathbb{R}^d)\times U & \to L^2(I\times\Omega\times\mathbb{R}^d)\times L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}^-})\times L^2(\Omega\times\mathbb{R}^d) \\ (f,\boldsymbol{\alpha}) & \mapsto \begin{pmatrix} \left(\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha}}{m}\cdot\boldsymbol{\nabla_v}\right) f - Q(f) \\ f(\boldsymbol{v}) - f(-\boldsymbol{v}) \\ f|_{t=t_0} - f_{t_0} \end{pmatrix} \ , \end{cases} \tag{5.7}$$

whereby $\mathbb{R}^d_{\boldsymbol{n}^+} := \left\{\boldsymbol{v}\in\mathbb{R}^d : \boldsymbol{v}\cdot\boldsymbol{n} > 0\right\}$ and $\mathbb{R}^d_{\boldsymbol{n}^-} := \left\{\boldsymbol{v}\in\mathbb{R}^d : \boldsymbol{v}\cdot\boldsymbol{n} < 0\right\}$ denote the half spaces which are given for the outward pointing normal vector $\boldsymbol{n}(\boldsymbol{r})$ at any $\boldsymbol{r}\in\partial\Omega$. The goal functional J reads

$$J : \begin{cases} H^1(I\times\Omega\times\mathbb{R}^d)\times U & \to \mathbb{R}_{\geq 0} \\ (f,\boldsymbol{\alpha}) & \mapsto \frac{1}{2}\int_I\int_\Omega (\boldsymbol{u_f} - \boldsymbol{u}^*)^2 \ d\boldsymbol{r}dt + \frac{\alpha_{reg}}{2}\int_I\int_\Omega (B\boldsymbol{\alpha})^2 d\boldsymbol{r}dt \end{cases} \tag{5.8}$$

where $\boldsymbol{u}^* : I \times \Omega \to \mathbb{R}^d$ is a desired velocity distribution and $\alpha_{reg} \in \mathbb{R}_{\geq 0}$ a regularisation parameter.

Lagrange's formalism as stated in [**142**, p. 270] is to be applied to derive a necessary condition for an optimal solution of (5.1). In the here established notation it reads:

Theorem 5.1. *Let $(f^*, \boldsymbol{\alpha}^*)$ be a solution of the problem formulated in* (5.1), *$J : H^1(I \times \Omega \times \mathbb{R}^d) \times U \to \mathbb{R}$ and $\boldsymbol{G} : H^1(I \times \Omega \times \mathbb{R}^d) \times U \to Y$ with Banach spaces U, Y. If there exists an open neighborhood $K(f^*, \boldsymbol{\alpha}^*) \subseteq H^1(I \times \Omega \times \mathbb{R}^d) \times U$ where $\boldsymbol{G}|_{K(f^*,\boldsymbol{\alpha}^*)} \in C^1$ and $\boldsymbol{\nabla} \otimes \boldsymbol{G}(f^*, \boldsymbol{\alpha}^*) : H^1(I \times \Omega \times \mathbb{R}^d) \times U \to Y$ is surjective, then there exists $\boldsymbol{\lambda}^* \in Y'$ such that the following holds:*

$$\boldsymbol{\nabla} J(f^*, \boldsymbol{\alpha}^*; h_f, \boldsymbol{h_\alpha}) + \boldsymbol{\lambda}^* \boldsymbol{\nabla} \otimes \boldsymbol{G}(f^*, \boldsymbol{\alpha}^*; h_f, \boldsymbol{h_\alpha}) = 0 \tag{5.9}$$

for all directions $(h_f, \boldsymbol{h_\alpha}) \in H^1(I \times \Omega \times \mathbb{R}^d) \times U$.

PROOF. cf. Zeidler [**142**, p. 270] □

One of the premises of the theorem itself sets up the first condition of the necessary optimality system. It is required that the side condition must hold for an optimum $(f^*, \boldsymbol{\alpha}^*)$, i.e.

$$\boldsymbol{G}(f^*, \boldsymbol{\alpha}^*) = \boldsymbol{0} \qquad \text{in } I \times \Omega \times \mathbb{R}^d \ . \tag{5.10}$$

This condition is nothing but the equations which characterise the underlying fluid flow problem of the considered optimisation problem. In this context, (5.10) is commonly said to be the *primal problem* or *state equation*.

If one assumes that all other requirements of Theorem 5.1 are fulfilled, two more necessary conditions for an optimum can be obtained as a consequence of the theorem. Thereunto, (5.9) is split into two equations by choosing in one case $\boldsymbol{h_\alpha} \equiv \boldsymbol{0}$ and in the other $h_f \equiv 0$. Then, one gets the *adjoint equation* according to

$$\frac{\partial}{\partial f} J(f^*, \boldsymbol{\alpha}^*; h_f, \boldsymbol{h_\alpha}) + \boldsymbol{\lambda}^* \frac{\partial}{\partial f} \boldsymbol{G}(f^*, \boldsymbol{\alpha}^*; h_f, \boldsymbol{0}) = 0 \quad (h_f \in \ H^1(I \times \Omega \times \mathbb{R}^d)) \tag{5.11}$$

and the *optimality equation* according to

$$\boldsymbol{\nabla_\alpha} J(f^*, \boldsymbol{\alpha}^*; \boldsymbol{\alpha}) + \boldsymbol{\lambda}^* \boldsymbol{\nabla_\alpha} \otimes \boldsymbol{G}(f^*, \boldsymbol{\alpha}^*; 0, \boldsymbol{h_\alpha}) = 0 \qquad (\boldsymbol{h_\alpha} \in \ U) \ . \tag{5.12}$$

In the following three subsections, the three conditions constituting the whole optimality system are derived for the special choice of $\boldsymbol{G}$ and J which are defined according to (5.7) and (5.8). Finally, in Subsection 5.2.4 the optimality system is stated for the considered class of distributed control problems.

5.2.1. Derivation of the Primal Problem. For an optimum $(f^*, \boldsymbol{\alpha}^*)$ of (5.1) the side condition needs to be fulfilled. In the case of considering a distributed control problem for which the corresponding side condition is defined as in (5.7), one obtains the primal problem according to

$$\begin{aligned} \left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha}^*}{m} \cdot \boldsymbol{\nabla_v} \right) f^* &= Q(f^*) && \text{in } I \times \Omega \times \mathbb{R}^d \\ f^*(\boldsymbol{v}) &= f^*(-\boldsymbol{v}) && \text{on } I \times \partial\Omega \times \mathbb{R}^d_{\boldsymbol{n}^-} \\ f^*|_{t=t_0} &= f_{t_0} && \text{in } \Omega \times \mathbb{R}^d \ . \end{aligned} \tag{5.13}$$

5.2.2. Derivation of the Adjoint Problem. Inserting the special choices for J and $\boldsymbol{G}$, namely (5.8) and (5.7), into (5.11) yields

$$0 = \frac{\partial}{\partial f}\left(\frac{1}{2}\int_I\int_\Omega (\boldsymbol{u_f} - \boldsymbol{u^*})^2 \; d\boldsymbol{r}dt + \frac{\alpha_{reg}}{2}\int_I\int_\Omega (B\boldsymbol{\alpha})^2 d\boldsymbol{r}dt\right)(f^*, \boldsymbol{\alpha^*}; h_f) \quad (5.14)$$

$$+ \left\langle \varphi^*, \frac{\partial}{\partial f}\left(\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha}}{m}\cdot\boldsymbol{\nabla_v} - Q\right)(f^*, \boldsymbol{\alpha^*}; h_f)\right\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \quad (5.15)$$

$$+ \left\langle \gamma^*, \frac{\partial}{\partial f}\left(f(\boldsymbol{v}) - f(-\boldsymbol{v})\right)(f^*, \boldsymbol{\alpha^*}; h_f)\right\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}^-})} \quad (5.16)$$

$$+ \left\langle \eta^*, \frac{\partial}{\partial f}\left(f|_{t=t_0} - f_{t_0}\right)(f^*, \boldsymbol{\alpha^*}; h_f)\right\rangle_{L^2(\Omega\times\mathbb{R}^d)} \quad (5.17)$$

for $(h_f, \boldsymbol{0}) \in H^1(I\times\Omega\times\mathbb{R}^d)\times U$. Y' can be identified with $L^2(I\times\Omega\times\mathbb{R}^d)\times L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}^-})\times L^2(I\times\partial\Omega)$. Applying Riesz's representation theorem the *Lagrange multiplier* $\boldsymbol{\lambda^*} \in Y'$ can be represented by a certain $(\varphi^*, \gamma^*, \eta^*)^T \in L^2(I\times\Omega\times\mathbb{R}^d)\times L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}^-})\times L^2(I\times\partial\Omega)$.

The remainder of this subsection is dedicated to transform the terms with the derivatives (5.14), (5.15), (5.16) and (5.17) step by step and finally determine another system of equations which can be discretised and numerically solved. Thereunto, the following corollary is found to be useful:

Corollary 5.2. *For functionals $f, g \in H^1(\mathbb{R}^d)$ and $\boldsymbol{a}\in\mathbb{R}^d$ the following equality holds:*

$$\int_{\mathbb{R}^d} \boldsymbol{a}\boldsymbol{\nabla} f(\boldsymbol{v})\; g(\boldsymbol{v}) d\boldsymbol{v} = -\int_{\mathbb{R}^d} f(\boldsymbol{v})\; \boldsymbol{a}\boldsymbol{\nabla} g(\boldsymbol{v}) d\boldsymbol{v}\;.$$

PROOF. Let $f, g \in H^1(\mathbb{R}^d)$. Since $C_0^\infty(\mathbb{R}^d)\cap H^1(\mathbb{R}^d)$ is dense in $H^1(\mathbb{R}^d)$, there exists a sequence $(f_n)_{n\in\mathbb{N}} \in C_0^\infty(\mathbb{R}^d)\cap H^1(\mathbb{R}^d)$ such that in the $H^1(\mathbb{R}^d)$-norm $f_n \overset{n\to\infty}{\longrightarrow} f$. The definition of the $H^1(\mathbb{R}^d)$-norm implies that $f_n \overset{n\to\infty}{\longrightarrow} f$ and $\boldsymbol{\nabla} f_n \overset{n\to\infty}{\longrightarrow} \boldsymbol{\nabla} f$ in the $L^2(\mathbb{R}^d)$-norm. Furthermore, taking once more the definition of the $H^1(\mathbb{R}^d)$-norm into account, the following equivalences hold:

$$\begin{aligned}\int_{\mathbb{R}^d} \boldsymbol{a}\boldsymbol{\nabla} f(\boldsymbol{v})\; g(\boldsymbol{v})\; d\boldsymbol{v} &= \lim_{n\to\infty}\int_{\mathbb{R}^d} \boldsymbol{a}\boldsymbol{\nabla} f_n(\boldsymbol{v})\; g(\boldsymbol{v})\; d\boldsymbol{v} \\ &= \lim_{n\to\infty} -\int_{\mathbb{R}^d} f_n(\boldsymbol{v})\; \boldsymbol{a}\boldsymbol{\nabla} g(\boldsymbol{v})\; d\boldsymbol{v} \\ &= -\int_{\mathbb{R}^d} f(\boldsymbol{v})\; \boldsymbol{a}\boldsymbol{\nabla} g(\boldsymbol{v})\; d\boldsymbol{v}\;.\end{aligned}$$

□

The derivative (5.14) of the goal functional at $(f^*, \boldsymbol{\alpha^*})$ in an arbitrary direction $h_f \in H^1(I\times\Omega\times\mathbb{R}^d)$ can be simplified according to

$$\begin{aligned}&\frac{\partial}{\partial f}\left(\frac{1}{2}\int_I\int_\Omega (\boldsymbol{u_f} - \boldsymbol{u^*})^2 \; d\boldsymbol{r}dt + \frac{\alpha_{reg}}{2}\int_I\int_\Omega (B\boldsymbol{\alpha})^2 d\boldsymbol{r}dt\right)(f^*, \boldsymbol{\alpha^*}; h_f) \\ &\quad= -\int_I\int_\Omega n_{h_f}\frac{(\boldsymbol{u^*} - \boldsymbol{u}_{f^*})\left(\boldsymbol{u}_{h_f} - \boldsymbol{u}_{f^*}\right)}{n_{f^*}}\; d\boldsymbol{r}dt \\ &\quad= -\int_I\int_\Omega\int_{\mathbb{R}^d} \frac{(\boldsymbol{u^*} - \boldsymbol{u}_{f^*})\left(\boldsymbol{v}h_f(\boldsymbol{v}) - \boldsymbol{u}_{f^*}h_f(\boldsymbol{v})\right)}{n_{f^*}}\; d\boldsymbol{v}d\boldsymbol{r}dt \\ &\quad= -\left\langle \frac{(\boldsymbol{u^*} - \boldsymbol{u}_{f^*})(\boldsymbol{v} - \boldsymbol{u}_{f^*})}{n_{f^*}}, h_f\right\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)}\;.\end{aligned} \quad (5.18)$$

The second summand (5.15) of the adjoint equation is split into two summands which represent the propagation and the collision part. The computation of the directional derivative of the propagation operator $\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha}}{m} \cdot \boldsymbol{\nabla_v}$ is an easy task since it is linear. The result is further transformed using partial integration for the first two summands and applying Corollary 5.2 for the last summand. Altogether, this leads to

$$\begin{aligned}
&\big\langle \varphi^*, \frac{\partial}{\partial f}\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha}}{m} \cdot \boldsymbol{\nabla_v}\right)(f^*, \boldsymbol{\alpha^*}; h_f)\big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \\
&\quad = \big\langle \varphi^*, \left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha^*}}{m} \cdot \boldsymbol{\nabla_v}\right) h_f \big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \\
&\quad = \big\langle \boldsymbol{v} \cdot \boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d)} - \big\langle \left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha^*}}{m} \cdot \boldsymbol{\nabla_v}\right)\varphi^*, h_f \big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \\
&\qquad + \big\langle \varphi^*|_{t=t_1}, h_f|_{t=t_1} \big\rangle_{L^2(\Omega\times\mathbb{R}^d)} - \big\langle \varphi^*|_{t=t_0}, h_f|_{t=t_0} \big\rangle_{L^2(\Omega\times\mathbb{R}^d)}
\end{aligned}$$

which holds for every direction $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$.

The differentiation of the non-linear collision operator Q with respect to f at $(f^*, \boldsymbol{\alpha^*})$ in an arbitrary direction $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$ and afterwards the isolation of h_f requires several transformation steps. At first, the non-linear term of Q, namely M_f^{eq}, is considered. For $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$ the following transformations hold true:

$$\begin{aligned}
&\big\langle \varphi^*, \frac{\partial}{\partial f} M_f^{eq}(f^*, \boldsymbol{\alpha^*}; h_f)\big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \\
&\quad = \big\langle \varphi^*, \frac{n_{h_f}\left(\left(\boldsymbol{u_{f^*}} - \boldsymbol{u_{h_f}}\right)\left(\boldsymbol{u_{f^*}} - \boldsymbol{v}\right) + RT\right)}{n_{f^*}RT} M_{f^*}^{eq} \big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \\
&\quad = \int_{t_0}^{t_1}\int_\Omega\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \varphi^*(\boldsymbol{v}) \frac{\left(\left(\boldsymbol{u_{f^*}} - \hat{\boldsymbol{v}}\right)\left(\boldsymbol{u_{f^*}} - \boldsymbol{v}\right) + RT\right)}{n_{f^*}RT} M_{f^*}^{eq}(\boldsymbol{v}) h_f(\hat{\boldsymbol{v}})\, d\boldsymbol{v} d\hat{\boldsymbol{v}} d\boldsymbol{r} dt \\
&\quad = \big\langle \underbrace{\int_{\mathbb{R}^d} \varphi^*(\boldsymbol{v}) \frac{\left(\left(\boldsymbol{u_{f^*}} - \hat{\boldsymbol{v}}\right)\left(\boldsymbol{u_{f^*}} - \boldsymbol{v}\right) + RT\right)}{n_{f^*}RT} M_{f^*}^{eq}(\boldsymbol{v})\, d\boldsymbol{v}}_{=:dM_{f^*}^{eq}}, h_f \big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} .
\end{aligned}$$

With it, one obtains the differentiated operator Q according to

$$\big\langle \varphi^*, \frac{\partial}{\partial f} Q(f^*, \boldsymbol{\alpha^*}; h_f)\big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} = \big\langle \underbrace{-\frac{1}{\omega}\left(\varphi^* - dM_{f^*}^{eq}\right)}_{=:dQ(\varphi^*)}, h_f \big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)}$$

for every direction $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$.

The derivative at $(f^*, \boldsymbol{\alpha^*})$ in the third summand (5.16) of the adjoint equation in an arbitrary direction $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$ is now computed. One obtains for any $h_f \in H^1(I \times \Omega \times \mathbb{R}^d$

$$\begin{aligned}
&\big\langle \gamma^*, \frac{\partial}{\partial f}\left(f(\boldsymbol{v}) - f(-\boldsymbol{v})\right)(f^*, \boldsymbol{\alpha^*}; h_f)\big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} \\
&\quad = \big\langle \gamma^*, h_f(\boldsymbol{v}) - h_f(-\boldsymbol{v})\big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} .
\end{aligned}$$

Finally, the last summand (5.17) of the adjoint equation is transformed and computed. Due to the linearity of f

$$\big\langle \eta^*, \frac{\partial}{\partial f}\left(f|_{t=t_0} - f_{t_0}\right)(f^*, \boldsymbol{\alpha^*}; h_f)\big\rangle_{L^2(\Omega\times\mathbb{R}^d)} = \big\langle \eta^*, h_f|_{t=t_0}\big\rangle_{L^2(\Omega\times\mathbb{R}^d)}$$

holds for all $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$.

Now, all summands of the adjoint equation are computed for the special choice of $\boldsymbol{G}$ and J. It reads

$$\begin{aligned}
&- \Big\langle \Big(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha^*}}{m} \cdot \boldsymbol{\nabla_v} + \frac{(\boldsymbol{u^*} - \boldsymbol{u}_{f^*})\,(\boldsymbol{v} - \boldsymbol{u}_{f^*})}{n_{f^*}} + dQ \Big)\, \varphi^*, h_f \Big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \\
&+ \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d)} + \big\langle \gamma^*, h_f(\boldsymbol{v}) - h_f(-\boldsymbol{v}) \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} \\
&+ \big\langle \varphi^*|_{t=t_1}, h_f|_{t=t_1} \big\rangle_{L^2(\Omega\times\mathbb{R}^d)} + \big\langle \eta^* - \varphi^*|_{t=t_0}, h_f|_{t=t_0} \big\rangle_{L^2(\Omega\times\mathbb{R}^d)} = 0
\end{aligned} \tag{5.19}$$

and must hold for every $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$.

The latter expression (5.19) for the adjoint equation can be simplified further. If $h_f \in C_0^\infty(I \times \Omega \times \mathbb{R}^d)$, all boundary integrals vanish and one gets the necessary condition

$$0 = \Big\langle \Big(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha^*}}{m} \cdot \boldsymbol{\nabla_v} + \frac{(\boldsymbol{u^*} - \boldsymbol{u}_{f^*})\,(\boldsymbol{v} - \boldsymbol{u}_{f^*})}{n_{f^*}} + dQ \Big)\, \varphi^*, h_f \Big\rangle_{L^2(I\times\Omega\times\mathbb{R}^d)}$$

which must hold for any $h_f \in C_0^\infty(I \times \Omega \times \mathbb{R}^d)$. In particular, this is true if for $(f^*, \boldsymbol{\alpha^*})$ holds

$$\Big(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha^*}}{m} \cdot \boldsymbol{\nabla_v} + \frac{(\boldsymbol{u^*} - \boldsymbol{u}_{f^*})\,(\boldsymbol{v} - \boldsymbol{u}_{f^*})}{n_{f^*}} + dQ \Big)\, \varphi^* = 0 \tag{5.20}$$

in $I \times \Omega \times \mathbb{R}^d$.

Then, if one assumes that $(f^*, \boldsymbol{\alpha^*})$ and φ^* satisfy condition (5.20), equation (5.19) reduces to

$$\big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d)} + \big\langle \gamma^*, h_f(\boldsymbol{v}) - h_f(-\boldsymbol{v}) \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} = 0 \tag{5.21}$$

as long as $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$ with $h_f|_{t=t_0} = h_f|_{t=t_1} = 0$. Choosing h_f such that it is symmetric, i.e. that it satisfies $h_f(\boldsymbol{v}) = h_f(-\boldsymbol{v})$ on $I \times \partial\Omega \times \mathbb{R}^d_{\boldsymbol{n}-}$, (5.19) further reduces to

$$\begin{aligned}
0 &= \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d)} \\
&= \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}+})} + \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} \\
&= \big\langle -\boldsymbol{v}\cdot\boldsymbol{n}\varphi^*(-\boldsymbol{v}), h_f(-\boldsymbol{v}) \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} + \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} \\
&= \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\,(\varphi^*(\boldsymbol{v}) - \varphi^*(-\boldsymbol{v}))\,, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} ,
\end{aligned}$$

whereby for the last transformation advantage is taken of the symmetry property of h_f. Since h_f, considered as function which operates on $I \times \Omega \times \mathbb{R}^d_{\boldsymbol{n}-}$, is arbitrary, φ^* must be symmetric as well, i.e. it must satisfy

$$\varphi^*(t, \boldsymbol{r}, \boldsymbol{v}) = \varphi^*(t, \boldsymbol{r}, -\boldsymbol{v}) \qquad\qquad (t, \boldsymbol{r}, \boldsymbol{v}) \in I \times \Omega \times \mathbb{R}^d_{\boldsymbol{n}-} . \tag{5.22}$$

Provided that φ^* is symmetric, condition (5.21) can be transformed as follows:

$$\begin{aligned}
0 &= \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d)} + \big\langle \gamma^*, h_f(\boldsymbol{v}) - h_f(-\boldsymbol{v}) \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} \\
&= \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^*, h_f(\boldsymbol{v}) - h_f(-\boldsymbol{v}) \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} + \big\langle \gamma^*, h_f(\boldsymbol{v}) - h_f(-\boldsymbol{v}) \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} \\
&= \big\langle \boldsymbol{v}\cdot\boldsymbol{n}\varphi^* + \gamma^*, h_f(\boldsymbol{v}) - h_f(-\boldsymbol{v}) \big\rangle_{L^2(I\times\partial\Omega\times\mathbb{R}^d_{\boldsymbol{n}-})} .
\end{aligned}$$

Since this condition must hold for all $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$ one obtains that

$$\gamma^* = -\boldsymbol{v} \cdot \boldsymbol{n} \varphi^* \qquad \text{on } I \times \Omega \times \mathbb{R}^d_{\boldsymbol{n}^-} \ .$$

Yet, since γ^* does not have to fulfil any further conditions, the latter condition can be dropped.

Provided $(f^*, \boldsymbol{\alpha^*})$ satisfies condition (5.20) and choosing $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$ such that $h_f|_{t=t_0} \equiv 0$ and $h_f|_{\partial\Omega} \equiv 0$ condition (5.19) reduces to

$$\left\langle \varphi^*|_{t=t_1}, h_f|_{t=t_1} \right\rangle_{L^2(\Omega \times \mathbb{R}^d)} = 0 \ .$$

Since $h_f|_{t=t_1} \in L^2(\Omega \times \mathbb{R}^d)$ is an arbitrary function one obtains the necessary condition

$$\varphi^*|_{t=t_1} = 0 \qquad \text{in } \Omega \times \mathbb{R}^d \ . \tag{5.23}$$

On the other hand, if this condition is fulfilled and again $(f^*, \boldsymbol{\alpha^*})$ satisfies condition (5.20) and $h_f \in H^1(I \times \Omega \times \mathbb{R}^d)$ such that $h_f|_{\partial\Omega} \equiv 0$, condition (5.19) reduces to

$$\left\langle \eta^* - \varphi^*|_{t=t_0}, h_f|_{t=t_0} \right\rangle_{L^2(\Omega \times \mathbb{R}^d)} = 0$$

which leads to

$$\eta^* = \varphi^*|_{t=t_0} \qquad \text{in } \Omega \times \mathbb{R}^d \ .$$

However, since no further condition must be fulfilled by η^* the latter condition can be dropped.

Combining the three obtained conditions, namely (5.20), (5.22) and (5.23), one obtains the following system of equations:

$$\begin{aligned}
-\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} \frac{B\boldsymbol{\alpha^*}}{m} \cdot \boldsymbol{\nabla_v}\right) \varphi^* &= dQ(\varphi^*) \\
&\quad + \frac{(\boldsymbol{u^*} - \boldsymbol{u}_{f^*})\,(\boldsymbol{v} - \boldsymbol{u}_{f^*})}{n_{f^*}} && \text{in } I \times \Omega \times \mathbb{R}^d \\
\varphi^*(\boldsymbol{v}) &= \varphi^*(-\boldsymbol{v}) && \text{on } I \times \partial\Omega \times \mathbb{R}^d_{\boldsymbol{n}^-} \\
\varphi^*|_{t=t_1} &= 0 && \text{in } \Omega \times \mathbb{R}^d \ .
\end{aligned} \tag{5.24}$$

Summarising the results obtained in this subsection, it has been shown that if (5.24) holds for a particular choice of $(f^*, \boldsymbol{\alpha^*})$ and φ^*, the adjoint equation (5.11) of the considered distributed control problems, whereby J and $\boldsymbol{G}$ are defined according to (5.8) and (5.7), will also hold true. Therefore, (5.24) sets up a necessary condition for the adjoint equation and with Theorem 5.1 also a necessary condition for an optimum of the underlying optimisation problem. In the following, (5.24) is referred to as *adjoint problem.*

5.2.3. Derivation of the Optimality Condition. Supposing all premises of Theorem 5.1 are fulfilled, then, beside the adjoint equation another necessary condition for an optimum $(f^*, \boldsymbol{\alpha^*})$ of the fluid flow control problem (5.1) is set up by the optimality equation (5.12). For the considered distributed control problem with the special choice for $\boldsymbol{G}$ and J it is required that

$$\begin{aligned}
0 = \nabla_{\boldsymbol{\alpha}} &\left(\frac{\alpha_{reg}}{2} \int_I \int_\Omega (B\boldsymbol{\alpha})^2 \, d\boldsymbol{r} dt\right) (f^*, \boldsymbol{\alpha^*}; \boldsymbol{h_\alpha}) \\
&+ \left\langle \varphi^*, \boldsymbol{\nabla_\alpha} \left(\frac{B\boldsymbol{\alpha}}{m} \cdot \boldsymbol{\nabla_v}\right) (f^*, \boldsymbol{\alpha^*}; \boldsymbol{h_\alpha}) \right\rangle_{L^2(I \times \Omega \times \mathbb{R}^d)}
\end{aligned} \tag{5.25}$$

holds for all directions $(0, \boldsymbol{h_\alpha}) \in H^1(I \times \Omega \times \mathbb{R}^d) \times U$. As in the previous section the dual space Y' is identified with $L^2(I \times \Omega \times \mathbb{R}^d) \times L^2(I \times \partial\Omega \times \mathbb{R}^d_{\boldsymbol{n}^-}) \times L^2(I \times \partial\Omega)$

and $\boldsymbol{\lambda}^* \in Y'$ is represented by a certain $(\varphi^*, \gamma^*, \eta^*)^T \in L^2(I \times \Omega \times \mathbb{R}^d) \times L^2(I \times \partial\Omega \times \mathbb{R}^d_{\boldsymbol{n}^-}) \times L^2(I \times \partial\Omega)$.

Condition (5.25) is simplified according to

$$\begin{aligned} 0 &= \alpha_{reg} \langle B\boldsymbol{\alpha}^*, B\boldsymbol{h}_{\boldsymbol{\alpha}} \rangle_{L^2(I\times\Omega)^d} + \langle \varphi^*, \frac{B\boldsymbol{h}_{\boldsymbol{\alpha}}}{m} \cdot \boldsymbol{\nabla}_{\boldsymbol{v}} f^* \rangle_{L^2(I\times\Omega\times\mathbb{R}^d)} \\ &= \alpha_{reg} \langle B\boldsymbol{\alpha}^* + \int_{\mathbb{R}^d} \frac{\varphi^*}{m} \boldsymbol{\nabla}_{\boldsymbol{v}} f^* \, d\boldsymbol{v}, B\boldsymbol{h}_{\boldsymbol{\alpha}} \rangle_{L^2(I\times\Omega)^d} \\ &= \alpha_{reg} \langle \boldsymbol{\alpha}^* + B' \int_{\mathbb{R}^d} \frac{\varphi^*}{m} \boldsymbol{\nabla}_{\boldsymbol{v}} f^* \, d\boldsymbol{v}, \boldsymbol{h}_{\boldsymbol{\alpha}} \rangle_U \end{aligned}$$

which holds for any $\boldsymbol{h}_{\boldsymbol{\alpha}} \in U$ whereby B' denotes the adjoint operator of B. Since $\boldsymbol{h}_{\boldsymbol{\alpha}} \in U$ is arbitrary the latter condition further simplifies to

$$\boldsymbol{\alpha}^* = -\frac{1}{\alpha_{reg}} B' \int_{\mathbb{R}^d} \frac{\varphi^*}{m} \boldsymbol{\nabla}_{\boldsymbol{v}} f^* \, d\boldsymbol{v} \qquad \text{in } I \times \Omega \; . \tag{5.26}$$

This is another necessary condition for an optimum $(f^*, \boldsymbol{\alpha}^*)$ of the considered distributed control problems. It is referred to as *optimality condition*.

With this last condition all three conditions that build the first-order necessary optimality system for the considered distributed control problems are given. In order to summarise the results of the whole section, in the following subsection the corresponding optimality system is stated.

5.2.4. A First-order Necessary Optimality System for Distributed Control Problems. The following system of equations constitute a necessary condition for an optimum $(f^*, \boldsymbol{\alpha}^*)$ of a distributed control problem which is given according to (5.1) whereby J and $\boldsymbol{G}$ are defined by (5.8) and (5.7), respectively. It is established by the results presented in the previous three subsections, namely by the primal problem (5.13), the adjoint problem (5.24) and the optimality condition (5.26). It reads

$$\begin{aligned} \left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla}_{\boldsymbol{r}} + \frac{B\boldsymbol{\alpha}^*}{m} \cdot \boldsymbol{\nabla}_{\boldsymbol{v}}\right) f^* &= Q(f^*) && \text{in } I \times \Omega \times \mathbb{R}^d \\ f^*(\boldsymbol{v}) &= f^*(-\boldsymbol{v}) && \text{on } I \times \partial\Omega \times \mathbb{R}^d_{\boldsymbol{n}^-} \\ f^*|_{t=t_0} &= f_{t_0} && \text{in } \Omega \times \mathbb{R}^d \\ -\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla}_{\boldsymbol{r}} + \frac{B\boldsymbol{\alpha}^*}{m} \cdot \boldsymbol{\nabla}_{\boldsymbol{v}}\right) \varphi^* &= dQ(\varphi^*) && \\ &\quad + \frac{(\boldsymbol{u}^* - \boldsymbol{u}_{f^*})(\boldsymbol{v} - \boldsymbol{u}_{f^*})}{n_{f^*}} && \text{in } I \times \Omega \times \mathbb{R}^d \\ \varphi^*(\boldsymbol{v}) &= \varphi^*(-\boldsymbol{v}) && \text{on } I \times \partial\Omega \times \mathbb{R}^d_{\boldsymbol{n}^-} \\ \varphi^*|_{t=t_1} &= 0 && \text{in } \Omega \times \mathbb{R}^d \\ -\frac{1}{\alpha_{reg}} B' \int_{\mathbb{R}^d} \frac{\varphi^*}{m} \boldsymbol{\nabla}_{\boldsymbol{v}} f^* \, d\boldsymbol{v} &= \boldsymbol{\alpha}^* && \text{in } I \times \Omega \; . \end{aligned} \tag{5.27}$$

5.3. The Adjoint BGK-Boltzmann Equation

This section is devoted to derive the prototypical primary equation of the adjoint equation (5.4) which belongs to an optimisation problem whose side condition is governed by a BGK-Boltzmann equation akin (1.67). This resulting equation is in the following referred to as *adjoint BGK-Boltzmann equation*.

In order to derive the adjoint BGK-Boltzmann equation, only that part of the side condition $\boldsymbol{G}(f(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = \boldsymbol{0}$ $(\boldsymbol{\alpha} \in U)$ is considered at which a BGK-Boltzmann equation in the form of (1.67) enters. Let the particular part be fixed to be G_1. All further conditions are assumed to be fulfilled. Without loss of generality, it is assumed that the control $\boldsymbol{\alpha}$ enters the BGK-Boltzmann equation in form of substituting the force term $\boldsymbol{F}$ by $B\boldsymbol{\alpha}$ whereby B is an appropriate control extension operator. Hence, G_1 is defined as in (5.7), which is the governing part of the side condition of the distributed control problem studied in Section 5.2. The corresponding adjoint variable is $\lambda_1 =: \varphi$. Thus, the adjoint equation (5.4), which belongs to a particular $\boldsymbol{\alpha} \in U$ and with it also to a particular $f := f(\boldsymbol{\alpha})$, reduces to

$$\frac{\partial}{\partial f} J(f,\boldsymbol{\alpha}) = -\varphi \, \frac{\partial}{\partial f} \left(\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha}}{m} \cdot \boldsymbol{\nabla_v} \right) - Q \right) (f) \tag{5.28}$$

which must hold in $I \times \Omega \times \mathbb{R}^d$. Assuming $\varphi \in H^1(I \times \Omega \times \mathbb{R}^d)$, with the help of a duality argument a necessary condition for (5.28) can be derived as in Subsection 5.2.2 which yields

$$dJ_f = \left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla_r} + \frac{B\boldsymbol{\alpha}}{m} \cdot \boldsymbol{\nabla_v} + dQ \right) \varphi \qquad \text{in } I \times \Omega \times \mathbb{R}^d \tag{5.29}$$

whereby $dJ_f := \frac{\partial}{\partial f} J(f,\boldsymbol{\alpha})$. The term dQ is defined according to

$$dQ(\varphi) : \begin{cases} I \times \Omega \times \mathbb{R}^d & \to \mathbb{R} \\ (t,\boldsymbol{r},\boldsymbol{v}) & \mapsto -\dfrac{1}{\omega} \left(\varphi(t,\boldsymbol{r},\boldsymbol{v}) - dM_f^{eq}(t,\boldsymbol{r},\boldsymbol{v}) \right) \end{cases} \tag{5.30}$$

with

$$dM_f^{eq} : \begin{cases} I \times \Omega \times \mathbb{R}^d & \to \mathbb{R} \\ (t,\boldsymbol{r},\boldsymbol{v}) & \mapsto \displaystyle\int_{\mathbb{R}^d} \varphi(\hat{\boldsymbol{v}}) \frac{((\boldsymbol{u_f} - \boldsymbol{v})\,(\boldsymbol{u_f} - \hat{\boldsymbol{v}}) + RT)}{n_f RT} M_f^{eq}(\hat{\boldsymbol{v}}) \; d\hat{\boldsymbol{v}} \end{cases} \tag{5.31}$$

and with ω, R, T and M_f^{eq} defined as in the corresponding side condition function G_1 which is a BGK-Boltzmann equation similar to (1.67).

5.4. Adjoint Lattice Boltzmann Methods

An adjoint BGK-Boltzmann equation like (5.29) is an affine-linear integro partial differential equation. It is structured similarly to BGK-Boltzmann equations like (1.67). Therefore, applying discretisation strategies which are similar to LBM as they are studied in Section 2.1 of Chapter 2 seems to be promising. Thus, the subject of this section is to derive such discretisation methods for adjoint BGK-Boltzmann equations. The obtained strategies are referred to as *adjoint lattice Boltzmann methods* (ALBM) in the following.

The basic and common idea of all LBM and consequently also of all ALBM is the coupling of a discretisation parameter $h \in \mathbb{R}_{>0}$ with one or more scaling parameters of a Boltzmann or a Boltzmann-like equation. The modality of the connection depends on the regime in which an underlying macroscopic target equation is reached by the considered family of mesoscopic governing equations. In Chapter 2 the subclass of such LBM is considered which leads to LB equations whereas the target equation of choice is the incompressible Navier-Stokes equation (1.34). Continuing on this path, in the following only those ALBM are studied which are associated with this subclass of LBM.

The here presented discretisation approach is structured as the one for LBM which is presented in Section 2.1. As well, it consists of three main steps. At first, the underlying discrete time and phase space are defined. Then, the adjoint Lattice BGK-Boltzmann equation is considered in a finite velocity space which leads to an equation which is said to be the *velocity discrete adjoint BGK-Boltzmann equation* in the following. Finally, the transient position space is replaced by a discrete space. This leads to an equation which is referred to as the *adjoint Lattice Boltzmann equation* (ALB equation).

In this context, it is to be noted that the procedure of considering a family of BGK-Boltzmann equations in a regime which leads to an incompressible Navier-Stokes equation (cf. Chapter 1) might also be applied to link a family of adjoint BGK-Boltzmann equations to a particular adjoint incompressible Navier-Stokes equation. The results of the investigated example which are presented in Section 5.6 point in this direction. Providing that this relation can be confirmed, the question will arise if a family of ALB equations can be directly linked to this adjoint incompressible Navier-Stokes equation. This, however, is not subject of the following investigations. In fact, the discretisation methods presented in this section aim to derive discrete equations which are consistent to their continuous counterpart. The way the discretisation parameter h is coupled to a particular adjoint BGK-Boltzmann equation asks for a definition of the term consistency. It can be defined as for the derivation of LBM which is stated in Definition 2.1. However, in contrast to the established consistency of LB equations it turns out that the consistency of ALB equations cannot be proved without any further assumptions concerning the smoothness of the solution of the corresponding adjoint BGK-Boltzmann equation. Nevertheless, the results of the numerical experiments, which are presented in Section 5.6, clearly encourage further investigations of the proposed approaches.

5.4.1. Discrete Time and Phase Space. The transient phase space $I \times \Omega \times \mathbb{R}^d$ is discretised by $I_h \times \Omega_h \times Q$ exactly as described in Subsection (2.1.1). Thereby, $h \in \mathbb{R}_{>0}$ denotes the discretisation parameter. As for the LBM the particular choice of $I_h \times \Omega_h \times Q$ sets up an ALBM model which is denoted by $DdQq$ with d representing the dimension and q the cardinal number of $Q \subseteq \mathbb{R}^d$. Two commonly applied models are the $D2Q9$ and the $D3Q19$ model. They are both explicitly listed in Subsection (2.1.1). The $D3Q27$ model is another important model for the derivation to come. Its definition can be found in the literature, e.g. in [**28, 129, 140**].

5.4.2. Velocity Discrete Adjoint BGK-Boltzmann Equations. Starting point for the derivation is an adjoint BGK-Boltzmann equation which is defined as (5.29). Thereby, $\boldsymbol{\alpha} \in U$ is assumed to be fixed. With the definition of the side condition $G_1(f^h(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ also $h \in \mathbb{R}_{>0}$, $\nu \in \mathbb{R}_{>0}$ and its solution $f^h := f^h(\boldsymbol{\alpha})$ are given. The coupling between the discretisation parameter h and the considered adjoint BGK-Boltzmann equation is realised such as the coupling between h and a BGK-Boltzmann equation, namely by the settings (2.1) and (2.2). Then, substituting RT and ω, multiplying (5.29) with h^2, subtracting dQ on both sides and shortening the notation by taking advantage of Lagrange's description (5.29) becomes

$$h^2 \frac{d}{dt}\varphi = \frac{1}{3\nu}\left(\varphi - dM^{eq}_{f^h}\right) + \frac{\partial}{\partial f} J(f^h, \boldsymbol{\alpha}) \qquad \text{in } I \times \Omega \times \mathbb{R}^d \qquad (5.32)$$

with

$$dM^{eq}_{f^h} = \int_{\mathbb{R}^d} \varphi(\hat{\boldsymbol{v}}) \frac{\left(3h^2 \left(\boldsymbol{u_{f^h}} - \boldsymbol{v}\right)\left(\boldsymbol{u_{f^h}} - \hat{\boldsymbol{v}}\right) + 1\right)}{n_{f^h}} M^{eq}_{f^h}(\hat{\boldsymbol{v}})\ d\hat{\boldsymbol{v}}\ . \tag{5.33}$$

Since (5.32) comprises an h a corresponding solution will generally also depend on h. For the remainder of this subsection it is assumed that such a solution exists and that it is unique. To indicate the dependence on h, it is denoted by φ^h.

Let φ^h be the solution of (5.32) and defining in $I \times \Omega$ for $\boldsymbol{v}, \hat{\boldsymbol{v}} \in \mathbb{R}^d$

$$a(\boldsymbol{v}, \hat{\boldsymbol{v}}) := \varphi^h(\hat{\boldsymbol{v}}) \left(3h^2 \left(\boldsymbol{u_{f^h}} - \boldsymbol{v}\right)\left(\boldsymbol{u_{f^h}} - \hat{\boldsymbol{v}}\right) + 1\right)$$

to shorten the notation. Then, $dM^{eq}_{f^h}$ can be transformed according to

$$\begin{aligned} dM^{eq}_{f^h} &= \int_{\mathbb{R}^d} a(\boldsymbol{v}, \hat{\boldsymbol{v}}) \frac{h^d}{\left(\frac{2}{3}\pi\right)^{d/2}} \exp\left(-\frac{3}{2}\left(\hat{\boldsymbol{v}}\ h - \boldsymbol{u_{f^h}}\ h\right)^2\right)\ d\hat{\boldsymbol{v}} \\ &= \int_{\mathbb{R}^d} a(\boldsymbol{v}, \hat{\boldsymbol{v}}) \frac{h^d}{\left(\frac{2}{3}\pi\right)^{d/2}} \exp\left(-\frac{3}{2}h^2\hat{\boldsymbol{v}}^2\right) \exp\left(3h^2\ \hat{\boldsymbol{v}} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2}h^2 {\boldsymbol{u_{f^h}}}^2\right)\ d\hat{\boldsymbol{v}} \\ &= \int_{\mathbb{R}^d} a(\boldsymbol{v}, \frac{\sqrt{2}}{h\sqrt{3}}\overline{\boldsymbol{v}}) \frac{1}{(\pi)^{d/2}} \exp\left(-\overline{\boldsymbol{v}}^2\right) \exp\left(\sqrt{6}h\ \overline{\boldsymbol{v}} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2}h^2 {\boldsymbol{u_{f^h}}}^2\right)\ d\overline{\boldsymbol{v}}\ , \end{aligned} \tag{5.34}$$

in $I \times \Omega$ and for all $\boldsymbol{v} \in \mathbb{R}^d$ with the substitution $\overline{\boldsymbol{v}} := \sqrt{\frac{3}{2}}h\hat{\boldsymbol{v}}$. Applying a Gauss-Hermite quadrature of order three (cf. for example Hämmerlin [**58**]) one obtains in $I \times \Omega \times \mathbb{R}^d$

$$dM^{eq}_{f^h} = \sum_{i=0}^{q-1} a(\boldsymbol{v}, \boldsymbol{v_i}) \frac{w_i}{w} \exp\left(3h^2\ \boldsymbol{v_i} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2}h^2 {\boldsymbol{u_{f^h}}}^2\right) + R_{10,t,\boldsymbol{r},\boldsymbol{v}} \tag{5.35}$$

whereby for $d = 2$ the terms v_i, w_i and w are defined as for the $D2Q9$ model which is stated in Subsection 2.1.1 and Subsection 2.1.2. Similarly, for $d = 3$ one can derive v_i, w_i and w which leads to the $D3Q27$ model. The quadrature error $R_{10,t,\boldsymbol{r},\boldsymbol{v}}$ incorporates the derivatives of φ^h with respect to $\boldsymbol{v}$ up to order six. Without any further assumption concerning the asymptotic behaviour of φ^h and its derivatives, an analytical estimate of the quadrature error seems to be hard to find. However, the numerical experiments discussed in Section 5.6 suggest that this error vanishes as $h \to 0$.

Taking advantage of the approximations and definitions stated in Subsection 2.1.2, the summands of (5.35) can be transformed according to

$$\begin{aligned} &a(\boldsymbol{v_j}, \boldsymbol{v_i}) \frac{w_i}{w} \exp\left(3h\ \tilde{\boldsymbol{v}}_{\boldsymbol{i}} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2}h^2 {\boldsymbol{u_{f^h}}}^2\right) \\ &= a(\boldsymbol{v_j}, \boldsymbol{v_i}) \frac{w_i}{w} \left(\left(1 + 3h\ \tilde{\boldsymbol{v}}_{\boldsymbol{i}} \cdot \boldsymbol{u_{f^h_i}} - \frac{3}{2}h^2 {\boldsymbol{u_{f^h_i}}}^2 + \frac{9}{2}h^2\left(\tilde{\boldsymbol{v}}_{\boldsymbol{i}} \cdot \boldsymbol{u_{f^h_i}}\right)^2\right) + R_{11,t,\boldsymbol{r},\boldsymbol{v_i}}\right) \\ &= \varphi^h(\boldsymbol{v_i}) \frac{\left(3\left(h\boldsymbol{u_{f^h}} - \tilde{\boldsymbol{v}}_{\boldsymbol{j}}\right)\left(h\boldsymbol{u_{f^h}} - \tilde{\boldsymbol{v}}_{\boldsymbol{i}}\right) + 1\right)}{n_{f^h_i}} M^{eq}_{f^h_i} + \varphi^h(\boldsymbol{v_i}) R_{12,t,\boldsymbol{r},\boldsymbol{v_i}} \\ &= \varphi^h(\boldsymbol{v_i}) \frac{\left(3\left(h\boldsymbol{u_{f^h_i}} - \tilde{\boldsymbol{v}}_{\boldsymbol{j}}\right)\left(h\boldsymbol{u_{f^h_i}} - \tilde{\boldsymbol{v}}_{\boldsymbol{i}}\right) + 1\right)}{n_{f^h_i}} M^{eq}_{f^h_i} + R_{13,t,\boldsymbol{r},\boldsymbol{v_i}}\ . \end{aligned} \tag{5.36}$$

The equalities will hold true in $I \times \Omega$ for all $\boldsymbol{v_i}, \boldsymbol{v_j} \in Q$ if the conditions of Theorem 2.2 are fulfilled. The first equality holds because the exponential term is at first expanded in a Taylor series in h which leaves a truncation error of order $\mathcal{O}(h^3)$. Then, the integral term $\boldsymbol{u_{f^h}}$ is replaced by the sum $\boldsymbol{u_{f_i^h}}$. In accordance with (2.6) this leaves the truncation error to be $R_{11,t,\boldsymbol{r},\boldsymbol{v_i}} \in \mathcal{O}(h^2)$. The next equality is obtained by inserting $n_{f_i^h}/n_{f_i^h}$ so that a term turns out to be identical with $M^{eq}_{f_i^h}$. Further, $a(\boldsymbol{v_j}, \boldsymbol{v_i})$ is re-substituted and the truncation error is separated. Since $\left(3\left(h\boldsymbol{u_{f^h}} - \tilde{\boldsymbol{v}}_{\boldsymbol{j}}\right)\left(h\boldsymbol{u_{f^h}} - \tilde{\boldsymbol{v}}_{\boldsymbol{i}}\right) + 1\right) \in \mathcal{O}(1)$ the truncation error $R_{12,t,\boldsymbol{r},\boldsymbol{v_i}} \in \mathcal{O}(h^2)$. If $\varphi^h(\boldsymbol{v_i}) \in \mathcal{O}(1)$ for all $\boldsymbol{v_i} \in Q$ $R_{13,t,\boldsymbol{r},\boldsymbol{v_i}}$ will be a function of $\mathcal{O}(h^2)$ because the replacement of $\boldsymbol{u_{f^h}}$ by $\boldsymbol{u_{f_i^h}}$ just leaves another truncation error of order two which can be included in $R_{13,t,\boldsymbol{r},\boldsymbol{v_i}}$. To be stressed is that the latter condition sets up another assumption on the solution φ^h.

The velocity discrete operator $dM^{eq}_{f_i^h}$ which belongs to $dM^{eq}_{f^h}$ is defined in $I \times \Omega \times Q$. By setting $\varphi_i^h(t, \boldsymbol{r}) := \varphi^h(t, \boldsymbol{r}, \boldsymbol{v_i})$ for all $t \in I$, $\boldsymbol{r} \in \Omega$ and $\boldsymbol{v_i} \in Q$ it reads

$$dM^{eq}_{f_i^h}(\boldsymbol{v_j}) := \sum_{i=0}^{q-1} \varphi_i^h(\boldsymbol{v_i}) \frac{\left(3\left(h\boldsymbol{u_{f_i^h}} - \tilde{\boldsymbol{v}}_{\boldsymbol{j}}\right)\left(h\boldsymbol{u_{f_i^h}} - \tilde{\boldsymbol{v}}_{\boldsymbol{i}}\right) + 1\right)}{n_{f_i^h}} M^{eq}_{f_i^h} \tag{5.37}$$

for all $\boldsymbol{v_i} \in Q$ in $I \times \Omega$. With (5.35) and (5.36) the truncation error yields

$$dM^{eq}_{f^h} - dM^{eq}_{f_i^h} = R_{10,t,\boldsymbol{r},\boldsymbol{v}_j} + \sum_{i=0}^{q-1} R_{13,t,\boldsymbol{r},\boldsymbol{v}_i} \in \mathcal{O}(h^2) \qquad \text{in } I \times \Omega \times Q \tag{5.38}$$

providing that $\varphi_i^h \in \mathcal{O}(1)$ and that $R_{11,t,\boldsymbol{r},\boldsymbol{v_j}} \in \mathcal{O}(h^2)$ for $i, j = 0, 1, ..., q-1$ in $I \times \Omega$.

Since the derivative dJ_{f^h} of the goal functional J does not depend on φ the term dJ_{f^h} generates in principle no further truncation errors. However, it depends on a certain f^h which is the solution of a BGK-Boltzmann equation. If an LBM is to be applied to solve it, just certain f^h, namely f_i^h $(i = 0, 1, ..., q-1)$, will be known. Thus, also dJ_{f^h} needs to be replaced by a function which just depends on f_i^h. Consequently, this function is denoted by $dJ_{f_i^h}$. The form of J and with it of dJ_{f^h} strongly depends on the actual considered optimisation problem. It is to be noted that in general the truncation error $dJ_{f^h} - dJ_{f_i^h} =: R_{14,t,\boldsymbol{r},\boldsymbol{v}_j}$ should be of a particular order which preserves the total consistency.

Considering the distributed control problem introduced in Section 5.2 at which the goal function is defined as in (5.8), according to (5.18) its derivative is given by

$$dJ_{f^h} = -\frac{\left(\boldsymbol{u}^* - \boldsymbol{u}_{f^h}\right)\left(\boldsymbol{v} - \boldsymbol{u}_{f^h}\right)}{n_{f^h}} \qquad \text{in } I \times \Omega \times \mathbb{R}^d \; .$$

Then, replacing n_{f^h} and $\boldsymbol{u}_{f^h}$ by $n_{f_i^h}$ and $\boldsymbol{u}_{f_i^h}$ leads to

$$dJ_{f_i^h} = -\frac{\left(\boldsymbol{u}^* - \boldsymbol{u}_{f_i^h}\right)\left(\boldsymbol{v_j} - \boldsymbol{u}_{f_i^h}\right)}{n_{f_i^h}} \qquad \text{in } I \times \Omega \times Q \; . \tag{5.39}$$

Under the condition of Theorem 2.2 the truncation error is

$$R_{14,t,\boldsymbol{r},\boldsymbol{v}_j} = dJ_{f^h} - dJ_{f_i^h} \in \mathcal{O}(1) \qquad \text{in } I \times \Omega \times Q$$

which is founded on (2.5), (2.6) and the fact that $\boldsymbol{v_j} \in \mathcal{O}(h^{-1})$ for $j = 0, 1, ..., q-1$.

Finally, one obtains the *velocity discrete adjoint BGK-Boltzmann equation* as a set of q coupled equations according to

$$h^2 \frac{d}{dt}\varphi_j = \frac{1}{3\nu}\left(\varphi_j - dM^{eq}_{f_i^h}\right) + h^2 dJ_{f_i^h} \qquad \text{in } I \times \Omega \times Q \; . \tag{5.40}$$

The total truncation error of its family with respect to the family of adjoint BGK-Boltzmann equation in the form of (5.32) is given by

$$\frac{1}{3\nu} R_{10,t,\boldsymbol{r},\boldsymbol{v_j}} + \frac{1}{3\nu} \sum_{i=0}^{q-1} R_{13,t,\boldsymbol{r},\boldsymbol{v_i}} + h^2 R_{14,t,\boldsymbol{r},\boldsymbol{v_j}} \qquad \text{in } I \times \Omega \times Q \ . \tag{5.41}$$

Assuming that $R_{10,t,\boldsymbol{r},\boldsymbol{v}_j} \in \mathcal{O}(h^2)$ and $\varphi_i^h, R_{14,t,\boldsymbol{r},\boldsymbol{v_j}} \in \mathcal{O}(1)$ hold in $I \times \Omega \times Q$ and providing further that the conditions of Theorem 2.2 are fulfilled, then, this truncation error will be of second order for the $D2Q9$ and $D3Q27$ model. It is to be noted that the condition $R_{14,t,\boldsymbol{r},\boldsymbol{v_j}} \in \mathcal{O}(1)$ in $I \times \Omega \times Q$ especially holds true if the goal functional is defined as in (5.8).

5.4.3. Adjoint Lattice Boltzmann Equations. After deriving the family of velocity discrete adjoint BGK-Boltzmann equations (5.40), now, the time and position space $I \times \Omega$ is to be replaced by a discrete space $I_h \times \Omega_h$. This will lead to equations which are referred to as adjoint Lattice Boltzmann equations. It is stressed that this approach is similar to that presented in Subsection (2.1.3) where consistent LB equations are derived from velocity discrete BGK-Boltzmann equations. As there, first of all, the force term is assumed to vanish, i.e. $\boldsymbol{F} = \boldsymbol{0}$ in $I \times \Omega$. However, in the following subsection a possible treatment is proposed.

To continue the discretisation process, let the definitions which are introduced in the previous subsection also apply here. Recall, φ^h denotes the assumed solution of (5.32) and f^h denotes the assumed solution of the corresponding BGK-Boltzmann equation.

If one assumes that φ^h satisfies $\frac{d^3}{dt^3}\varphi_j^h \in \mathcal{O}(1)$ for $j = 0, 1, ..., q-1$, with Taylor's theorem one gets a central difference approximation of $\frac{d}{dt}\varphi_j^h$ according to

$$h^2 \frac{d}{dt}\varphi_j^h(t - h^2/2) = \varphi_j^h(t) - \varphi_j^h(t - h^2) + R_{15,t}(h) \qquad \text{in } I_h \ , \tag{5.42}$$

whereby for the remainder term holds $R_{15,t}(h) \in \mathcal{O}(h^6)$ for all $t \in I_h$.

Providing $\frac{d^2}{dt^2}\varphi_j^h \in \mathcal{O}(1)$ for $j = 0, 1, ..., q-1$, similarly to (2.13) one obtains

$$\varphi_j^h(t - h^2/2) = \varphi_j^h(t) + \frac{1}{2}\left(\varphi_j^h(t - h^2) - \varphi_j^h(t)\right) + R_{16,t}(h) \qquad \text{in } I_h \tag{5.43}$$

whereby $R_{16,t}(h) \in \mathcal{O}(h^4)$ for all $t \in I_h$.

Then, as in (2.14) for $M_{f_i^h}^{eq}$, $dM_{f_i^h}^{eq}$ needs to be shifted by $h^2/2$. Applying Taylor's expansion technique,

$$dM_{f_i^h}^{eq}(t - h^2/2) = dM_{f_i^h}^{eq}(t) + R_{17,t}(h) \qquad \text{in } I_h \tag{5.44}$$

holds with $R_{17,t}(h) \in \mathcal{O}(h^2)$ for $t \in I_h$ if $\frac{d}{dt}dM_{f_i^h}^{eq}(t) \in \mathcal{O}(1)$ for $t \in I_h$. This is in particular the case if $\varphi_j^h, \frac{d}{dt}\varphi_j^h, f_i^h, \frac{d}{dt}f_i^h \in \mathcal{O}(1)$ for all $t \in I_h$.

Similarly to $dM_{f_i^h}^{eq}$, the term $h^2 dJ_{f_i^h}$ is to be shifted by $h^2/2$. Again with Taylor's expansion technique it holds

$$h^2 dJ_{f_i^h}(t - h^2/2) = h^2 dJ_{f_i^h}(t) + R_{18,t}(h) \qquad \text{in } I_h \tag{5.45}$$

with $R_{18,t}(h) \in \mathcal{O}(h^2)$ if $h^2 \frac{d}{dt} dJ_{f_i^h}(t) \in \mathcal{O}(1)$ for $t \in I_h$. In general J and with it $dJ_{f_i^h}$ depend on the actually considered fluid flow optimisation problem. A particular goal functional is defined in (5.8). As stated in (5.39), a possible consistent approximation of dJ_{f^h} is $dJ_{f_i^h}$. If one furthermore assumes that $f_i^h, \frac{d}{dt}f_i^h \in \mathcal{O}(1)$ for all $t \in I_h$ and $\boldsymbol{u}^*$ is differentiable, one will obtain that the truncation error $R_{18,t}(h)$ is a function of $\mathcal{O}(h^2)$.

To obtain the ALB equation, at first, the velocity discrete adjoint BGK-Boltzmann equation (5.40) is multiplied with $\frac{6\nu}{6\nu+1}$. Then, all terms are substituted according to (5.42), (5.43), (5.44) and (5.45). This yields

$$\begin{aligned}
&\frac{6\nu}{6\nu+1}\Big(h^2\frac{d}{dt}\hat{v}\varphi_j(t-h^2/2)-\frac{1}{3\nu}\Big(\varphi_j^h(t-h^2/2)-dM^{eq}_{f_i^h}(t-h^2/2)\Big)\\
&\quad -h^2dJ_{f_i^h}(t-h^2/2)\Big)\\
&\quad =\frac{6\nu}{6\nu+1}\Big(f_i^h(t)-\varphi_j^h(t-h^2)-\frac{1}{3\nu}\Big(\varphi_j^h(t)+\frac{1}{2}\left(\varphi_j^h(t-h^2)-\varphi_j^h(t)\right)\\
&\quad -dM^{eq}_{f_i^h}(t)\Big)-h^2dJ_{f_i^h}(t)\Big)+R_{19,t}(h)\\
&\quad =f_i^h(t)-\varphi_j^h(t-h^2)-\frac{1}{3\nu+1/2}\Big(\varphi_j^h(t)-dM^{eq}_{f_i^h}(t)\Big)\\
&\quad +\frac{6\nu}{6\nu+1}h^2dJ_{f_i^h}(t)+R_{19,t}(h)
\end{aligned}$$

which holds for $j=0,1,...,q-1$ in I_h.

$$R_{19,t}(h)=\frac{6\nu}{6\nu+1}\Big(R_{15,t}(h)-\frac{1}{3\nu}\left(R_{16,t}(h)-R_{17,t}(h)\right)+R_{18,t}(h)\Big)\qquad\text{in } I_h$$

which is under the before mentioned conditions of second order.

Based on this derivation, the *adjoint lattice Boltzmann equation* (ALB equation) can be stated. It reads

$$\begin{aligned}
\varphi_j(t)-\varphi_j(t-h^2)=\frac{1}{3\nu+1/2}&\Big(\varphi_j(t)-dM^{eq}_{f_i^h}(t)\Big)\\
&-\frac{6\nu}{6\nu+1}h^2dJ_{f_i^h}(t)\quad\text{for } t\in I_h,\ j=0,1,...,q-1\ .
\end{aligned}\tag{5.46}$$

The latter truncation error $R_{19,t}$ is as well the truncation error which occurs then considering the consistency of the family of ALB equations (5.46) to the family of the velocity discrete adjoint BGK-Boltzmann equations (5.40). Summarising the results, the order of consistency will be two if $h^2\frac{d}{dt}dJ_{f_i^h}(t)$, f_i^h, $\frac{d}{dt}f_i^h$, $\frac{d^k}{dt^k}\varphi_j^h\in\mathcal{O}(1)$ for $k=0,1,2,3$ and $i,j=0,1,...,q-1$ in I_h. In particular, if the goal functional is defined as in (5.8) and f_i^h fulfils the conditions of Theorem 2.2, it will hold that $h^2\frac{d}{dt}dJ_{f_i^h}(t)\in\mathcal{O}(1)$.

5.4.4. Treatment of External Forces. In the previous two subsections the velocity discrete adjoint BGK-Boltzmann equation and the ALB equation are derived. Thereby, the force term is assumed to vanish, i.e. $\boldsymbol{F}=\boldsymbol{0}$ in $I\times\Omega$. Yet, in many applications it needs to be considered. In the following, a way is sketched how a force term can be integrated, firstly in the LB equation and secondly in the ALB equation.

In the framework of LBM, there exist various approaches to take an external force into consideration. It is common practice to add another term T_i ($i=0,1,...,q-1$) to the right hand side of the lattice BGK-Boltzmann equation (2.15). The T_i ($i=0,1,...,q-1$) represent a source which is chosen such that their moments represent dedicated macroscopic quantities. An overview is given by Guo et al. in [**57**] and Buick et al. in [**23**]. The basic idea of one of these approaches is stated in [**59**]. It relies on replacing $\frac{\boldsymbol{F}}{m}\cdot\boldsymbol{\nabla_v}f^h$ by $\frac{\boldsymbol{F}}{m}\cdot\boldsymbol{\nabla_v}M^{eq}_{f^h}$. Then, $\boldsymbol{\nabla_v}M^{eq}_{f^h}$ can be computed and with it also its moments. With Gauss-Hermite quadrature one gets q supporting points v_i and weights w_i ($i=0,1,...,q-1$) for the $D2Q9$ and

$D3Q27$ model. For the $D3Q19$ model v_i and w_i $i = 0, 1, ..., q-1$ are chosen such that the same macroscopic quantities are obtained. The supporting points v_i and weights w_i $(i = 0, 1, ..., q-1)$ and also the term w are just the same as the one obtained then deriving the velocity discrete adjoint BGK-Boltzmann equation (cf. Subsection 2.1.2). With these definitions, T_i reads

$$T_i := 3h \, \frac{w_i}{w} n_{f_i^h} \frac{\boldsymbol{F}}{m} \cdot \tilde{\boldsymbol{v}}_{\boldsymbol{i}} \qquad \text{for } i = 0, 1, ..., q-1 \; . \tag{5.47}$$

It is to be noted that in contrast to most other approaches (cf. [**23, 57, 59**]) the macroscopic quantities are not scaled. Therefore, an additional factor h appears as part of all terms T_i $(i = 0, 1, ..., q-1)$.

To incorporate an external force in the ALB equation (5.46) the adjoint operator dT to T_i $(i = 0, 1, ..., q-1)$ is computed on a discrete base. Therefore, all T_i $(i = 0, 1, ..., q-1)$ are differentiated with respect to f_i^h into an arbitrary direction h_{f_i}:

$$\begin{aligned} \big\langle \varphi_i, \frac{d}{df} T_i(f_i^h; h_{f_i}) \big\rangle_{L^2(I_h \times \Omega_h \times Q)} &= \big\langle \varphi_i, 3h \, \frac{w_i}{w} n_{h_{f_i}} \frac{\boldsymbol{F}}{m} \cdot \tilde{\boldsymbol{v}}_{\boldsymbol{i}} \big\rangle_{L^2(I_h \times \Omega_h \times Q)} \\ &= \big\langle \underbrace{3h \frac{\boldsymbol{F}}{m} \cdot \sum_{i=0}^{q-1} \frac{w_i}{w} \varphi_i \tilde{\boldsymbol{v}}_{\boldsymbol{i}}}_{=:dT}, \widehat{h}_{f_i} \big\rangle_{L^2(I_h \times \Omega_h \times \widehat{Q})} \; . \end{aligned}$$

These terms are added on the right hand side of the ALB equation (5.46).

5.5. Implementation of Adjoint Lattice Boltzmann Algorithms

It is important to note that the structure of an ALB equation like (5.46) is very similar to that of an LB equation like (2.15). Main differences are its time inverse character and the additional term $\frac{6\nu}{6\nu+1} h^2 dJ_{f_i^h}$. However, its locality properties remain basically the same. This encourages to implement ALBM similarly to that strategy for LBM which is presented and discussed in Section 2.2. Moreover, also the adoption of the hybrid parallelisation strategy, which is presented and tested in Chapter 3, also seems possible.

In the following, at first an ALB algorithm is derived. Then, differences and similarities with respect to an LB algorithms are studied. In particular, it is checked if both the sequential and hybrid parallel implementation strategy dedicated for LB algorithms can also be applied for ALB algorithms.

Starting with (5.46), an iterative algorithm can be derived. It is executed step by step for decreasing $t \in I_h$. In each single time step two steps are to be performed for all $r \in \Omega_h$ and every $j = 0, 1, ..., q-1$, namely the *adjoint collision step*

$$\widetilde{\varphi}_j(t, \boldsymbol{r}) = \varphi_j(t, \boldsymbol{r}) - \frac{1}{3\nu + 1/2} \Big(\varphi_j(t, \boldsymbol{r}) - dM_{f_i}^{eq}(t, \boldsymbol{r}) \Big) + \frac{6\nu}{6\nu + 1} h^2 dJ_{f_i}(t) \tag{5.48}$$

and the *adjoint streaming step*

$$\varphi_j(t - h^2, \boldsymbol{r} - h^2 \boldsymbol{v}_{\boldsymbol{j}}) = \widetilde{\varphi}_j(t, \boldsymbol{r}) \tag{5.49}$$

which is alternatively referred to as *adjoint propagation step.*

As the collision step (2.19) in an LB algorithm the adjoint collision step (5.48) has a local character with respect to the position space. For this step the solution of the corresponding primal problem f_i $(i = 0, 1, ..., q-1)$ needs to be provided. It is important to note that for the computation of $\widetilde{\varphi}_j(t, \boldsymbol{r})$ $(j = 0, 1, ..., q-1)$ at a particular $t \in I_h$ and $r \in \Omega_h$ only all $f_i(t, r)$ $(i = 0, 1, ..., q-1)$ at the same t

and r are required. In order to obtain an efficient implementation, it is therefore recommendable to take advantage of this property. This can be realised by, for example, a local storage of the solution f_i $(i = 0, 1, ..., q - 1)$ with respect to φ_j $(j = 0, 1, ..., q-1)$. When it comes to realise a parallel approach, it is expected that the proposed treatment leads to a scalable implementation concerning the memory consumption.

While an adjoint streaming step (5.49) at a certain $(t, \boldsymbol{r}) \in I_h \times \Omega_h$ is performed, φ_j is manipulated only at direct neighbouring nodes. This also holds true for a streaming step (2.20) of an LB algorithm. However, the propagation takes place reversely.

Due to the structure of (5.48) and (5.49) combined with its mentioned locality properties an ALB algorithm can be implemented similarly to an LB algorithm akin Algorithm 1. Furthermore, because of the same reasons a data structure design as discussed in Subsection 2.2.1 is well-suited for an efficient implementation of ALB algorithms as well. With it, also the hybrid parallelisation strategy proposed and discussed in Chapter 3 can be applied. Executing an ALB scheme, it is expected that qualitatively it performs as efficiently as an LB scheme. Thereunto, a comparison of their respective performance results for an example, obtained both sequentially and in parallel, is presented and discussed in Subsection 5.6.4.

5.6. Numerical Experiments: A Distributed Control Problem

This section is devoted to study a stationary distributed control problem. Thereby, the external applied force $\boldsymbol{F}$ is to be manipulated within the whole domain Ω. The problem is formulated on a macroscopic base exactly as the distributed control problem considered in the previous chapter in Section 4.4. As already mentioned there, the problem is related to the 3D stationary fluid flow problem discussed in Section 2.3 where for a given force $\boldsymbol{F}$ the velocity distribution $\boldsymbol{u}$ is to be determined. Now, the problem is reversed, i.e. $\boldsymbol{u}$ is given in advance and $\boldsymbol{F}$ is to be computed.

The main goal of this section is to illustrate the adjoint-based fluid flow control and optimisation approach presented within this chapter. Furthermore, the section aims to validate the realisation and test its performance.

In the following, at first the test case in formulated. Then, in Subsection 5.6.2 the iterative optimisation algorithm of choice is stated. Afterwards, details regarding the chosen discrete models for the primal, the adjoint and the optimality equation are presented. Since an ALBM is applied to solve the stationary problem numerically, issues concerning an appropriate stop criteria need to be addressed. This is also subject of Subsection 5.6.2. In Subsection 5.6.3 the numerical results are presented and analysed. Finally, in the last part of this section the measured performance results for the realisation are studied. In both the ultimate and the penultimate subsection the obtained results are compared to those obtained for the AD-based first-discretise-then-optimise approach which are presented in Subsection 4.4.2 and Subsection 4.4.3.

5.6.1. Test Case Formulation with Corresponding Optimality System. The considered test case is a stationary distributed control problem which is formulated on a macroscopic base whereby a family of BGK-Boltzmann equations, differing in the choice of $h \in \mathbb{R}_{>0}$, serve as governing equation. The problem reads exactly as the distributed control problem formulated in Section 4.4. Taking advantage of the notations and definitions introduced there, the here considered optimisation problem is formulated. The goal functional J_∞ is defined according to (4.9). For a given $h \in \mathbb{R}_{>0}$ the function which defines the side condition is specific according to (4.10) and referred to as $\boldsymbol{G}^{\boldsymbol{h}}_{\boldsymbol{\infty}}$. Further, B_∞ is the control extension operator which is the identity map and $\alpha_{reg} \in \mathbb{R}_{\geq 0}$ is a regularisation parameter.

In Section 5.2 an optimality system for an abstract instationary distributed control problem is derived. It is very similar to the here considered test case, except for the vanishing time derivative and the boundary condition, which is here formulated as a macroscopic Dirichlet condition instead of a bounce back condition.

The first issue poses no difficulties. The stationary fluid flow problem that constitutes the side conditions can be considered naturally as quasi-instationary problem which converges in time towards the solution of the stationary problem. This procedure is illustrated just for the underlying fluid flow problem in Section 2.3. Then, the disappearance of the time derivative can be ensured by introducing an additional condition which is nothing but a stop criteria for the discrete algorithm. This condition can be easily transfered to the adjoint problem where it, as well, sets up the condition that the time derivative, though by now of φ, vanishes.

The second difference concerning the optimality system for the considered case, namely the macroscopic boundary condition formulation, cannot put down directly to the system derived in Section 5.2. A reason is that the handling of macroscopically formulated boundary conditions in a mesoscopic framework is in general a difficult task. Furthermore, most of the approaches are formulated on a discrete base (cf. Subsection 2.1.4). Therefore, in the following an indirect treatment via the macroscopic target equation of the BGK-Boltzmann equation which is an incompressible Navier-Stokes equation and its adjoint equation is proposed. The procedure in founded in the assumption that the series of adjoint BGK-Boltzmann equations is related to the particular adjoint incopressible Navier-Stokes equation in a limiting process whereby h tends to zero. Under this premise, one gets that the first moments of a solution φ^h are related to the adjoint variable $\boldsymbol{w}$ of $\boldsymbol{u}$ by

$$\boldsymbol{u}_{\boldsymbol{\varphi}^h} \overset{h\to 0}{\longrightarrow} \boldsymbol{w} \qquad\qquad \text{in } I \times \Omega\ . \tag{5.50}$$

Consider the corresponding incompressible Navier-Stokes equation with Dirichlet boundary conditions according to the ones defined as part of the side condition in (4.10), then, the corresponding boundary condition for the adjoint incompressible Navier-Stokes equation is a Dirichlet-zero condition, i.e. $\boldsymbol{w}|_{\partial\Omega} = \boldsymbol{0}$ in I (cf. Fursikov et al. [**43**]). This, in turn, motivates the appropriateness of the bounce back condition for the adjoint BGK-Boltzmann equations since it leads to

$$\int_{\mathbb{R}^d} \boldsymbol{v}\varphi^h(v)\ dv = \boldsymbol{0} \qquad\qquad \text{on } I \times \partial\Omega$$

for all h which ensures with (5.50) $\boldsymbol{w}|_{\partial\Omega} = \boldsymbol{0}$ in I. To be mentioned is that the brought up assumption (5.50) still remains to be verified analytically. However, the numerical results discussed later in this section are found to be consistent with that assumption. This encourages further investigations in this direction.

5.6.2. Numerical Realisation: Discretisation Issues and Optimisation Method of Choice. The previously formulated distributed control problem is solved numerically in accordance to the adjoint-based strategy presented in Section 5.1. The approach relies on the application of an iterative gradient-based optimisation algorithm as stated in (3). The method applied to solve the test case is a BFGS scheme combined with the Wolfe-Powell rule to determine the step lengths. The initial control is set to $\boldsymbol{\alpha}^{\boldsymbol{0}} := \boldsymbol{0}$. Then, in every optimisation step

$k = 1, 2, ...$ the primal and the adjoint equation is to be solved for a particular $\boldsymbol{\alpha^k}$. With the solution $f(\boldsymbol{\alpha^k})$ of the primal problem $J_\infty(f(\boldsymbol{\alpha^k}), \boldsymbol{\alpha^k})$ can be computed. If one additionally knows the solution of the corresponding adjoint equation φ, the gradient $\frac{d}{d\boldsymbol{\alpha}} J_\infty(f(\boldsymbol{\alpha^k}), \boldsymbol{\alpha^k})$ can be computed as stated in (5.6). Finally, with the optimisation algorithm one obtains the descent direction $\boldsymbol{d^k}$ and the step length δ^k.

For each $\boldsymbol{\alpha^k}$ ($k = 1, 2, ...$) the primal problem is given in terms of the side condition $\boldsymbol{G^h_\infty}(f(\boldsymbol{\alpha^k}), \boldsymbol{\alpha^k}) = \boldsymbol{0}$ for a particular $h \in \mathbb{R}_{>0}$. Except for the force term, each of these side conditions reads just as the governing equation of the fluid flow problem formulated in Subsection 2.3.1. For this problem a $D3Q19$ LBM is applied to solve it numerically which leads to the discrete space $I_h \times \Omega_h \times Q$. The same strategy is applied to solve the primal problems. In Subsection 2.3.2 further details concerning the applied discretisation strategy are stated. Thereunder, the applied methods to realise the boundary condition, the initial condition and also the stop criteria are specified.

The adjoint problem which is to be solved in every optimisation step $k = 1, 2, ...$ reads as (5.28). In order to solve the system numerically a $D3Q19$ ALBM as introduced in Section 5.4 is applied. The force term is treated separately as stated in Subsection 5.4.4. The obtained discrete space is the same as the one obtained for the discrete primal problem, namely $I_h \times \Omega_h \times Q$. In the primal problem the average density in Ω_h is kept close to a value of 1 by subtracting a constant offset from the distribution function f_i on every grid point in every time step (cf. Subsection 2.3.2). This technique is carried over to the adjoint algorithm. One obtains that the average of ρ_{φ_j} in Ω_h is to be computed and that an offset is to be subtracted from φ_j in $I_h \times \Omega_h \times Q$. With this procedure, it is ensured that the average of ρ_{φ_j} in Ω_h is kept to be close to zero. To check the convergence towards a steady state the change of the obtained solutions is monitored in an L^2-norm in every time step t. The simulation is stopped if the stop condition

$$e_h(t) := \|\varphi_j(t) - \varphi_j(t - h^2)\|_{L^2(\Omega_h \times Q)} < \epsilon \tag{5.51}$$

is fulfilled for a $t \in I_h$ whereby $\epsilon := 10^{-15}$.

Then, the control space $U_\infty = L^2(\Omega)^3$ is to be replaced by a finite dimensional space. As ansatz-space $U_n := L^2(\Omega_h)^3$ is chosen. It can be identified by $U_n \cong \mathbb{R}^n$ which is of dimension

$$n := 3\left(\frac{1}{h} - 1\right)^3 . \tag{5.52}$$

The discrete control extension operator is defined for $\boldsymbol{r} \in \Omega_h$ and $i = 1, 2, 3$ according to

$$\left(B_{h,n}(\alpha_n)\right)_i(\boldsymbol{r}) := \alpha_{3l(\boldsymbol{r})+i}$$

whereby $l(\boldsymbol{r}) \in \left\{1, 2, ..., \left(\frac{1}{h} - 1\right)^3\right\}$ is the index belonging uniquely to a particular $\boldsymbol{r} \in \Omega_h$. With it, both the primal and the adjoint problem can be solved numerically for a given $\boldsymbol{\alpha_n} \in U_n$.

The goal functional J_∞ is approximated applying the midpoint rule which leads to an expression $J_{h,n}$ which reads exactly as (4.11). Here, $J_{h,n}$ is defined for the considered $h \in \mathbb{R}_{>0}$ and for its by (5.52) coupled n. As for the test cases considered in the previous chapter the regularisation parameter is set to $\alpha_{reg} = 0$. Consequently, after one has solved the primal problem numerically, $J_{h,n}$ can be evaluated.

Finally, it is to be specified how (5.6) can be computed numerically for given $\boldsymbol{\alpha_n^k}, f_i(\boldsymbol{\alpha_n^k})$ and $\varphi_j(\boldsymbol{\alpha_n^k})$ $(i,j = 0,1,...,q-1)$. In the here considered cases (5.6) can be obtained just as the optimality condition is derived in Subsection 5.2.3 according to the transformations stated in (5.26). Suppose f_h is the solution of the primal problem and φ^h that of the adjoint problem which belongs to a dedicated $\alpha^k \in U_\infty$. Then, one obtains

$$\frac{d}{d\boldsymbol{\alpha}} J_\infty(f^h(\boldsymbol{\alpha^k}), \boldsymbol{\alpha^k}) = \alpha_{reg}\boldsymbol{\alpha^k} + B' \int_{\mathbb{R}^d} \frac{\varphi^h(\boldsymbol{\alpha^k})}{m} \boldsymbol{\nabla_v} f^h(\boldsymbol{\alpha^k})\, d\boldsymbol{v} \tag{5.53}$$

for $\boldsymbol{\alpha^k} \in U_\infty$ in $I \times \Omega$. This term needs to be evaluated (5.53) numerically based on the discretisation employed before. As it is common praxis for the BGK-Boltzmann equation the term $\boldsymbol{\nabla_v} f^h(\boldsymbol{\alpha^k})$ is not treated directly. In Subsection 5.4.4 one of many available approaches is addressed shortly. It relies on replacing the term $\boldsymbol{\nabla_v} f^h(\boldsymbol{\alpha^k})$ by $\boldsymbol{\nabla_v} M^{eq}_{f^h(\boldsymbol{\alpha^k})}$. Then considering the moments of $\boldsymbol{\nabla_v} f^h(\boldsymbol{\alpha^k})$, a Gauss-Hermite quadrature leads to q supporting points v_i and weights w_i $(i = 0,1,...,q-1)$ which are identical to those belonging to the $D2Q9$ and $D3Q27$ model. This *modus operandi* is now applied to evaluate the integral term of 5.53. The replacement of $f^h(\boldsymbol{\alpha^k})$ by $M^{eq}_{f^h(\boldsymbol{\alpha^k})}$ leads to

$$\begin{aligned}\frac{d}{d\boldsymbol{\alpha}} J_\infty(f^h(\boldsymbol{\alpha^k}), \boldsymbol{\alpha^k}) &\approx B' \int_{\mathbb{R}^d} \frac{\varphi^h(\boldsymbol{\alpha^k})}{m} \boldsymbol{\nabla_v} M^{eq}_{f^h(\boldsymbol{\alpha^k})}\, d\boldsymbol{v} \\ &= -B' \int_{\mathbb{R}^d} \underbrace{\frac{\varphi^h(\boldsymbol{\alpha^k})(\boldsymbol{v})}{m} 3h^2(\boldsymbol{v} - \boldsymbol{u_{f^h(\alpha^k)}})}_{:=b(\boldsymbol{v})} M^{eq}_{f^h(\boldsymbol{\alpha^k})}\, d\boldsymbol{v}\end{aligned}$$

for $\boldsymbol{\alpha^k} \in U_\infty$ in $I \times \Omega$, whereby the regularisation parameter is set to be $\alpha_{reg} = 0$. Then, Gauss-Hermite quadrature can be applied analogue to (5.34) by setting $a(\cdot, \boldsymbol{v}) := b(\boldsymbol{v}) n_{f^h}$ $(v \in \mathbb{R})$ and assuming that the conditions imposed on a in Subsection 5.4.2 also hold for bn_{f^h}. This yields

$$\begin{aligned}&\frac{d}{d\boldsymbol{\alpha}} J_\infty(f^h(\boldsymbol{\alpha^k}), \boldsymbol{\alpha^k}) \\ &\quad\approx -B' \sum_{i=0}^{q-1} \frac{w_i}{w} \frac{\varphi^h(\boldsymbol{v_i})}{m} 3h^2(\boldsymbol{v_i} - \boldsymbol{u_{f^h}}) \exp\left(3h^2 \boldsymbol{v_i} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2} h^2 \boldsymbol{u_{f^h}}^2\right) \\ &\quad= -B' \sum_{i=0}^{q-1} \frac{w_i}{w} \frac{\varphi_j^h}{m} 3h(\widetilde{\boldsymbol{v}}_{\boldsymbol{i}} - h\boldsymbol{u_{f^h}}) n_{f^h} \exp\left(3h \widetilde{\boldsymbol{v}}_{\boldsymbol{i}} \cdot \boldsymbol{u_{f^h}} - \frac{3}{2} h^2 \boldsymbol{u_{f^h}}^2\right)\end{aligned}$$

for $\boldsymbol{\alpha^k} \in U_\infty$ in $I \times \Omega$, whereby v_i are supporting points and w_i are weights for $i = 0,1,...,q-1$ which are identical to those belonging to the $D2Q9$ and $D3Q27$ model. Expanding the exponential function around 0 and neglecting all terms which are in $\mathcal{O}(h^3)$ yields

$$\begin{aligned}\frac{d}{d\boldsymbol{\alpha}} J_\infty(f^h(\boldsymbol{\alpha^k}), \boldsymbol{\alpha^k}) &\approx -B' \sum_{i=0}^{q-1} \frac{w_i}{w} \frac{\varphi_i^h}{m} 3h \widetilde{\boldsymbol{v}}_{\boldsymbol{i}} n_{f^h} && \text{in } I \times \Omega \\ &\approx -B' \sum_{i=0}^{q-1} \frac{w_i}{w} \frac{\varphi_i^h}{m} 3h \widetilde{\boldsymbol{v}}_{\boldsymbol{i}} n_{f_i^h} && \text{in } I_h \times \Omega_h\end{aligned}$$

for $\boldsymbol{\alpha^k} \in U_\infty$ whereat in the last transformation n_{f^h} is replaced consistently by $n_{f_i^h}$ (cf. Subsection 2.1.2). Finally, the gradient is evaluated just for $\boldsymbol{\alpha_n^k} \in U_n$.

Therefore, B is replaced by $B_{h,n}$. Denoting the discretised evaluated gradient with $\boldsymbol{gJ_{h,n}}(\boldsymbol{\alpha_n^k})$ one obtains for $\boldsymbol{\alpha_n^k} \in U_n$

$$\frac{d}{d\boldsymbol{\alpha}} J_\infty(f^h(\boldsymbol{\alpha_n^k}), \boldsymbol{\alpha_n^k}) \approx \boldsymbol{gJ_{h,n}}(\boldsymbol{\alpha_n^k}) := -B'_{h,n} \sum_{i=0}^{q-1} \frac{w_i}{w} \frac{\varphi_i^h}{m} 3h \widetilde{\boldsymbol{v}}_{\boldsymbol{i}} n_{f_i^h} \quad \text{in } I_h \times \Omega_h \ .$$

With it, all terms can be computed on a discrete base so that the test problem can be solved numerically.

5.6.3. Presentation and Discussion of the Numerical Results. In this subsection the numerical results obtained for the considered test case are presented and analysed, whereby the discretisation parameter h is varied. The main aim is to validate the presented approach and its realisation numerically. For that purpose the obtained results are compared to those obtained for the first-discretise-then-optimise approach presented in the previous chapter. Another goal of the presentation is to provide a basis for comparisons with results obtained for the same test case by applying other methods. In particular, it would be interesting to compare the results to those which are obtained by modelling the underlying fluid flow problem macroscopically, i.e. with the corresponding incompressible Navier-Stokes equation as governing equation, and solving it by e.g. a FEM.

At first, for the dual problems, which are solved by a $D3Q19$ ALB scheme, the convergence behaviour towards a quasi-stationary solution is studied. Exemplarily, the system which occurs in the first optimisation step is considered. A comparison with the convergence behaviour obtained for the following optimisation steps reveals no significant differences for the considered test case. Aiming to analyse the characteristics, the change of the obtained solutions $e_h(t)$, which is defined according to (5.51), is monitored in every time step $t \in I_h$. The results are depicted for different discretisation parameters, namely $h = 1/10, 1/20, 1/40, 1/80, 1/160$, in Figure 5.1.

The obtained results clearly show an exponential decrease of e_h to a certain level around 10^{-16} for all considered discretisation parameters h. This limitation can be explained with the accuracy of the employed data type which is of double precision. This implies that an relative precision error of more than 10^{-16} cannot be expected. The differences in the reached level might be due to the accumulation of rounding errors occurring then computing the $L^2(\Omega_h)$-norm as a sum over $q\left(\frac{1}{h} - 1\right)^3$ terms.

Further, it is observed that $e_{1/40}$, $e_{1/80}$ and $e_{1/160}$ show approximately the same convergence characteristics. This result is not surprising. It indicates an underlying physical meaning of the set of problems which might be founded in the already mentioned relation between the hierarchy of adjoint BGK-Boltzmann equations to a particular adjoint incompressible Navier-Stokes equation.

Moreover, from this observed convergence characteristics one can deduce an experimental order of convergence for the costs to solve a linear system with $N := q\left(\frac{1}{h} + 1\right)^3$ unknowns, which is given by an ALB equation, by means of an ALB scheme. It is to be expected that the computational costs to perform one time step depend linearly on N as the adjoint collision step (5.48) and the adjoint propagation step (5.49) reveal. This theoretical result is confirmed experimentally as shown in Table 5.1. As the graph in Table 5.1 illustrates, a certain accuracy $e_h(t^*)$ is reached independently of h at about the same physical time t^*. However, the number of time steps needed depends linearly on h^2 so that e.g. for the test case with $h = 1/80$ four times more time steps are to be performed to reach t^* as for the one with $h = 1/40$. With it, one obtains an experimental order of convergence

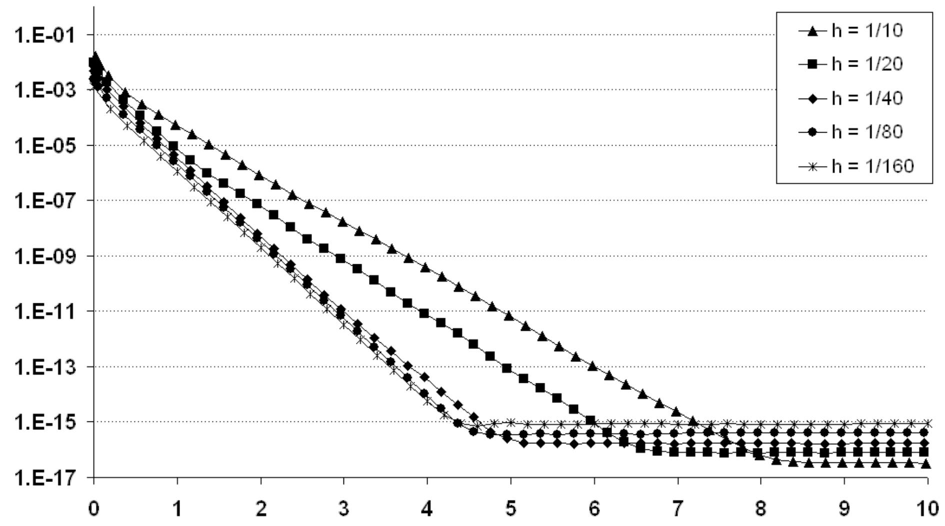


FIGURE 5.1. In order to check the convergence behaviour, the development towards a stationary solution is illustrated by the plot. Therefore, $e_h(t) = \|\varphi_j(t) - \varphi_j(t - h^2)\|_{L^2(\Omega_h \times Q)}$ is depicted as a function of time for the first optimisation step and different discretisation parameters h.

which is a function of $\mathcal{O}(N^{5/3})$.

In Figure 5.2 the obtained numerical results for the goal functionals $J_{h,n}$ of the considered test case are presented for $h = 1/10, 1/20, 1/40, 1/80$. To compare the results obtained for the different refinements levels, the evolution of the normalised goal functionals $J_{h,n}(f_i(\boldsymbol{\alpha_n}^k), \boldsymbol{\alpha_n}^k)/J_{h,n}(f_i(\boldsymbol{\alpha_n}^1), \boldsymbol{\alpha_n}^1)$ is shown as a function of performed optimisation steps $k = 1, 2, \dots$. The results for the corresponding approximated derivatives are depicted in Figure 5.3. Here, for better comparability the normalised normed gradients $\|\boldsymbol{gJ_{h,n}}(\boldsymbol{\alpha_n^k})\|_{L^2(U_n)}/\|\boldsymbol{gJ_{h,n}}(\boldsymbol{\alpha_n^1}))\|_{L^2(U_n)}$ are stated for all studied discretisation parameters h.

For all considered h it is observed that the value of the goal functionals decreases as the number of optimisation steps increases. However, they are not found to tend to zero. In the case where $h = 10$, after 40 optimisation steps $J_{h,n}(f_i(\boldsymbol{\alpha_n}^k), \boldsymbol{\alpha_n}^k)/J_{h,n}(f_i(\boldsymbol{\alpha_n}^1), \boldsymbol{\alpha_n}^1)$ reaches a value of approximately $4.7 \cdot 10^{-3}$ which decreases very slowly to $3.9 \cdot 10^{-3}$ at $k = 50$; then, no further improvements are obtained. Yet, it is measured that for all $k > 6$ the values obtained for the normalised goal functional become smaller the smaller h is chosen. Further, in all investigated cases a smaller relative error for the velocity field can be reached than the one obtained as relative discretisation and model error for the underlying fluid flow problem whereby $\boldsymbol{F} = \boldsymbol{F}^*$ (cf. Section 2.3 and Figure 2.5). For $h = 10$ for example, the relative error after 30 optimisation steps has been performed is found to be $\|\boldsymbol{u_{f_i(\alpha_n{}^30)}} - \boldsymbol{u}^*\|_{L^2(\Omega_{10})}/\|\boldsymbol{u}^*\|_{L^2(\Omega_{10})} \approx 0.024$ while the relative discretisation and model error is found to be $\|\boldsymbol{u_{f_i}} - \boldsymbol{u}^*\|_{L^2(\Omega_{10})}/\|\boldsymbol{u}^*\|_{L^2(\Omega_{10})} \approx 0.06$. The development of the gradients shows a similar behaviour. Depending on h after a certain number of optimisation steps no further improvement can be observed. For $h = 10$, again, this is the case from about $k = 40$ on. In the case where $h = 20$ after $k = 70$ optimisation steps and for $h = 40$ after $k = 90$ the value of the gradient does not decrease monotonously anymore. Summing up all these observations, it seems

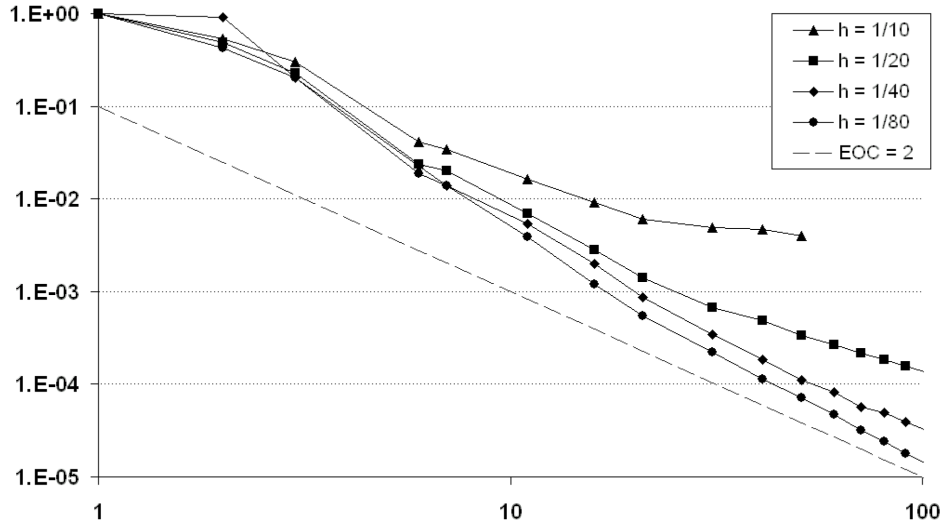


FIGURE 5.2. The graph of the normalised goal functional $J_{h,n}(f_i(\boldsymbol{\alpha_n}^k), \boldsymbol{\alpha_n}^k)/J_{h,n}(f_i(\boldsymbol{\alpha_n}^1), \boldsymbol{\alpha_n}^1)$ understood as a function of performed optimisation steps k is displayed for different discretisation parameters h.

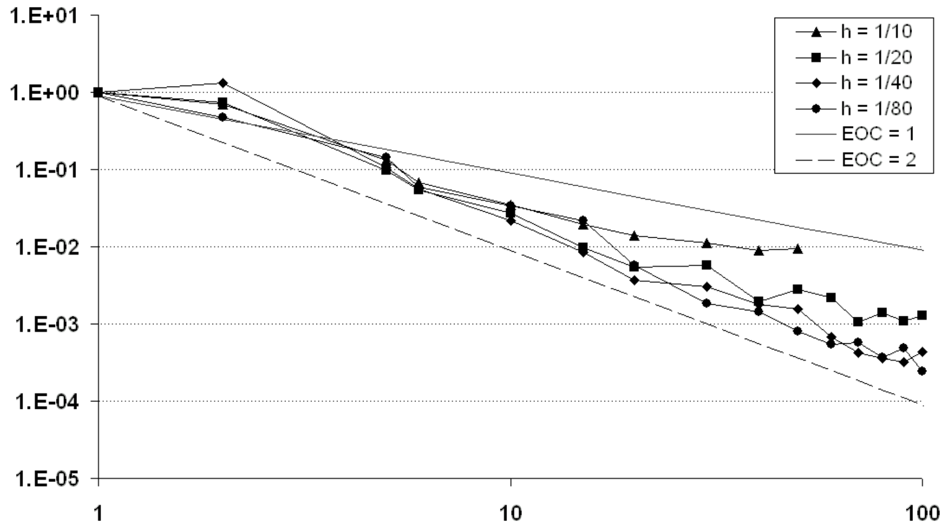


FIGURE 5.3. The graph of the normalised normed gradient $\|\boldsymbol{gJ_{h,n}}(\boldsymbol{\alpha_n^k})\|_{L^2(U_n)}/\|\boldsymbol{gJ_{h,n}}(\boldsymbol{\alpha_n^1}))\|_{L^2(U_n)}$ understood as a function of performed optimisation steps k is displayed for different discretisation parameters h.

plausible that the discretisation errors occurring, e.g. when solving the primal and the dual problem, become visible in the mentioned behaviour.

It is interesting to note that the results are found in fairly good agreement to those obtained applying the AD-based first-discretise-then-optimise strategy proposed in the previous chapter, which are discussed in detail in Subsection 4.4.3.

The conformity is observed both quantitatively and qualitatively. Comparing the rate of decrease of the goal functionals, both approaches lead to a quadratically decreasing goal functional ($EOC = 2$) for the smallest h and greatest n which is in each case considered. Also the order of decline in the gradients is found to be similar for both experiments. Quantitatively, the magnitude of improvement of both the goal functional and its normed derivative are the same when comparing the two approaches applied for similar discretisation parameters h and similar dimensions n of the control variable. One observed difference is the range of fluctuation found for the gradients. The results for the first-discretise-then-optimise approach show a much higher fluctuation than the other results. However, it is measured that the fluctuation is the smaller the higher the chosen dimensions n is.

Summing up, all discussed results are found plausible so that the proposed approach and its implementation can be seen as validated by the numerical experiments. This encourages a further deeper study of the approach both theoretically and practically.

5.6.4. Presentation and Discussion of the Performance Results. This last part of Chapter 5 is dedicated to present and discuss the performance results obtained for the considered test case, which is solved numerically by the adjoint-based approach as presented within this chapter. Thereby, solving the primal and adjoint problem in every optimisation step is the most computing time demanding part of the algorithm. In the considered framework, the primal problem is nothing but a fluid flow problem those governing equation is a BGK-Boltzmann equation. The performance of solving such problems numerically by applying an LBM is discussed in detail in Chaper 3. Therefore, in the following, exclusively the performance results concerning solving an adjoint BGK-Boltzmann equation by an ALBM are studied. In addition, the results of both the sequentially and the in parallel executed code are discussed and compared to those which are obtained by applying the first-optimise-then-discretise strategy based on AD and which are presented in Subsection 4.4.4.

The here considered results are obtained on the $HPXC4000$ supercomputer (cf. Section 3.3). Thereby, as for the tests discussed in Section 3.3, only one core per node is employed. The total number of cores involved is denoted by p. The code is executed three times to avoid random hardware-caused outliers. Then, in the following, always the averages of the three results are stated. All source code is compiled with the Intel compiler (version 10.1.022) using the optimisation option $-O3$ and linking Hewlett Packard's MPI library (version 2.3.1).

The computing times needed to perform 100 time steps of an $D3Q19$ ALB algorithm are measured. Thereby, in each test case, time step 100 to 199 of the first optimisation step $k = 1$ is taken as a basis. The results are captured in the variable $t_p(h, n)$ whereby h and, respectively, n characterise the test case with the convention that $n = 0$ stands for the measured time needed to perform 100 times steps of a $D3Q19$ LB algorithm.

At first, the results which are obtained by executing the source code sequentially are presented. Varying the discretisation parameter h, the test results are condensed in Table 5.1. The measured quantities are found to confirm the theoretical expectation that the computing times $t_1(h, n)$ grow linearly with the number of unknowns $N_h = q\,(1/h + 1)^3$. Thereunto, the ratios N_h/N_{2h} and $t_1(h, n)/t_1(2h, n)$ provide this information. Since the computational costs for a primal problem also

h	1/10	1/20	1/40	1/80
N_h	25, 289	175, 959	1, 309, 499	10, 097, 379
N_h/N_{2h}		6.96	7.44	7.71
$t_1(h,0)$	0.19 s	1.47 s	9.62 s	69.80 s
$t_1(h,0)/t_1(2h,0)$		7.74	6.54	7.26
$t_1(h,n)$	0.30 s	2.83 s	23.53 s	181.03 s
$t_1(h,n)/t_1(2h,n)$		9.43	8.31	7.69
$t_1(h,n)/t_1(h,0)$	1.58	1.69	2.45	2.59

Table 5.1. This table displays the computing times $t_1(h,n)$ for 100 times steps solving the primal ($n=0$) and the adjoint problem ($n = 3\left(\frac{1}{h}-1\right)^3$) of the first optimisation step $k=1$ on lattices with different discretisation parameters h. The measured times are obtained by executing the code on one core of one node on the *HP XC4000*. The numbers of unknowns N_h are given also as fractions relating the measured computing times of different discretisation parameters h.

increase linearly with N_h, the ratio of the costs for a time step of the primal problem compared to one of the adjoint problem with the same number of unknowns is expected to be constant. Experimentally, this ratio, i.e. $t_1(h,n)/t_1(h,0)$, is measured to approximate a value of about 2.5 if h gets small. Here, it is to be mentioned that the implementation of the ALB algorithm can still be optimised. In doing so, it is expected that a ratio of about $t_1(h,n)/t_1(h,0) \approx 1$ can be reached. Comparing this results with those obtained for the AD-based approach, the found results differ substantially. There, $t_1(h,n)$ and $t_1(h,0)$ are found to be related by (4.12) with the experimentally gained constant $C_{grad}(1,h,n) \approx 2.1$. This results in a ration of $t_1(h,n)/t_1(h,0) \approx 1+2.1\cdot n$ so that the AD-based approach is more expensive than the adjoint-based approach in any of the considered cases.

Executing the code in parallel, the obtained results are found in accordance to those presented in Section 3.4. This is not surprising, since, as remarked in Section 5.5, the same parallelisation strategy (cf. Chaper 3) is applied to obtain a parallel ALB scheme. In Figure 5.4 one finds the experimentally gained efficiencies E_{ff} plotted for different discretisation parameters h. Thereby, the quantity E_{ff} is defined according to (3.1). The graph clearly shows that the smaller h is chosen the better the corresponding measured efficiencies become with a few exceptions for small values of p. Then employing $p = 256$ cores the best efficiency of $E_{ff} \approx 0.44$ is obtained for $h = 1/80$. An efficiency of $E_{ff} \approx 0.57$ is obtained for the test case considered in Section 3.4. Indeed, both problems have the same underlying geometry $\Omega = [0,1]^3$ in common and are both solved numerically by applying a $D3Q19$ LBM. Yet, the discretisation parameter of the second problem is chosen much smaller, namely $h = 1/200$. Taken this difference in h, together with the results for other choices of h, shown there in Table 3.5, into account, extrapolation yields the reasoning that the here measured results match, also quantitatively, fairly well.

Summing up all observations, two conclusions can be drawn. The first one concerns the computational costs for solving by an ALB scheme. The costs for 100 times steps are found to grow proportionally to the number of unknowns N

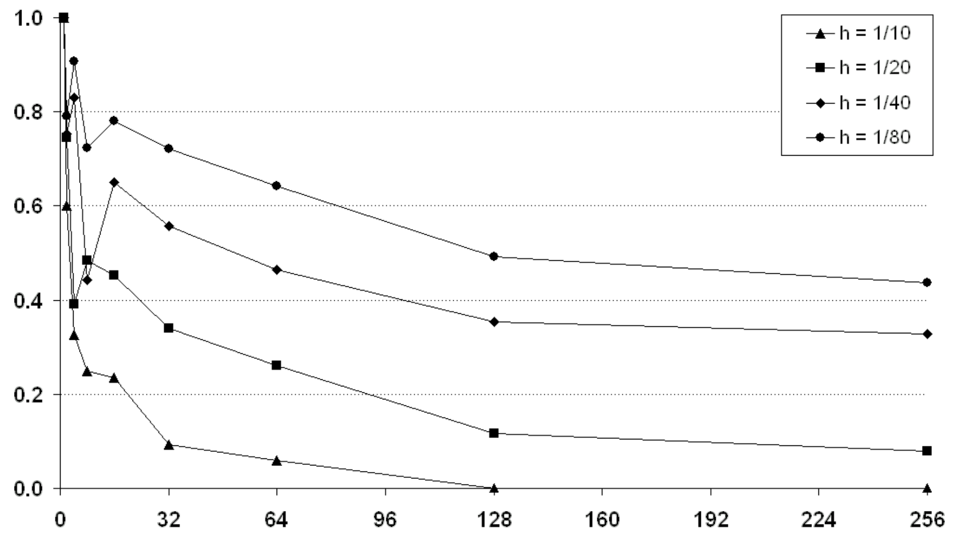


FIGURE 5.4. Efficiency as a function of the number of cores employed for 100 time steps and various grid sizes obtained on the *HP XC4000* for the MPI-based approach.

which is in contrast to the AD-based approach investigated in the previous chapter. For them, the costs are found to grow proportionally to nN, i.e. the costs are about as many times higher as the dimension n of the control space U_n. Secondly, the parallelisation strategy proposed for LB schemes can also be applied to ALB schemes whereat, as expected due to its common structure, both quantitatively and qualitatively similar performance results are obtained. This second observation also holds for the AD-based first-optimise-then-discretise approach. Nevertheless, although the adjoint-based approach is clearly superior in terms of performance, for many relevant applications in practice the easier implementation of an AD-based strategy should not be ignored.

CHAPTER 6

Numerical Simulation of Human Respiratory Flows

As already remarked in the introduction, the numerical simulation of the full human respiratory system corresponds to one of the *grand challenges* in scientific computing nowadays. The field of applications arising from this capability is tremendous. For example, it encompasses the *a priori* analysis of possible implications on the respiratory tract due to surgery or the environmental impact on lungs. Thereby, especially the highly complex multi-physics phenomenology which involves multi-scale features and the underlying complex geometry are main difficulties. In this chapter some aspects of this problems are challenged. The presented study is mainly dedicated to present the numerical results obtained for simulations of respirations in an individual human nose and upper part of the human lungs. Whereto, many methods and techniques introduced in the former chapters are applied. For instance, the underlying flow problems are formulated by means of the mesoscopic model introduced in Chapter 1 which governing equations are BGK-Boltzmann equations. The problems are solved numerically applying LBM as they are studied in Chapter 2. The source code is executed in parallel taking advantage of the hybrid parallelisation strategy which is in the focus of Chapter 3.

activity male adult	resting (sleeping)	sitting awake	light exercise	heavy exercise
tidal volume	630 ml	750 ml	1300 ml	1900 ml
respiration frequency	12 min^{-1}	12 min^{-1}	20 min^{-1}	26 min^{-1}
mean ventilation rate	126 ml/s	150 ml/s	433 ml/s	823 ml/s
max. ventilation rate	198 ml/s	236 ml/s	681 ml/s	1293 ml/s

TABLE 6.1. Reference values for respiratory parameters at different levels of physical activity for an adult man. The tidal volume and the respiration frequency values are taken from Valentin et al. [**134**]. There, more details are given concerning e.g. the definition of the level of activity and how the reference values are obtained. The mean ventilation rate is computed directly from the former two values. For the computation of the maximum ventilation rate it is assumed that the ventilation rate is a sine-shaped function of time.

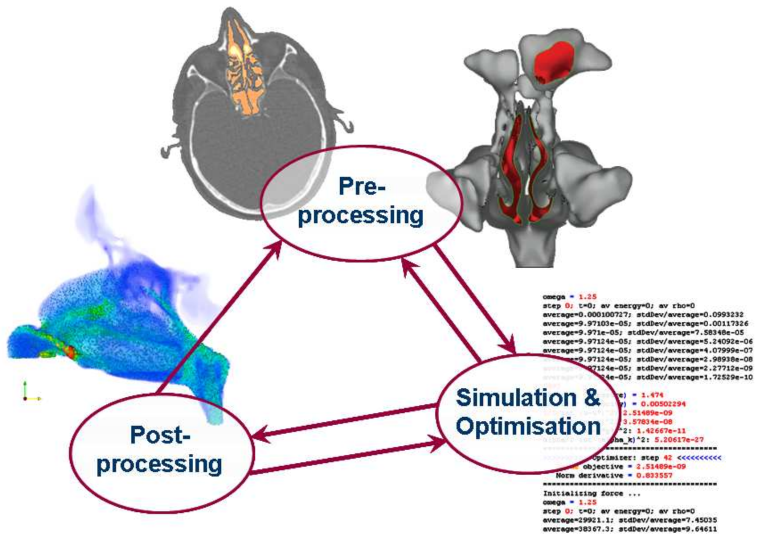
Recalling the aims stated in the introduction, the proposed approach is meant to enable other scientists to get a better fundamental understanding of the respiratory functionalities but also to accelerate the progress towards patient-specific treatment strategies. Hence, in the following special care is taken to obtain reliable results for individual patients. Firstly, original patient data together with realistic everyday situations (cf. Table 6.1) build the basis for all considered test cases. The underlying geometry data for the numerical simulations are obtained by computer tomography (CT) scans of a patient with a *peripheral obstructive ventilation disorder* diagnosed by Giotakis [**16**]. The male patient is 46 years old, 1.98 m tall and his weight is 105 kg. Secondly, a convergence study is carried out. Thirdly, the obtained results are compared with those obtained by others both experimentally and numerically. Fourthly, the obtained results are compared to individual measurements corresponding to the underlying geometries. Furthermore, a strategy for a widely automated preprocessing dedicated for LB simulations in complex geometries is introduced.

Flow characteristics in the human airways have been studied before both experimentally and numerically. In a general manner, Kleinstreuer and Zhang provide in [**81**] a detailed overview about publications considering state-of-the-art models, experimental observations and computer simulations for all parts of the respiratory system. In this work, short reviews of related published work are given separately with respect to that part which is actually considered, namely for the human nose in Section 6.2 and for the human lungs in Section 6.3.

In detail, this chapter is organised as follows. At first, in Section 6.1 a strategy for a widely automated complete preprocessing, starting with CT data and ending with start of the actual numerical simulation, is presented. The approach is dedicated to prepare the discrete computational domain of complex geometries for LB simulations. The approach is illustrated by considering two examples, namely the lungs and nose which build the basis for the latter simulation. However, it is stressed that the approach is by no means restricted to this application. In Section 6.2 the flow of air in the human nose is considered by means of two test configurations which are simulated. The first one aims to establish numerical evidence for pseudo steady states and, furthermore, to validate the results by means of a comparison with numerical results obtained by others and experimentally obtained data for other similar geometries. The second test suite is formulated in a way which enables a comparison with measurements obtained for the actual considered patient. Finally in the last section of this chapter, a feasibility study is presented which enables applying a two-scale model describing the respiration in the complete human lungs. Thereto, an expiration at a fixed flow rate in the upper part of the considered human lungs is simulated and the results are analysed. In each of the last two sections a short overview about the anatomy and physiology of the human nose and, respectively, the human lungs is given. Both parts aim to establish the terminology and physiology which is of interest for the remainder of the respective sections.

6.1. Preprocessing for Complex Geometries: From CT Data to the Simulation Setup

Before a numerical simulation can be started, a representation of the discrete geometry together with the corresponding values for the initial and boundary conditions need to be provided. Thereby, the required data must meet precise requirements which depend on the considered numerical method. In the framework of LBM as they are introduced in Chapter 2, the geometry incorporates an inner fluid region Ω_h and a boundary Γ_h which must be defined as stated in Subsection 2.1.1. Yet, usually the raw data is not given in this particular format. Hence, it needs to be prepared. In the considered case, images of the respiratory tract obtained by *computer tomography* (CT) and certain measurements constitute the starting basis.



```
omega = 1.25
step 0; t=0; av energy=0; av rho=0
average=0.000100727; stdDev/average=0.0993232
average=9.97103e-05; stdDev/average=0.00117326
average=9.971e-05; stdDev/average=7.58348e-05
average=9.97124e-05; stdDev/average=5.24092e-06
average=9.97124e-05; stdDev/average=4.07999e-07
        =9.97124e-05; stdDev/average=2.98938e-08
          97124e-05; stdDev/average=2.27712e-09
            4e-05; stdDev/average=1.72529e-10

              e) = 1.474
              y) = 0.00502294
                 2.51489e-09
                 3.57834e-08
               ^2: 1.42667e-11
          pha_k)^2: 5.20617e-27
=================================
        Optimizer: step 42 <<<<<<<<<<
           objective = 2.51489e-09
   Norm derivative = 0.833557
=======================================
Initializing force ...
omega = 1.25
step 0; t=0; av energy=0; av rho=0
average=29921.1; stdDev/average=7.45035
average=38367.3; stdDev/average=9.64611
```

FIGURE 6.1. Circle of full numerical simulation exemplified for the simulation of the flow in a human nose. The importance of the preprocessing as part of a full numerical simulation becomes obvious. Consequently, automatisation of preprocessing steps becomes indispensable.

All steps that are to be taken from the before mentioned starting point to the actual start of a simulation are, in what follows, abstracted by the term *preprocessing*. The importance of the preprocessing greatly comes to the fore if one is faced to deal with complex geometries. In the cases considered here, the geometries are highly complex. For example, parts of the geometry of both, the human lungs and the inner nose, are of such small scales that they cannot be captured by the best imaging technique available nowadays. Therefore, adequate preprocessing techniques need to be developed and applied to enable realistic numerical simulations of physical phenomena. Furthermore, one wants to or has to prepare the data for a numerical simulation many times e.g. for patient-individual flows. In this context to be mentioned are adaptive techniques which couple preprocessing, simulation and optimisation techniques (cf. e.g. [**9**] and references therein). In this framework e.g. *a posteriori error estimation methods* [**3**] are employed together with grid refinement strategies to obtain more accurate results. All these examples clearly show that the preprocessing cannot be seen isolated. In fact, the preprocessing is rather a part of the *full numerical simulation* which comprises also the actual numerical simulation and optimisation as well as the postprocessing. The relations of the single parts of the full numerical simulation are depicted schematically in Figure 6.1.

As a consequence of the prominent role of the preprocessing in the framework of full numerical simulation the requirement of a high grade of automatisation of preprocessing steps becomes obvious. This need is strengthened by the fact that the complexity of the geometry makes handiwork to an exhausting and hence almost impossible task.

In the following, a concept for a complete preprocessing dedicated to simulate the flow in the human lungs and nose with LBM is presented. Thereby, emphasis is placed to obtain a preferably automatised approach. The overall strategy consists of three main steps which constitutes the following chain:

$$\text{CT data} \overset{\text{Subsec. 6.1.1}}{\longrightarrow} \text{Surface mesh} \overset{\text{Subsec. 6.1.2}}{\longrightarrow} \text{Voxel mesh} \overset{\text{Subsec. 6.1.3}}{\longrightarrow} \text{Simulation} .$$

A separate subsection is devoted to describe each of these steps. There, the concept is explained by means of considering the preprocessing for LB simulations of the flow in the human nasal cavity and the human lungs. All preprocessing steps are illustrated by a series of images in the Figures 6.3 to 6.6, showing the segmented CT data, the surface mesh, the volume mesh and finally the obtained numerical results at six particular chosen slices of the nasal cavity.

Similar proprocessing approaches which are as well dedicated for complex geometries have been discussed before. For example, in [**74**] Inthavong et al. present a preprocessing concept at which also CT data of a human nose serves as starting point. However, the approach aims to create an unstructured tetrahedral mesh and hence cannot be applied for LB simulations. Freitas et al. [**41**] extract a voxel mesh representation of a human lungs cast including the main part of the trachea down to the sixth generation of the bronchial tree. Thereby, as well CT data build the starting point and a surface mesh is created in an intermediate step. Yet, there are no details concerning an automatised preprocessing nor further deeper descriptions of the employed preprocessing steps given.

6.1.1. From CT Data to Surface Mesh. Standard image processing methods which are often applied in the area of medicine are e.g. *computer tomography* (CT) or *magnetic resonance imaging* (MRI). A common standard file format to store the obtained images beside other data is the *digital imaging and communications in medicine* (DICOM) standard [**100**]. The considered cases are not exceptions. The data are obtained by a CT scan with a *SOMATOM Sensation 64* [**121**] and provided as DICOM files. In Figure 6.2 some images of the raw CT data are displayed. Different levels of grey scales reflect different material densities. In the presented images the original colours are negated, so that a dark colour represents a high material density while a light one indicates a low material density.

The mean resolution of the scans is according to [**121**] 0.4 mm. Even with this high resolution it is not possible to resolve all parts of the investigated human lungs and nose. Additionally, minor movements of the patient and electronic noise impede to obtain clear shapes of boundaries. In order to segment the volume of the human lungs and nose which is filled with air an *image segmentation technique* is to be applied. An overview of such approaches is given e.g. by Haralick et al. in [**60**]. Among that great variety of methods offered one finds *seeded* [**2**] and *dynamic region growing schemes* [**123**]. These enable robust and partly automated segmentation of a region which can be identified by its colour level given by thresholds. Similar methods are implemented for example in the software package *Mimics* [**106**] from *Materialise NV* which is employed for the considered cases. For the human nose in the Figures 6.3 and 6.5 the segmented CT data of chosen slices are depicted.

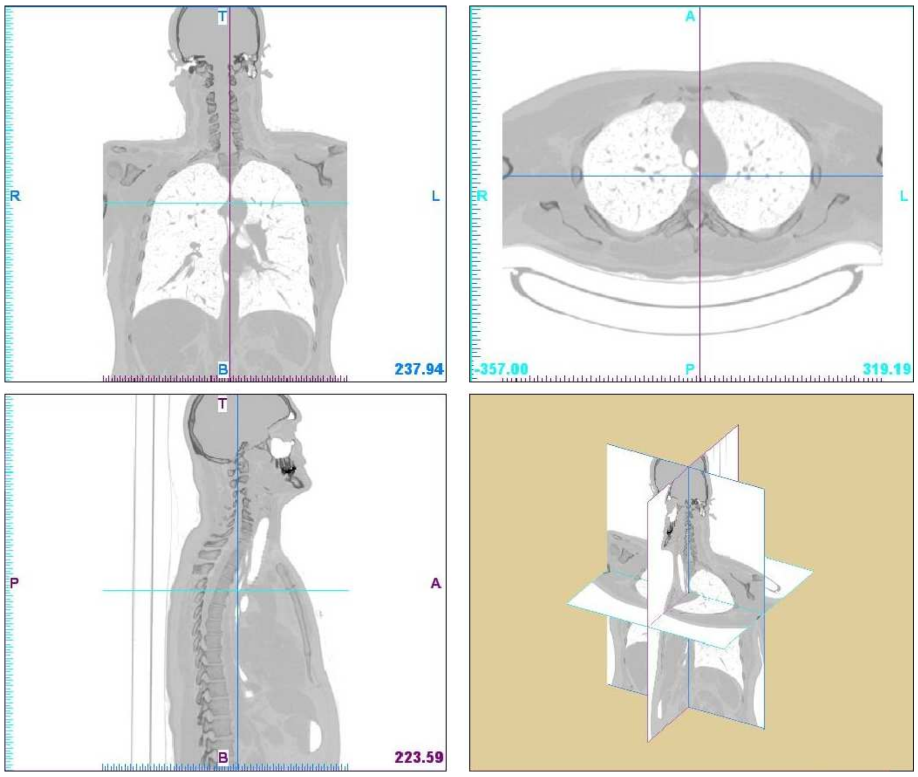


FIGURE 6.2. The CT raw data of the considered patient. For better visualisation the original colours are negated, so that now a dark colour represents a high material density while a light one indicates a low material density. The letters stand for **A**nterior, **P**osterior, **T**op, **B**utton, **L**eft and **R**ight and the numbers for the position in an underlying Cartesian coordinate system.

After the two regions of interest (inner human lungs and nose) have been segmented from the CT images the corresponding boundaries are extracted. The surface is at first smoothened and later approximated by a certain number of triangles. With this procedure a finer resolution of the volume mesh than that of the original CT data can be obtained. As data format for the surface mesh serves the quasi file format standard *STL* which is native to the stereolithography computer-aided design (CAD) software created by 3D Systems [**1**]. These steps are as well performed with help of the software package *Mimics* [**106**]. In the Figures 6.3 and 6.5 the smoothened surface of the considered inner nose geometry is visualised. The pictures show the geometry in the same chosen planes as before for the segmented CT data. The volume of this model is found to have a capacity of approximately 111.2 ml. 258, 186 triangles represent the surface which covers an area of about 469 cm^2. Figure 6.14 shows the whole segmented CT data area of the human lungs and the surface of that part of the human lungs which could be clearly extracted from the segmented CT data. The obtained surface model consists of 17, 940 triangles which cover an area of about 176 cm^2 and give room for a capacity of approximately 51.86 ml.

6.1.2. From Surface Mesh to Voxel Mesh. LBM require as discrete position space a uniform mesh $\Omega_h \cup \Gamma_h$ with discretisation parameter $h \in \mathbb{R}_{>0}$ which is called the lattice. To handle different kinds of boundary condition Γ_h is split

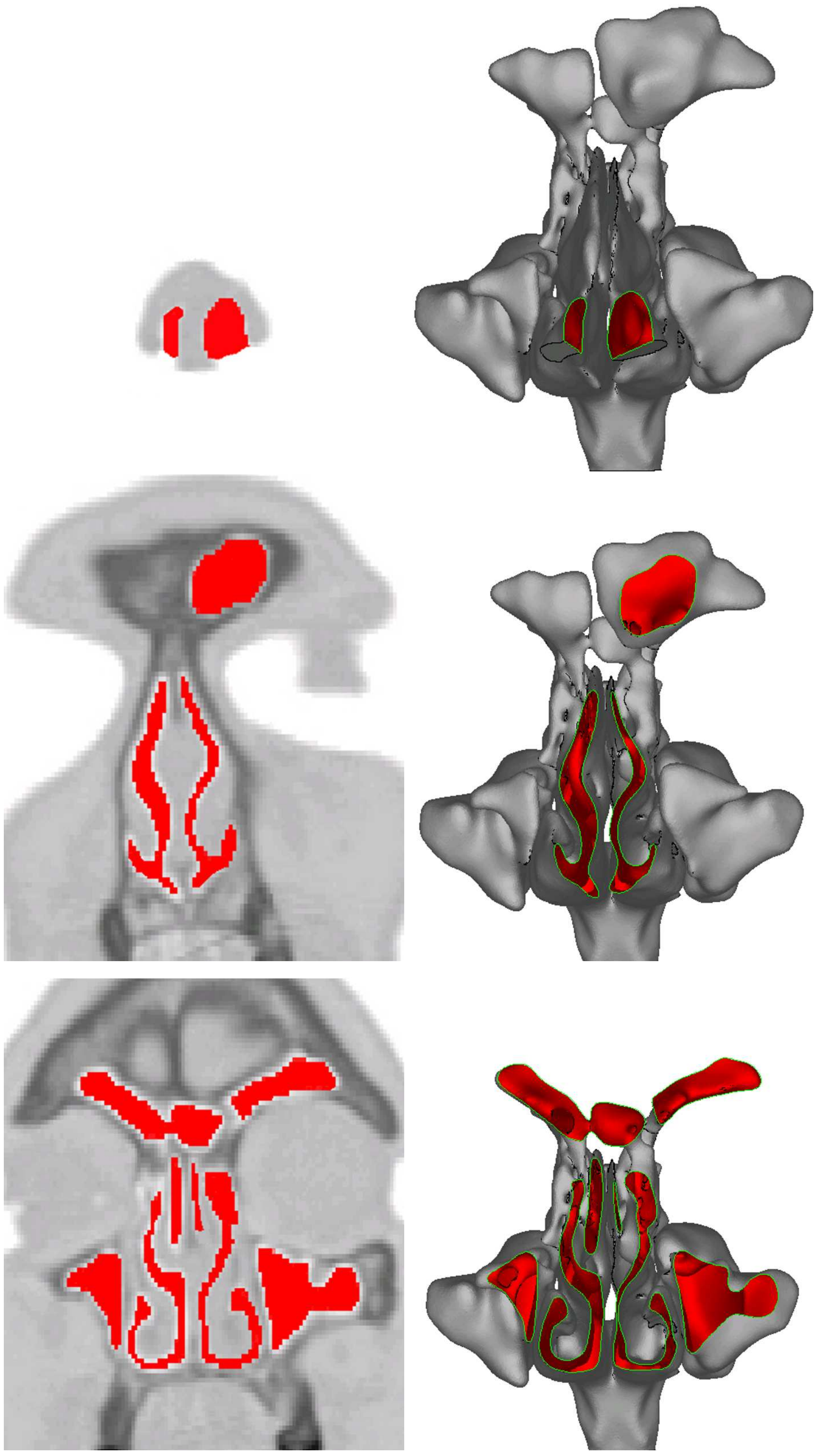


FIGURE 6.3. Segmented CT data (left) and cross-section of the STL surface representation (right) both from top to bottom at the planes $y = r_2 = 0.01, 0.04, 0.05$ m.

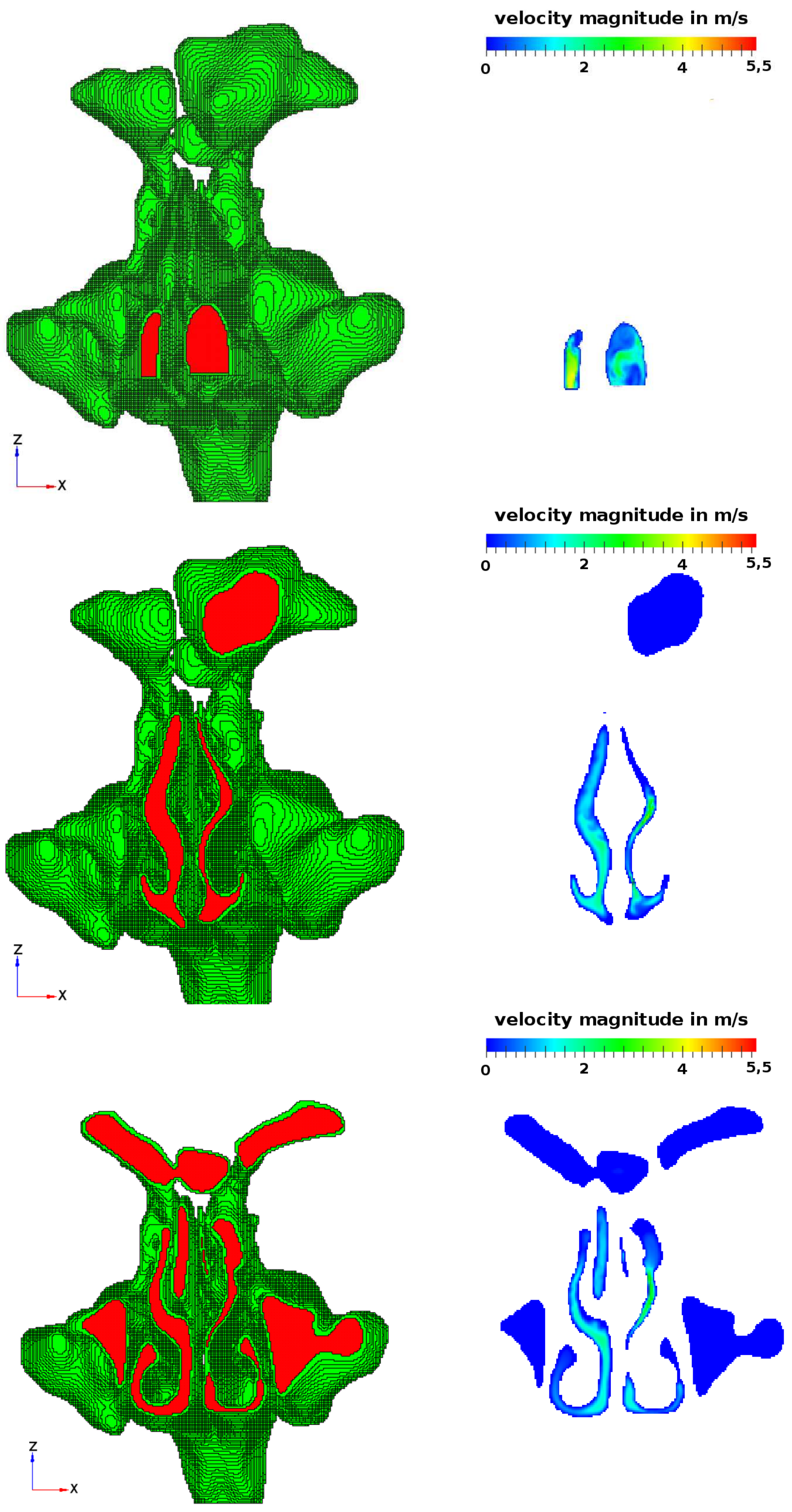

velocity magnitude in m/s
0
2
4
5,5
z
x
velocity magnitude in m/s
0
2
4
5,5
z
x
velocity magnitude in m/s
0
2
4
5,5
z
x

FIGURE 6.4. Cross-section of the voxel mesh (left) and simulation results (right) obtained for a ventilation rate of 250 m/s both from top to bottom at the planes $y = r_2 = 0.01, 0.04, 0.05$ m.

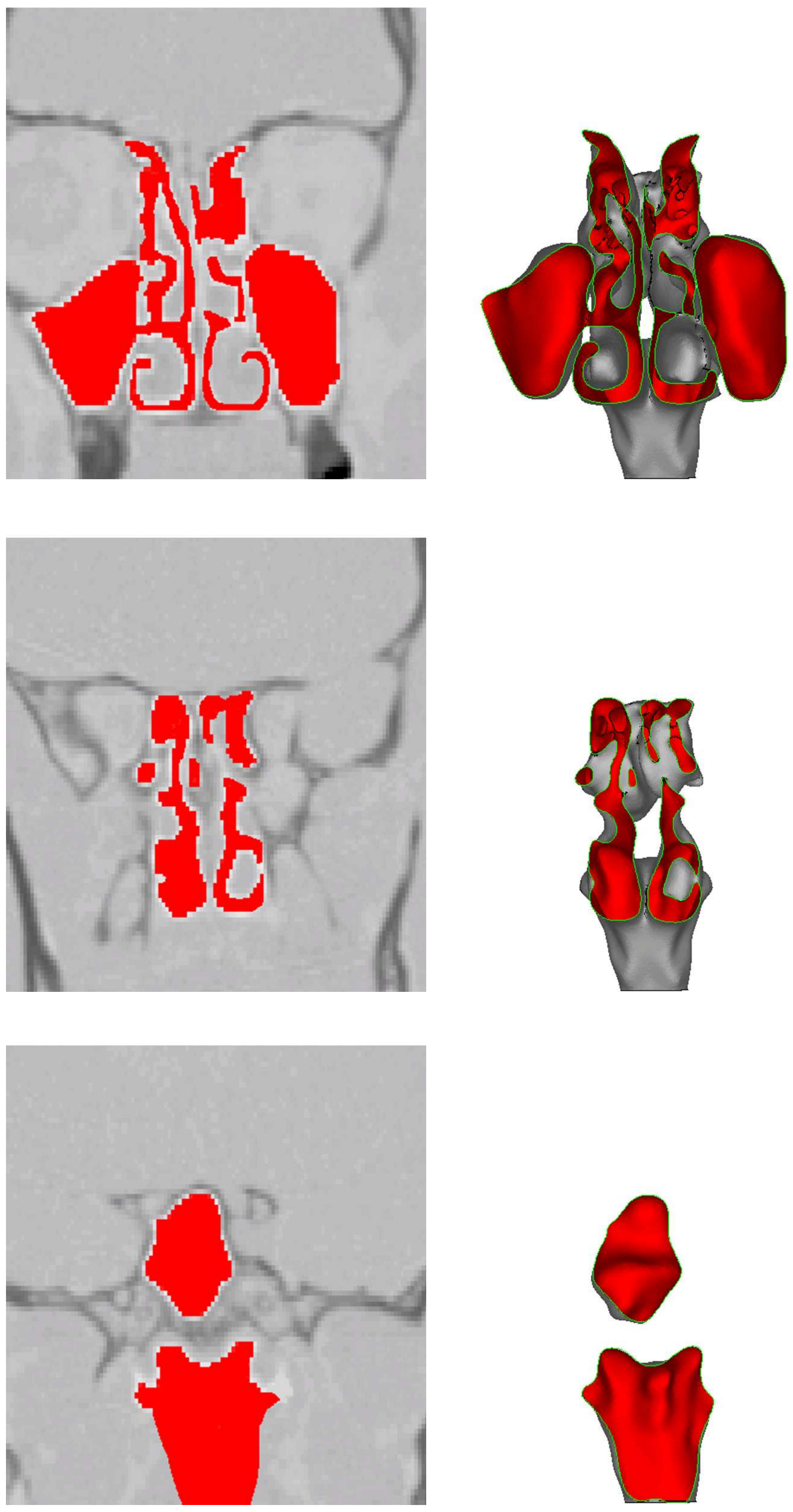


FIGURE 6.5. Segmented CT data (left) and cross-section of the STL surface representation (right) both from top to bottom at the planes $y = r_2 = 0.07, 0.09, 0.10$ m.

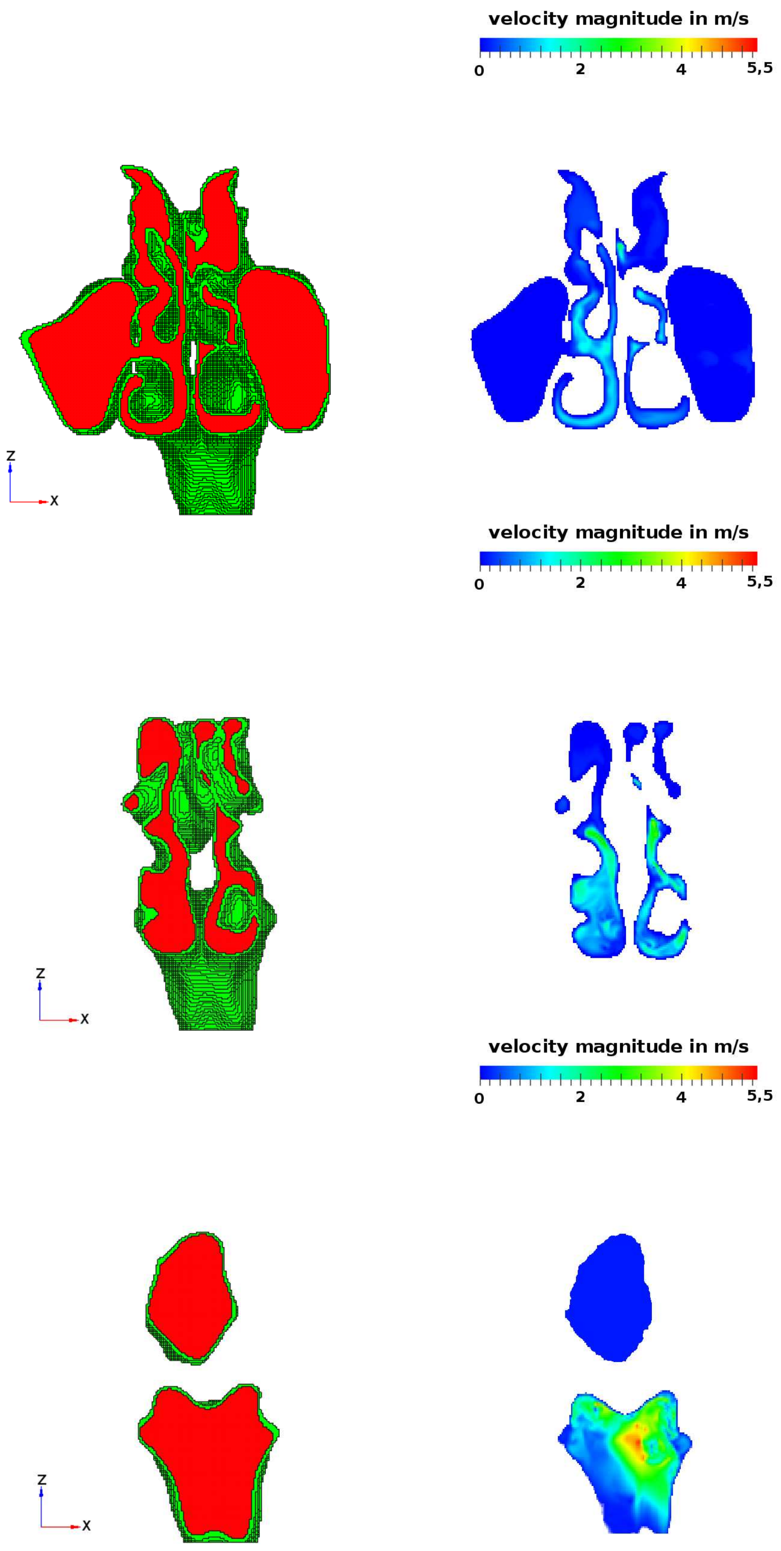

velocity magnitude in m/s
0
2
4
5,5
z
x
velocity magnitude in m/s
0
2
4
5,5
z
x
velocity magnitude in m/s
0
2
4
5,5
z
x

FIGURE 6.6. Cross-section of the voxel mesh (left) and simulation results (right) obtained for a ventilation rate of 250 m/s both from top to bottom at the planes $y = r_2 = 0.07, 0.09, 0.10$ m.

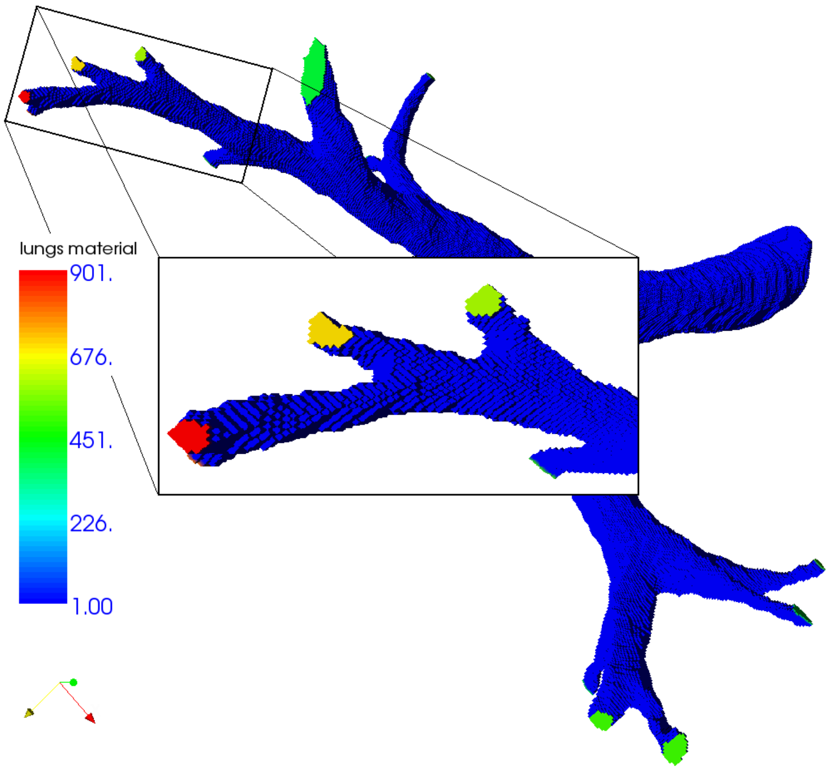


FIGURE 6.7. Voxel mesh representation of the considered human lungs with zoom to some bronchioles of lower generations up to order 9. Different voxel colours indicate different boundary conditions. The discretisation parameter is set to the value $h = 0.23$ mm. In Table 6.6 the material numbers belonging to the considered bronchioles are listed.

in $k \in \mathbb{N}$ disjoint parts Γ_h^i, i.e. $\Gamma_h = \bigcup_{i=2}^{k+1} \Gamma_h^i$. The basic idea in order to obtain a representation of $\Omega_h \cup \Gamma_h$ is to extend $\Omega_h \cup \Gamma_h$ to a cuboid-shaped lattice $\widetilde{\Omega}_h$ which can be identified by a d-dimensional matrix $\boldsymbol{A} \in \times_{j=1}^{d} \mathbb{N}^{n_j}$ where $n_j \in \mathbb{N}$ for $j = 1, 2, ..., d$. Then, each $\boldsymbol{r} \in \widetilde{\Omega}_h$ is mapped to exactly one entry $\boldsymbol{A}(\boldsymbol{r})$ in $\boldsymbol{A}$. In the following, its value is said to be the *material number*. It is defined according to

$$\boldsymbol{A}(\boldsymbol{r}) := \begin{cases} 1 & : \text{ if } \boldsymbol{r} \in \Omega_h \\ i & : \text{ if } \boldsymbol{r} \in \Gamma_h^i \\ 0 & : \text{ otherwise .} \end{cases} \tag{6.1}$$

With it, as it is illustrated in the next subsection, a generic setting up of LB simulations is simplified. To obtain the matrix $\boldsymbol{A}$ from a given surface representation of a given geometry is the aim of this subjection.

The lattice $\Omega_h \cup \Gamma_h$ can be interpreted geometrically. Each $r \in \Omega_h \cup \Gamma_h$ is seen as the centre of a hexahedron with a length, hight and depth of h. The hexahedrons are cubes which are in the following referred to as *voxels*. Consequently, a lattice is also said to be a *voxel mesh*. There exists a varity of techniques that enable obtaining such meshes for a given surface mesh and for arbitrary $h \in \mathbb{R}_{>0}$. Among them one finds methods known as *ray tracing*, *ray stabbing* or *parity count*. The

material number	description	number voxel
0	no fluid	$11,451,556$
1	fluid	$1,065,810$
2	wall	$325,887$
3	nasopharynx	510
4	right nostril	425
5	left nostril	672

TABLE 6.2. Obtained number of voxels which are assigned to different material numbers for the lattice $\Omega_h \cup \Gamma_h$ with $h = 0.45$ mm discretising the considered human nasal cavity. The corresponding voxel mesh is shown in Figure 6.3 and Figure 6.5.

latter two are proposed by Nooruddin et al. in [**105**]. They are implemented in the framework of the *Common Versatile Multi-purpose Library for C++* (CVMLCPP) which is an open source software invented by Beekhof [**17**]. Similar techniques are offered by the commercial software *HyperMesh* [**4**]. The software additionally offers a graphical user interface that allows an interactive manual marking of single voxels but also of sets of voxels. This facility becomes very useful when considering the geometry of an upper lungs model which incorporates various inflows, respectively, outflows. In the considered model one is faced with 15 inflow and one outflow areas and vice versa (cf. Figure 6.7). Therefore, for the voxelising and marking process *HyperMesh* version 10.0 is used to obtain $\boldsymbol{A}$. However, aiming to automatise a marking process which is especially designed for human lungs or nose geometries an adaption of the CVMLCPP implementation seems very attractive to enable patient-individual simulations.

For the geometry of the nose cavity setting $h = 0.45$ mm, the voxelising and marking process leads to a mesh with in total $1,393,304$ voxels. The assigned material numbers as well as the corresponding numbers of voxels are listed in Table 6.2. In Figures 6.3 and 6.5 the obtained voxel mesh is visualised by means of its material numbers.

The considered lungs geometry is discretised for $h = 0.23$ mm. This leads to $2,043,832$ fluid and $283,525$ boundary voxels. Here, the material numbers $0, 1, 2$ are set as done for the nose cavity. The material number of voxels at the trachea inlet, respectively, outlet is set to the value 3 while those of the 15 inlets, respectively, outlets corresponding to the considered bronchioles are set to the value $100m + i$. Whereby, m represents the generation of the actual bronchiole according to the numbering of Weibel [**135, 136**], and $i = 1, 2, ...$ serves as counter to distinguish the bronchioles of the same generation. The actual numbering is given in Table 6.6, and the obtained mesh for the lungs is visualised in Figure 6.7.

6.1.3. From Voxel Mesh to Simulation. This subsection is dedicated to describe the last part of the presented preprocessing approach for complex geometries. Thereby, the way to go starts with a discrete representation of a geometry equipped with additional information regarding different types of boundaries. The data are provided by means of a d-dimensional matrix $\boldsymbol{A}$ which is defined according to (6.1). The way ends with the setting up of an LB simulation, i.e. with step 2 of

Algorithm 1.

A challenge is posed by the realisation of an automated assignment of boundary conditions to particular voxels of the geometry. The reason is that in general boundary conditions in LBM must be defined on a mesoscopic base, i.e. that the distribution function f_i is to be defined for in the fluid pointing directions $v_i \in Q^{in}$ (cf. Subsection 2.1.4). This is not a problem for the *bounce back* condition which is imposed to realise a *no-slip* wall. However, it becomes a challenge for the realisation of inlet and outlet conditions where particular macroscopic moments are fixed. Here, the number of in the fluid pointing directions $v_i \in Q^{in}$ varies depending on the neighbourhood of the considered node $\boldsymbol{r} \in \Gamma_h$. Frequently used approaches like those of Inamuro et al. [**73**], Skordos [**124**] or Latt [**88, 89**] use finite differences to compute the wanted distributions f_i for $v_i \in Q^{in}$. Latt realises the mentioned methods in the framework of OpenLB for 46 cases of possible neighbourhoods for the three-dimensional case $d = 3$ (cf. OpenLB user guide associated with release 0.4 [**86**]). Yet, in practice these classification is not sufficient to handle all possible cases emerging when considering complex geometries. Zimny faces this challenge in [**146**]. At first, the number of all possible cases is reduced by posing the following conditions, namely

(1) Every single boundary voxel is at least neighbour of one fluid voxel.
(2) Non of the fluid voxels is a neighbour of any non-fluid voxel.

Then, routines are realised which enable to modify the originial voxel mesh matrix $\boldsymbol{A}$ automatically such that the two conditions hold. Additionally, with the help of a refinement algorithm the number of cases can be reduced to 96. The refinement scheme splits one voxel into N^d where $N \in \mathbb{N}$ and is in the following called the *refinement level*. Afterwards, in accordance with the first of the two conditions stated above all not needed boundary voxels are removed. Further, Zimny succeeds in leading back the 96 cases to the before mentioned 46 categories. An automated assignment of one of the 46 to a boundary voxel is implemented and validated by comparing the results for a benchmark problem, namely the flow around a round cylinder, with those obtained by others which are condensed in the work of Schäfer et al. [**118**].

For the two considered geometries the mentioned techniques realised by Zimny are applied to automise the initialisation process. The marcroscopic values imposed for the inlet and outlet conditions are provided by means of one function for each different boundary type $i = 2, 3, ...$ of Γ_h^i.

6.2. Numerical Simulation of Intranasal Flows

In this section, numerical simulation results of intranasal flows are presented. The aims of this study are, firstly, to illustrate the application of parallel LBM for a realistic complex problem and, secondly, to provide validated insights in the flow characteristics of intranasal flows. Especially the latter goal is interesting since the underlying computational domain is obtained from a patient with a diagnosed pathology. Thereunto, emphasis is placed to reveal possible abnormal specifics.

Flow characteristics in the human nose have already been studies both experimentally and numerically. Thereby, the focus is often placed to reveal the morphological dependencies on the flow regime. For example, the experimental-based studies of Ploetz [**113**], Courtiss and Goldwyn [**30**], Scherer et al. [**119**] and Elad et al. [**33**] discuss controversially the role of the turbinates concerning laminarity and turbulence. Churchill et al. [**29**] also addresses this issue but varies other morphological parameters like the nostrils angle. Furthermore, this work is interesting hence a detailed review about further investigations of morphological issues is provided. Numerical simulations of intranasal flows have been performed at least since 1995 (cf. Keyhani et al. [**79**]). Since then, correlated with the increase of computational power the underlying geometry and physical models have become more and more complex. In [**74**] for example, an overview of recently published articles in this context is provided. LBM are considered to face the problem by Freitas [**39, 40**] and Finck et al. [**38**]. In contrast to the here presented approach, in many studies the full complexity of the geometry, in particular the sinuses, is not mapped (cf. [**38, 39, 40, 71, 74**]). Moreover, convergence studies of the numerical simulations are hardly provided at all. In the remainder of this section other approaches are mentioned, whereby further distinctions are stressed.

In order to challenge the request to obtain validated results, two problems are formulated and solved. The problem specification and the chosen LBM are stated in detail in Subsection 6.2.2. In the following subsection, at first an analysis of the convergence behaviour of the numerical results obtained for the first problem with different underlying discretisation parameters is presented. Then, results of this test case are validated by comparing them to data obtained both numerically and experimentally by others. Finally, samples of the obtained results are presented in form of pictures of the computed flow field. The results of the second considered problem are presented and discussed in Subsection 6.2.4 which is the last part of this section. There, the numerical results are compared to measurements obtained for the very patient. Another subsection is placed in front of all others. It is devoted to present the common professional terminology concerning the anatomy of the human nose as well as generally accepted facts about its physiology.

6.2.1. About the Anatomy and Physiology of the Human Nose. In this subsection a brief overview is given. What follows is gathered from standard literature [**102, 120, 122, 145**] and displays a terminology and points of views concerning the physiology which is widely accepted.

The nose of a human is located centrally on the face continuing inside the head. Its outer visible part is protuberance-shaped and houses the *nostrils* which are two bean-shaped holes admitting the inspiration and expiration of air. In Figure 6.8 the outer part is made visible by means of a surface mesh which is generated from CT data as described in Subsection 6.1.1. The shape of the whole nose is determined by the *ethmoid bone* and the *nasal septum.* The ethmoid bone separates the nose cavity from the brain. The nasal septum consists mostly of cartilage. It separates the two nostrils and continues in the inner part of the nose building a barrier which forms two independent airways which are called in the following the *right* and the *left airway.*

The inner part of the nose hosts pairs of the *nasal vestibule*, the *nasal cavity*, pairs of several *paranasal sinuses* and the *nasopharynx.* The nasal vestibules are located right behind the nostrils. The nasal cavity consists of the two airways. Three horizontal outgrowths, called *turbinates* or *conchae*, divide the two airways in three channels which are known as the *superior*, the *middle* and the *inferior (nasal) meatus.* Paranasal sinuses are spaces filled with air which are located around the nasal cavity. Among them one finds the *maxillary sinuses*, which are the largest sinuses situated right and left of the airways, and the *frontal sinuses* which are the second largest based on top of the nasal cavity. Behind the nasal cavity the two airways fuse in the nasopharynx which is the upper part of the *pharynx.* Through the pharynx which partly belongs to the digestive system the air passes to the rest

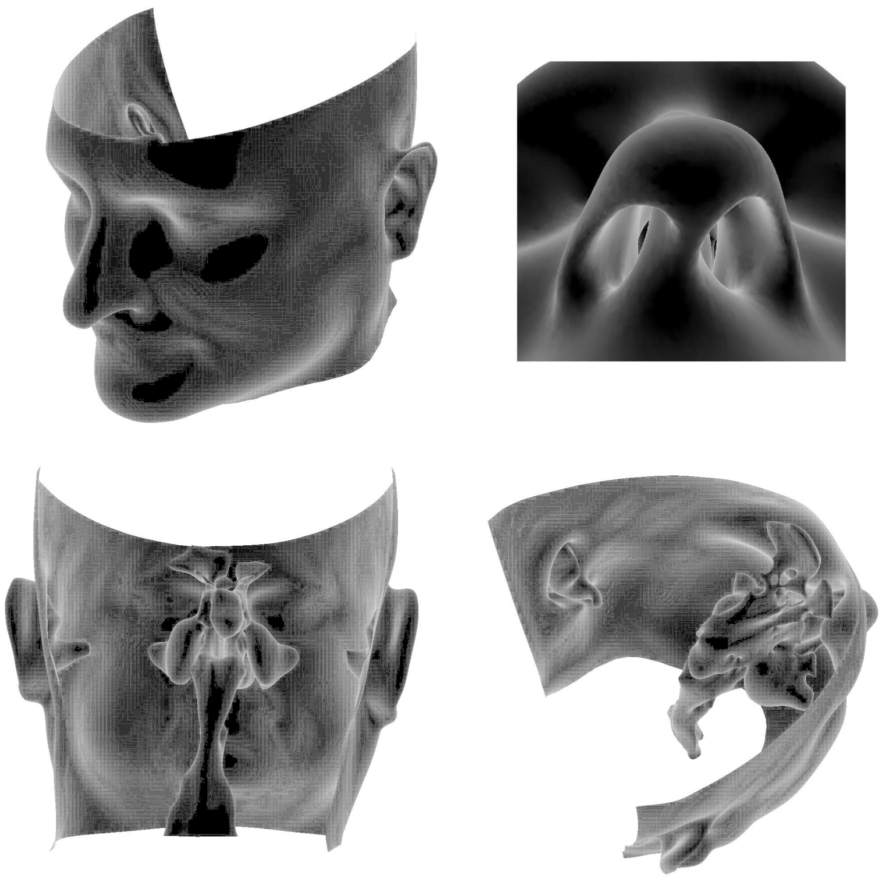


FIGURE 6.8. The outer part of the nose is pictured on the two images on the top. The left image shows the surface of the left part of the face with the nose right in the middle of the face while the right one depicts a closer look of the nostrils. Thereunder, the inner part of the nose is displayed in two images. In both pictures the surface of the face is shown from behind revealing a view inside the head. In front of the surface the inner part of the nose is visible. The images are obtained from CT data of the investigated person as described in Subsection 6.1.1 in joint work with Mayer and Weber in the framework of the United Airways project [**16**].

of the respiratory system.

Medical scientists have different opinions concerning the physiology of the human nose. According to [**102, 120, 122, 145**] generally accepted are the following functionalities:

- enables breathing,
- smelling,
- conditioning of inhaled air: warming, making it more humid, filtering.

Yet, ambiguous is e.g. whether the nasal cavity and the sinuses are vocal resonators for the human voice or not [**102, 145**].

Another phenomenon is observed, namely that the flow rate ratio of the two airways constantly alters. That means that at one time more air is passes through the right than the left airway and at another time the other way around. It is called the *nasal cycle* and researched e.g. by Reinefeld [**114**].

Details concerning the above issues have been in focus of research for a long time. For example, the air conditioning capability of the nose is already discussed by Lehmann [**91**] in 1933. However, many questions are not answered yet. Nowadays, numerical simulations deliver new insights. The impact of the geometry on the nose flow is numerically investigated recently, for instance by Kelly et al. [**78, 77**], Weinhold et al. [**137**], Hörschler et. al [**71**], Zachow et al. [**141**] and Finck et al. [**38**]. Inthavong et al. [**74**] considers the conditioning of the inhaled air also with the help of numerical simulations.

6.2.2. Problem Formulations and Discretisation Issues. In this subsection at first two series of fluid flow problems are formulated. Then, the numerical methods to solve the problems are stated. Emphasis is placed to formulate the problems in a way that the results can be validated. The results of the first series are to be compared to other numerically and experimentally gained results. The second test suite is conceived such that the results can be validated by measurements obtained by a method which is known as rhinomanometry (cf. Subsection 6.2.4).

The flow of air in a human nose which occurs during an inspiration or an expiration is to be simulated for different fixed flow rates Fl. Breathing is naturally a transient process. In that context the question arises if the problems are well-defined, i.e. if for every considered flow rate Fl there exists in each case exactly one solution. Providing that this is the case, it is expected that applying an LB scheme leads in any of the considered cases to a unique discrete solution. If convergence towards these *pseudo steady states* is observed the simulation will be stopped.

The underlying geometry is the inner nose of the before mentioned Central European male. It is to be noted, that for this patient a peripheral obstructive ventilation disorder is diagnosed by Giotakis [**16**]. The geometry data are obtained by CT scans and prepared for the simulation as described in the previous section. With these preprocessing steps a discrete representation of the inner nose is obtained for the discretisation parameter of $h = 0.45$ mm$/N$ whereby $N \in \mathbb{N}$ is the refinement level belonging to the refinement strategy introduced in Subsection 6.1.3. Different boundary regions are defined as stated in Table 6.2.

The first series of tests considers expirations with one inflow which is located at the nasopharynx and two outflows at both nostrils. The other boundary is assumed to be a wall which does not allow the air to slip. The considered flow rates vary, namely $Fl = 100$ ml/s, 125 ml/s, 167 ml/s, 250 ml/s, 333 ml/s . These rates are typically measured for male adults who rest, sit awake or do light exercises (cf. Table 6.1). In the following, these cases are referred to as *ex total*.

The second test suite aims to validate the results by measurements obtained by rhinomanometry. Here, inspirations and expirations with outflows and, respectively, inflows which are located at the nasopharynx are considered. Yet, in the considered cases there is only one outflow or, respectively, one inflow, either at the right nostril or the left nostril. The other nostril is assumed to be closed, i.e. the boundary condition at Γ_h^4 or Γ_h^5 is to adapted as a no-slip condition. This leads to two cases which are referred to as *re right* and, respectively, *re left*. The underlying geometry is chosen as for the *ex total* case. The flow rates Fl of interest vary up to

material number	description boundary	applied boundary conditions for cases		
		ex total	*re right*	*re left*
2	wall	bounce back	bounce back	bounce back
3	nasopharynx	velocity bc	velocity bc	velocity bc
4	right nostril	pressure bc	pressure bc	bounce back
5	left nostril	pressure bc	bounce back	pressure bc

TABLE 6.3. This table provides an overview of the considered tests. The fluid flow problems basically differ in the applied boundary conditions at particular boundary parts. With the *bounce back* condition a no-slip wall can be simulated. A boundary condition of type *velocity bc* fixes the macroscopic velocity and *pressure bc* the pressure at the boundary to a certain value.

a maximum flow rate of $Fl = 250$ ml/s.

The types of boundary conditions which are applied at the different boundary parts Γ_h^i $(i = 2, 3, 4, 5)$ depend on the actual considered problem. Thereto, in Table 6.3 an overview is given.

The no-slip condition which is applied at the walls and, if required, at a closed nostril is realised by the *bounce back* condition as stated in Subsection 2.1.4. At the nasopharynx the macroscopic velocities are fixed to certain values. Simplifying, it is assumed that the velocity is distributed as in a pipe where Poiseuille's law holds. Thus, the velocity is a quadratic function of the distance from the centre of the plane Γ_h^3. The magnitude of the velocity distribution is increased linearly with the number of time steps up to $10,000$ steps. Thereby, the slope is chosen so that the desired flow rate is reached at time step $10,000$. This start-up process aims to avoid unnatural behaviour as well as numerical artefacts caused by e.g. violation of smoothness. The boundary conditions at the open nostrils are set as pressure conditions, i.e. the pressure is fixed while the macroscopic velocity is not. Both macroscopic boundary conditions, namely the velocity and pressure condition, are realised for the $D3Q19$ model by means of interpolation schemes which are proposed by Inamuro et al. [**73**] as they are mentioned in Subsection 2.1.4 in more detail.

For both test suites the pressure at the open nostrils is fixed to be $1,013$ hPa and the air is assumed to flow at a temperature of $T = 20\,^\circ$C. Thus, the air can be considered at normal conditions so that its speed of sound is $c_s = 343$ m/s, its density is $\rho = 1.225$ kg/m^3 and its kinematic viscosity is $\nu = 1.4 \cdot 10^{-5}$ m^2/s. The area of the boundary Γ_h^3 at the nasopharynx is found to be $A = 1,0328$ cm^2. To reach the desired flow rates Fl the mean speed $U_{mean}(Fl)$ that is needed can be computed. Let the characteristic macroscopic velocity be set to this mean speed. Further, let the characteristic macroscopic length be fixed to

$$L := \max_{\boldsymbol{r}\in\Gamma_h^3} r_1 - \min_{\boldsymbol{r}\in\Gamma_h^3} r_1 = 0.0162 \text{ m} .$$

Then, for each considered flow rate Fl the Reynolds number Re and the Mach number Ma can be computed. Table 6.4 shows the obtained characteristic quantities for different flow rates.

Due to the relative small Mach numbers and the fact that under normal conditions air is a Newtonian fluid, the flow can be modeled by means of the BGK-Boltzmann equation (1.67) (cf. Chapter 1). To solve the problems numerically an

Fl	U_{mean}	Re	Ma
100 ml/s	0.97 m/s	1,120	0.0028
125 ml/s	1.21 m/s	1,401	0.0035
150 ml/s	1.45 m/s	1,681	0.0042
167 ml/s	1.62 m/s	1,871	0.0047
250 ml/s	2.42 m/s	2,801	0.0071
333 ml/s	3.22 m/s	3,731	0.0094

TABLE 6.4. Obtained characteristic speeds U_{mean}, Reynolds numbers Re and Mach numbers Ma for different flow rates Fl. The air is considered at normal conditions (1,013 hPa, 20 °C). Thus, the kinematic viscosity is $\nu = 1.4 \cdot 10^{-5}$ m^2/s and the speed of sound is $c_s = 343$ m/s. The characteristic length is defined by $L := \max_{\boldsymbol{r}\in\Gamma_h^3} r_1 - \min_{\boldsymbol{r}\in\Gamma_h^3} r_1 = 0.0162$m.

LBM is applied. The chosen model is the $D3Q19$ model as it is derived in Section 2.1 and the pressure and velocity boundary conditions are realised as proposed by Skordos [**124**].

In order to reduce the number of time steps needed to reach steady states, in the following the problems are considered after a scaling. Thereto, the characteristic speeds and the kinematic viscosity are set to

$$\begin{aligned} \widetilde{U}_{mean} &:= \frac{U_{LB}}{h} U_{mean} \ , \\ \widetilde{\nu} &:= \frac{U_{LB}}{h} \nu \end{aligned} \tag{6.2}$$

for different, so called *lattice speeds* $U_{LB} := 0.1, 0.05, 0.01$. With this, the Reynolds numbers stay the same while the new Mach numbers $\widetilde{Ma}$ increase. According to (2.1) the speed of sound in an LB simulation is set to $c_s = h^{-1}$. Thus, one obtains $\widetilde{Ma} = U_{LB} U_{mean}$ which is in all considered cases greater than the actual Mach number Ma. If one restricts for $Fl = 333$ ml/s the choices for the lattice speeds to $U_{LB} = 0.05, 0.01$, it will hold that $\widetilde{Ma} < 0.3$. For simplification reasons, the density $\rho = 1.225$ kg/m^3 is scaled to the value of $\widetilde{\rho} = 1$. As initial condition serves the Maxwell distribution with $\boldsymbol{u} = \boldsymbol{0}$ and $\widetilde{\rho} = 1$. The obtained results are finally rescaled to the original standard unit system.

6.2.3. Presentation and Discussion of the Numerical Results I.

In this subsection the obtained numerical results of an expiration through both nostrils are presented and discussed. The flow rates are fixed to be $Fl = 100$ ml/s, 125 ml/s, 167 ml/s, 250 ml/s, 333 ml/s. In the previous subsection, all considered test cases are specified in detail. There, they are referred to as *ex total*.

All numerical experiments are carried out in parallel either on the *HP XC4000* or on the *JUROPA* high performance computer which are shortly described in Section 3.3. Thereby, both the full hybrid parallelisation approach as introduced in Chapter 3, which is realised using OpenMP and MPI, and its pure MPI-based part are employed. The hybrid parallel code related to the problem with $N = 2$, $U_{LB} = 0.05$ and $Fl = 125$ ml/s takes about four days to be executed on the *HP XC4000* employing in total 128 cores on 32 nodes. The set-up time is included but

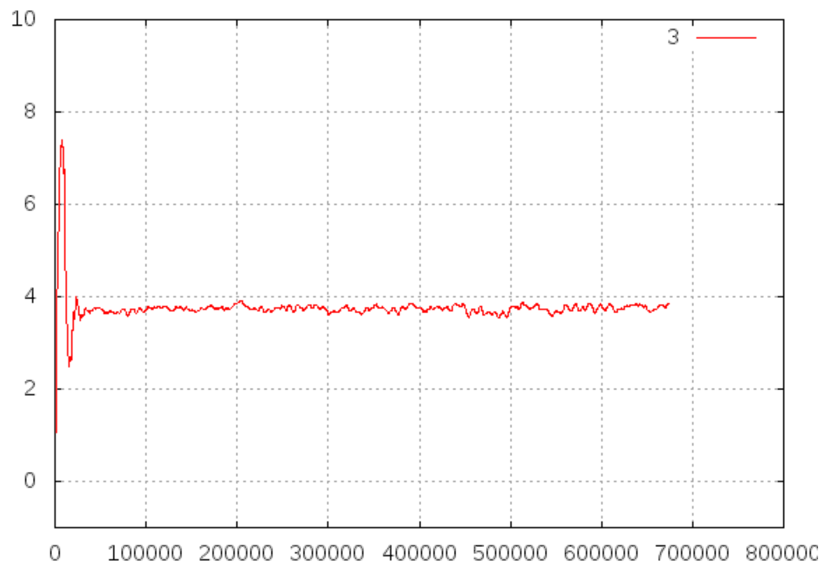


FIGURE 6.9. Time evolution of the total pressure drop $p_{\text{total}}(Fl)$ at an expiration flow rate of $Fl = 125$ ml/s towards a pseudo steady state. The lattice speed is chosen to be $U_{LB} = 0.05$, and the refinement level is set to $N = 3$.

for this example no output files for visualisation are written.

The quantities of interest in the following are the pressure drops

$$p_{\text{total}}(Fl) := p_{\text{nasopharynx}}(Fl) - p_{\text{nostrils}}(Fl) \tag{6.3}$$

occurring between the pressure at the inflow $p_{\text{nasopharynx}}$ and the outflows p_{nostrils}. The positions, where the values for the pressures are taken, are always fixed to be in the fluid close to the position in the centre of the area Γ_h^i, namely at

$$\boldsymbol{r}^i_{\text{mid}} := \frac{1}{2}\begin{pmatrix} \max_{\boldsymbol{r}\in\Gamma_h^i} r_1 + \min_{\boldsymbol{r}\in\Gamma_h^i} r_1 \\ \max_{\boldsymbol{r}\in\Gamma_h^i} r_2 + \min_{\boldsymbol{r}\in\Gamma_h^i} r_2 \\ \max_{\boldsymbol{r}\in\Gamma_h^i} r_3 + \min_{\boldsymbol{r}\in\Gamma_h^i} r_3 \end{pmatrix} - hN\boldsymbol{n} \qquad (i = 3, 4, 5)\ . \tag{6.4}$$

At first, the convergence towards pseudo steady states for the simulated total pressure drops $p_{\text{total}}(Fl)$ is studied. Thereunto, exemplary the case with an expiration flow rate of $Fl = 125$ ml/s is studied. For all other cases a similar behaviour is observed. In Figure 6.9 the result for $p_{\text{total}}(Fl)$ is plotted as a function of time steps obtained for an LB simulation with a lattice speed of ($U_{LB} = 0.05$) based on a lattice which is refined with level $N = 3$.

The plot shows that after a certain number of time steps the quantity of interest oscillates in a certain range. This behaviour is also observed for other choices of lattice speeds ($U_{LB} = 0.1, 0.05, 0.01$) and refinement levels ($N = 2, 3, 4$). Thereby, the frequency is found to be the smaller the greater the lattice speed is chosen. For the amplitude no significant differences are observed. The reasons for this noise might be related to the choice of the boundary condition at the outlet (see the discussion of artificial boundary conditions and especially *do-nothing* pressure boundary conditions for similar problems in Specovius-Neugebauer et al. [**20, 126**]). To quantify the observations the minimal $p^-_{\text{total}}(125)$ and maximal $p^+_{\text{total}}(125)$ computed values for the pressure drop occurring between time step $400,000$ and $600,000$ are determined. Then, in order to obtain a basis for further comparisons, $\overline{p}_{\text{total}}(125)$ is

N	U_{LB}	$p^{+}_{\text{total}}(125)$	$p^{-}_{\text{total}}(125)$	$\overline{p}_{\text{total}}(125)$
2	0.05	3.99	4.38	4.19
3	0.10	3.62	3.95	3.79
3	0.05	3.54	3.88	3.71
3	0.01	3.69	3.89	3.79
4	0.05	3.49	3.79	3.64

TABLE 6.5. Obtained results for the total pressure drop for different choices of lattice speeds U_{LB} and refinement levels N. $p^{-}_{\text{total}}(125)$ and $p^{+}_{\text{total}}(125)$ are the minimal and maximal computed values for the pressure drop occurring between time step $400,000$ and $600,000$ while $\overline{p}_{\text{total}}(125)$ denotes their mean.

set to the mean of the minimum and maximum. Similarly, this procedure is also applied to all other in the remainder of this chapter referred results for pressure drops which are consequently marked with an overbar. The quantitative results of the investigated case ($Fl = 125$ ml/s) are given in Table 6.5. They clearly show a satisfying convergence behaviour so that the assumption of the existence of pseudo steady states can be seen as numerically evidenced.

Now, the numerical results are compared to those obtained by three others. Thereunder, one result is obtained experimentally while the other two are obtained by numerical simulations with commercial CFD software. All three results are captured in the graph in Figure 6.10.

The first of the three considered approaches is the experimentally obtained one. Kelly et al. [**77**] consider for their experiments a plastic replica of nasal airways. Its underlying geometry is captured by means of MRI scans of a normal 53 year old male (cf. Swift [**130**]). The cast does not encompass all details of a human nose. In particular, the maxillary and frontal sinuses are not captured. However, the two airways up to the nasopharynx are considered in similar measures to those of the underlying geometry researched here.

The second data for comparison is computed numerically by Weinhold and Mlynskiet [**137**]. The referred results are obtained for a 27 year old male with a normal nasal anatomy. The computational domain is obtained by reconstructing a CT scan with a resolution of 1 mm to a 3D model whereby the sinuses are not captured. A finite volume scheme is employed to solve the incompressible Navier-Stokes equation together with a turbulence model of k-ϵ-type.

The last dataset is provided by Inthavong et al. [**74**] who consider a more complex physical model which describes besides the flow also the heat transfer in a human nose. The underlying geometry is the inner nose of a 25 years old healthy Asian male, again, without the sinuses. The 3D model is constructed based on CT scans. For the computation of flow rates up to 250 ml/s a laminar and for high flow rates a turbulent k-ϵ-type model is employed.

All three cases differ to a greater or lesser extent to each other but also to the here considered case. Distinctive features are e.g. their underlying geometries, the chosen physical models or the employed numerical methods. Disregarding these differences, all results for the pressure drops are found fairly close to each other.

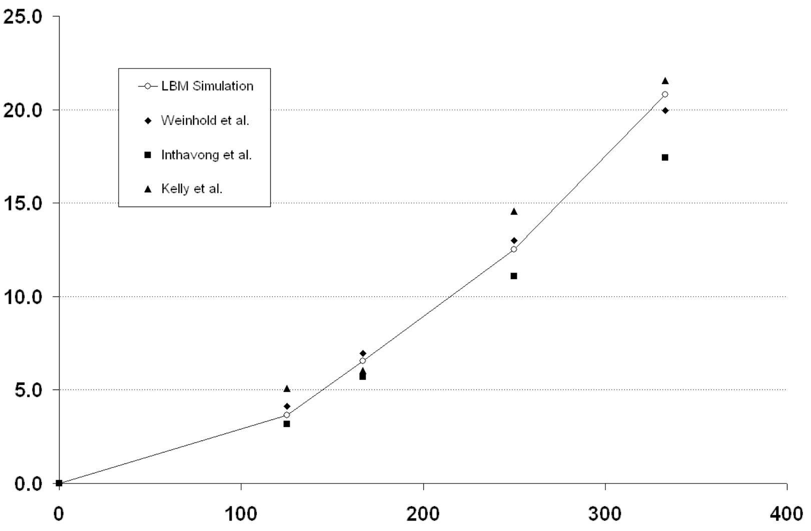


FIGURE 6.10. Comparison between the numerical results for the pressure drop $\overline{p}_{\text{total}}(Fl)$ (Pa) for different flow rates Fl (ml/s) and the results obtained numerically by Weinhold et al. [**137**], Inthavong et al. [**74**] and experimentally by Kelly et al. [**77**].

On the one hand it is concluded that the presented approach for the particular case is validated. On the other hand, in light of the mentioned differences critical questions concerning the informative value of the computed total pressure drop arises. Further, for the considered patient a peripheral obstructive ventilation disorder is diagnosed (cf. Giotakis [**16**]). Yet, the results are found in good agreement with other results which are founded on data from patients who are considered to have no pathologies. This leads to the reasoning that the benefit of simulating the total pressure drop seems to be limited for medical applications.

The obtained velocity field for a flow rate of $Fl = 250$ ml/s is presented in the Figures 6.3 and 6.5 by means of a coloured representation of the velocity magnitude in six chosen cross-sections. A three-dimensional impression of the computed flow field is suggested through the pictures in Figure 6.11. There, the velocity magnitude is visualised by a number of coloured three-dimensional spheres for an expiration flow rate of $Fl = 100$ ml/s. Particularly important to note is the right upper picture which clearly displays high velocity magnitudes at the left inferior meatus. Indeed, a deeper analysis reveals that the highest values in the whole domain, namely about 1.64 m/s, are obtained there. Further, it is determined that only approximately 26 ml of the in total 100 ml/s leave the left nostril. This found asymmetrical behaviour may be caused by a stenosis in that particular part of the geometry which is possibly the reason for the diagnosed peripheral obstructive ventilation disorder.

6.2.4. Presentation and Discussion of the Numerical Results II. This subsection is dedicated to present and discuss the numerical results for the test cases *re right* and *re left* which are specified in Subsection 6.2.2. The problems are formulated in a way that a comparison with measured data obtained by a method called *rhinomanometry* is enabled. As before for the case *ex total* all experiments are carried out in parallel applying the same techniques either on the *HP XC4000*

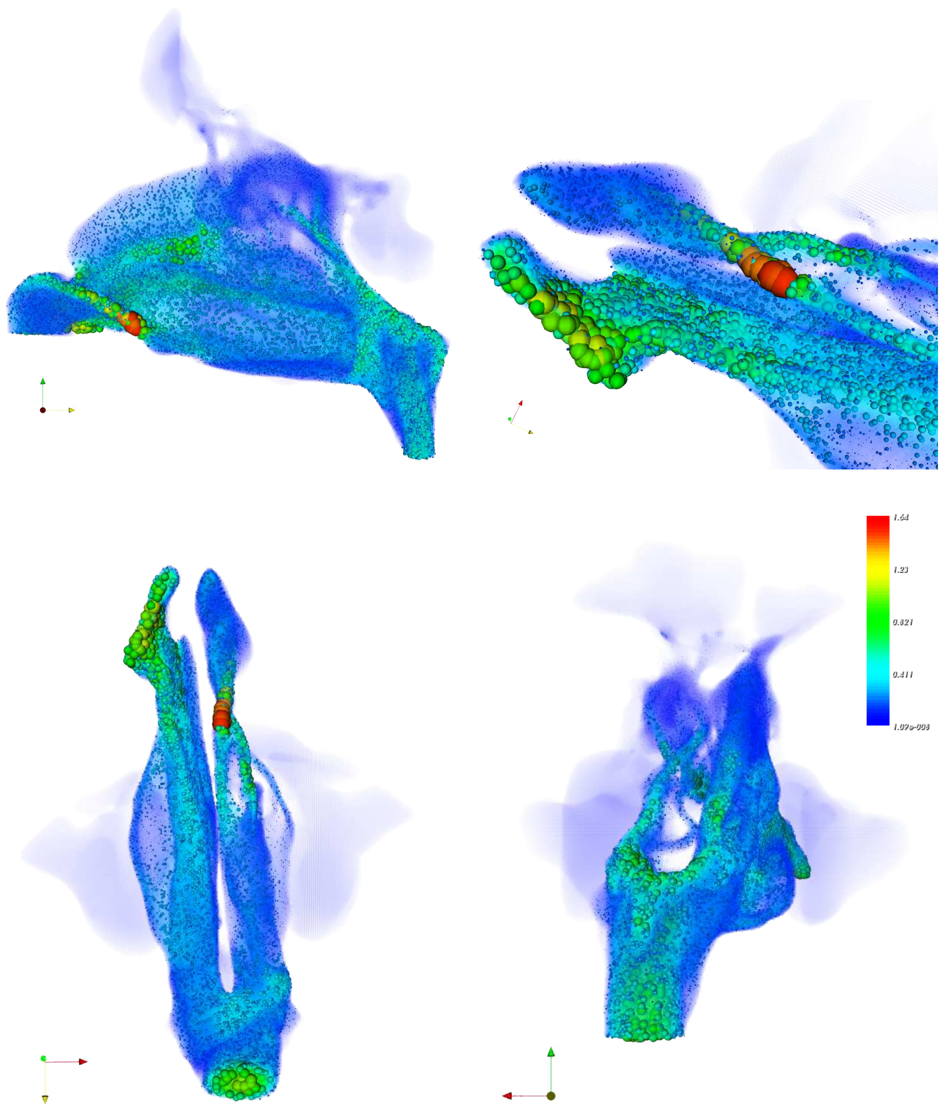


FIGURE 6.11. The simulated flow field for an expiration flow rate of $Fl = 100$ ml/s is shown from different point of views, whereby the right upper picture displays a closer look to a significant narrow at the left inferior meatus. The magnitude of the velocity is visualised as coloured three-dimensional spheres. The lattice speed in the LB simulation is set to $U_{LB} = 0.01$. No refinement is done ($N = 1$); and the simulation is stopped after $550,000$ times steps has been performed.

or on the *JUROPA* high performance computer.

The particular method which is employed to obtain the measurements is referred to as *active anterior rhinomanometry* (cf. [**72**]). During the measuring the patient sits in an upright position. A mask with integrated flow and pressure sensors is places densely upon his outer nose. Then, pairs of pressure drops and flow rates are measured during an inspiration and an expiration through each of the two

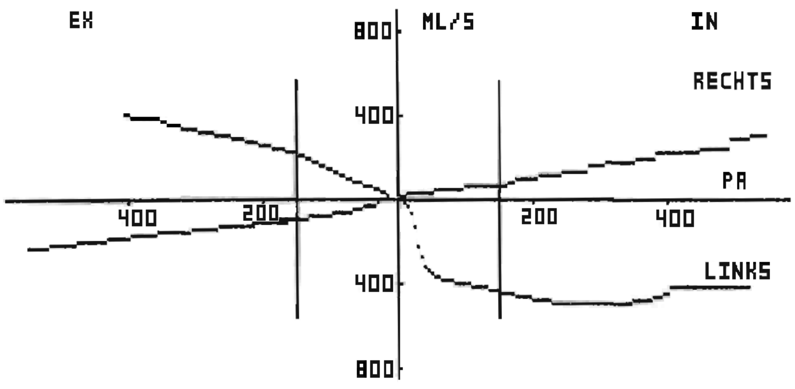


FIGURE 6.12. Scan of the original measured data ${p^m_{\text{right}}}^{-1}$ and ${p^m_{\text{left}}}^{-1}$ which are obtained from the considered patient by means of an active anterior rhinomanometry method. It is to be noted that the inverse functions are plotted.

nostrils independently. A drawing of the schematic setup is given by Huizing et al. in [**72**, Fig. 2.116]. The two obtained series of pairs can be represented by the functions $p^m_{\text{right}}(Fl)$ and $p^m_{\text{left}}(-Fl)$ which correspond to the pressure drops obtained by means of the numerical simulation. They are defined similarly to (6.3), namely

$$p_{\text{right}}(Fl) := p_{\text{right nostril}}(Fl) - p_{\text{left nostril}}(Fl) \ ,$$
$$p_{\text{left}}(-Fl) := p_{\text{left nostril}}(-Fl) - p_{\text{right nostril}}(-Fl)$$

which are taken at $\boldsymbol{r}^i_{\text{mid}}$ for $i = 4, 5$ which are defined as before in (6.4).

The originally measured rhinomanometry data of the very patient whose geometry is considered for the numerical simulations is provided in Figure 6.12. The measurements are just taken once which considerably lessens the informative value since the absence of significant possible measuring errors can not be guaranteed. However, according to Giotakis [**16**] the peripheral obstructive ventilation disorder is clearly displayed by the unusual asymmetric shape of the two graphs. In particular, the extreme flat graph of p^m_{right} is found deviant. In the literature [**72**] $p^m_{\text{right}}(\pm 400 \text{ ml/s}) \approx p^m_{\text{left}}(\pm 400 \text{ ml/s}) \approx 150$ Pa is given as reference value for a healthy human. Yet, the considered rhinomanometry data are found to differ significantly from this standard value.

For the computed pressure drops p_{right} and p_{left} a convergence characteristics towards pseudo steady states similar to those studies in the previous subsection is observed. In order to obtain well defined results the averaging strategy proposed in there is also applied here. In Figure 6.13 the corresponding computed mean pressure drops $\overline{p}_{\text{right}}$ and $\overline{p}_{\text{left}}$ are stated. Thereby, the display format is chosen as those of the rhinomanometry data presented in Figure 6.12. With this, a qualitative comparison is enabled.

The simulation results are found to show similar characteristics as the measured data. The similarities are in particular that the two graphs are not symmetric and the graph of ${\overline{p}_{\text{right}}}^{-1}$ is abnormally flat while those of ${\overline{p}_{\text{left}}}^{-1}$ is much more of a standard healthy shape. Summing up, it is concluded that the experimental and simulation results qualitatively match. Furthermore, these results are in agreement with those observations concerning the found asymmetry of the velocity field and

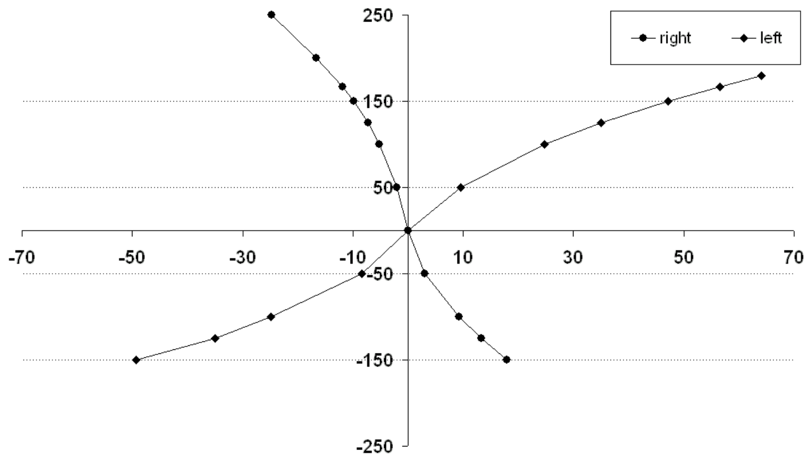


FIGURE 6.13. Simulated pressure drops $\overline{p}_{\text{right}}{}^{-1}$ and $\overline{p}_{\text{left}}{}^{-1}$. In order to enable easy comparison with the rhinomanometry data given in Figure 6.12, the inverse functions of the simulated pressure drops are plotted.

flow rate in the two airways. This further confirms the assumption that a stenosis in a particular part of the geometry is the reason for the diagnosed pathology. In this context, optimisation techniques as they are proposed in Chapter 4 or alternatively in Chapter 5 could help to develop strategies for adequate surgery planning.

6.3. Numerical Simulation of the Flow in the Upper Human Lungs

Describing the respiration in the human lungs seems unequally more difficult than describing it in the human nose. For example, the major part of the human lungs is such fine structured that nowadays available CT scanners cannot capture it. Furthermore, even if the geometry of a complete lungs could be reconstructed, it would be impossible to simulate the complete system due to associated enormous computational costs. Thus, adequate modelling is required to realistically map the underlying characteristics.

Various models have been proposed and applied in recent years. Kleinstreuer and Zhang provide a detailed overview in [**81**]). There, it is stated that the considered research topics concern mostly the effect of geometry changes during inhalation whereby the corresponding experiments and numerical simulations focus mainly on specific parts of the lungs. Yet, coupled approaches have already been proposed for the human lungs. For example, Baffico et al. [**10**] introduces, analyses and applies a multi-scale model for the human lungs (cf. Subsection 6.3.2). In recent years, the computational studies seem to be dominated by a discussion about the adequateness of different turbulent models (cf. e.g. [**11, 80, 92, 95**] and references therein). Besides many approaches which employ modelled bronchial trees as computational domain, one also finds some which employ realistic geometries obtained by e.g. CT or MRI scanners. Li et al. [**92**], Baffico et al. [**10**] and Fetita et al. [**37**] consider up to the first six generations[1] of a bronchial tree. In the latter two approaches, the underlying geometry is reconstructed applying a *marching cube-based adaptive*

[1]An approach for a numbering of generations in a bronchial tree is provided by Weibel in [**135, 136**].

approach as proposed by Fetita et al. [**37**] which leads to a relative accurate reconstruction in comparison with other approaches based on e.g. region growing methods. Further to be mentioned is the approach of Freitas and Schröder [**41**] where an LBM is applied to simulate the airflow in the upper part of the human lungs. The underlying geometry is based on a model comprising the first six generations.

The goal of this section is to study the feasibility of a two-scale approach for the simulation of an expiration at a fixed flow rate of $Fl = 150$ ml/s in the human lungs. The upper part of the human lungs which can be reconstructed from CT scans serves as computational domain for an LB simulation. The rest of the lungs' geometry is modelled. Its functionality is described by another model. The two parts are coupled by means of boundary conditions.

The first part of this section focuses on introducing the relevant professional terminology concerning the anatomy of the human lungs and generally accepted facts about its physiology. Subsection 6.3.2 is dedicated to describe the considered two-scale model. Based on this description, a particular flow problem in the upper human lungs is formulated and details concerning the LB-based dicretisation are stated. Finally, in Subsection 6.3.4 the numerically obtained simulation results are presented and discussed.

6.3.1. About the Anatomy and Physiology of the Human Lungs. In the following, a brief overview is given. The subsection aims to establish the terminology and physiology which is of interest for the remainder of this section. The explanation to come follows essentially standard literature [**50, 122, 135**].

Human lungs are arranged on both sides of the heart in two cavities. Correspondingly, they are called *right lung* and *left lung*. The right lung consists of three *lobes* and the left one of two lobes. They are separated by fissures. According to their attributed functionally, the lungs can also be separated into two parts, namely the *respiratory zone* and the *conducting zone*. The respiratory zone consists amongst others of *respiratory bronchioles* and *alveoli* where the gas exchange actually occurs. The conducting zone provides a connection for the air to flow from the upper respiratory system, i.e. larynx and nose, to the respiratory zone. This airway system is shaped like the root of a tree. The *trachea* can be seen as the stem of the thought tree. It is a tube with an inner diameter of about 2 cm and a length of about 12 cm. The trachea starts at the upper respiratory system. Then, it splits into two airways which are called the *main bronchi*. Each bronchi continues to divide multiple times leading to smaller and smaller tube-shaped airways which subdivide as well and which are called *bronchioles*. Finally, they end in the *alveolar sacs* which are clusters of alveoli.

As indicated above, the main functionalities provided by the human lungs is the transport, dispersion and exchange of gas. Typically, obtained ventilation rates and frequencies are given in Table 6.1 for different levels of physical activity for an adult man. As well for an adult male, the total capacity of the lungs is found to be about 6 litre. According to Weibel [**136**], approximately 10 per cent of it equates to the volume that corresponds to the conducting zone. The same author [**135, 136**] employs a numbering common for *binary trees* for that of the bronchi and bronchioles. The result is a *bronchial tree* with about $m = 23$ *generations* leading to a number of about up to 2^m bronchi and bronchioles. Thereby, about the last

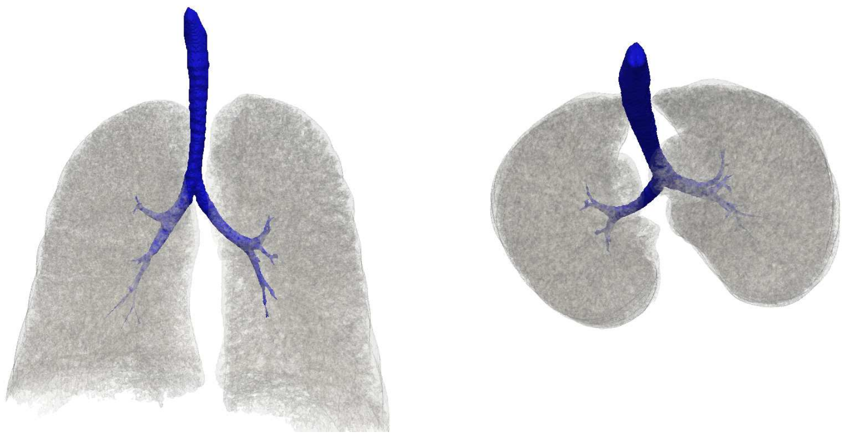


FIGURE 6.14. The human lungs shown from the front (left) and from above (right). Visualised are the segmented CT data area (grey) of the considered human lungs and the surface (blue) of that part of the upper human lungs which could be clearly extracted from the segmented CT data. The surface mesh consists of $17,940$ triangles which cover an area of about 176 cm^2 and give room for approximately 51.86 ml of air. The images are obtained by means of joint work with Gengenbach in the framework of the United Airways project [**15**].

6 generations consists of respiratory bronchioles which are part of the respiratory zone.

Beside the respiratory function of the human lungs, in the literature, e.g. in [**50, 122, 135**], one frequently finds that different parts of lungs provide other functionalities, like for example that breathed air is warmed, cleaned and humidified in the lungs.

The parts of the lungs which belong to the respiratory zone are of small scales, e.g. the diameter of an alveoli is about 50-250 μm. Thus, it is impossible to capture these fine structures by nowadays CT scanners which are used in everyday medical practice. But also the conducting zone of the lungs cannot be captured completely. Both parts of the considered human lungs, namely the resolvable and non-resolvable, are pictured in Figure 6.14. Not least due the complexity of the geometry of the human lungs, adequate models are required to describe and finally numerically simulate full respiration.

6.3.2. A Two-scale Model for the Respiration in the Human Lungs. The basic idea of the proposed two-scale model is to couple two individual models, one describing the flow in the upper, resolvable part of the human lungs and another one describing the flow and the actual gas exchange. The coupling is realised via an adequate setting of boundary conditions at the links which are given by the ends of the resolved bronchi tree. In Figure 6.15 the idea of the two-scale model is illustrated.

For the upper lungs model it is assumed that the fluid is incompressible and Newtonian. Thus, according to Chapter 1 the incompressible Navier-Strokes eqation as well as the BGK-Boltzmann equation are suitable governing equations. The latter mesoscopic model approach is considered in the following. Then, as condition

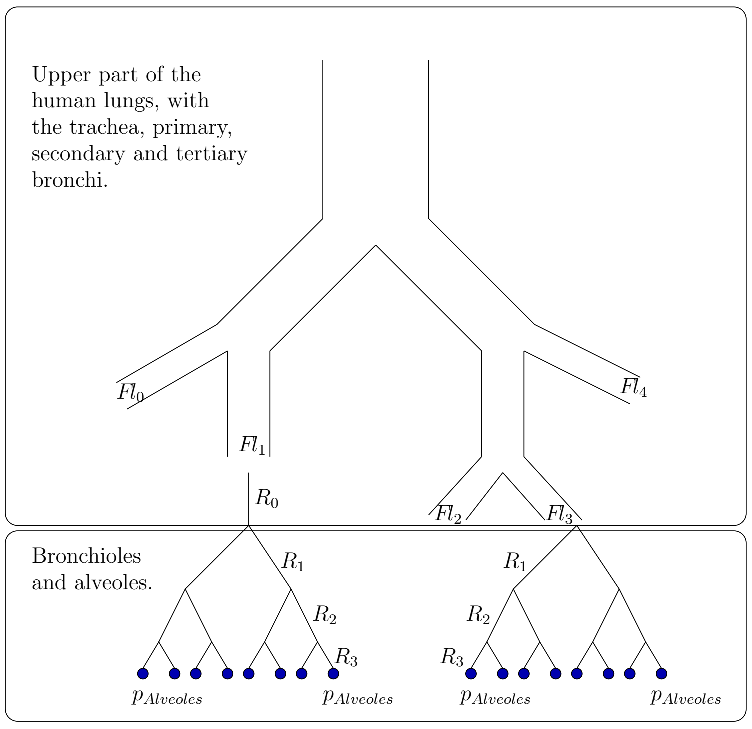


FIGURE 6.15. Scheme of the coupling between the models for the upper and the lower lungs. The image is obtained by means of joint work with Gengenbach in the framework of the United Airways project [**46**].

for the boundary located at the trachea a pressure condition is set. The conditions for the boundaries at the considered ends of the resolved bronchioles are set to be inlet or outlet velocity conditions. Whereby, the velocity is distributed as in a pipe aligned in normal direction to the boundary where Poiseuille's law holds. At each bronchiole the maximum velocity magnitude or alternatively the flow rate Fl, which both characterises the distribution, is assumed to be given by the model for the lower lungs. In turn, the upper lungs model provides, in particular the pressure distribution for the boundaries at all bronchi which serves as input for the lower lungs model. In this context to be mentioned is a similar model introduced by Baffico et al. [**10**]. It basically differs in the kind of boundary conditions for the upper model at any bronchiole's end. Instead of a velocity condition, it is proposed to set a pressure condition. The employed model for the lower lungs takes the velocity profiles at the bronchioles' ends boundaries as input.

Weibel's model for the geometry of the bronchial tree (cf. Subsection 6.3.1) provides the basis for many other models aiming to describe the respiration in the human lungs, which have been proposed in recent years. For example, Grandmont et al. [**51**] propose to model the breathing functionality of the lungs by a spring-mass system with dissipation. The bronchi tree is assumed to be a symmetrical dyadic tree of pipes while the flow in it obeys Poiseuille's law. At the outlets of the dyadic tree the flow is delimited by two successive masses which model the alveolar sacs. Yet, in the presented two-scale approach such a bronchial tree model is just employed to model the non-resolvable bronchioles of the bronchial tree. Thereto, at the end of any resolved bronchus a bronchial tree model like that of Grandmont et al. [**51**] is employed whereby its input data depends on the pressure distribution of the respective bronchus. A similar model is proposed by Gengenbach [**48**]. It takes besides the pressure distribution also the generation and the diameter of any bronchus, respectively bronchiole which represents the root of the respective tree, into account. This model relies, similarly to the one researched by Grandmont et al. [**51**], on the assumption that the non-resolvable parts of the bronchial tree can be described by a symmetrical dyadic tree of pipes whereby for the fluid flow Poiseuille's law holds.

6.3.3. Problem Formulation and Discretisation Issues. In the previous subsection a two-scale model describing the respiration in the human lungs is introduced. Thereby, it is suggested to model the flow in the resolvable, upper part of the human lungs by means of the BGK-Boltzmann equation. The aim of this subsection is to illustrate the feasibility of the sub-model for the upper lungs. Thereunto, in the following, a prototypical fluid flow problem is formulated and the applied numerical methods in order to solve it are stated.

An expiration for a realistic everyday situation is to be simulated. Thereby, the flow rate is fixed to be $Fl = 150$ ml/s which is typically observed for adult males in situations of resting or sitting awake (cf. Table 6.1). The air is considered at normal conditions, i.e. $1,013$ hPa and $T = 20\,^{\circ}$C. Thus, its speed of sound is $c_s = 343$ m/s, its dencity is $\rho = 1.225$ kg/m^3 and its kinematic viscosity is $\nu = 1.4 \cdot 10^{-5}$ m^2/s. Since the respiration is naturally a transient process, it is not clear if a unique solution for the considered configuration exists. However, encouraged by the numerical results presented in Subsection 6.2.3 for an intranasal flow for a fixed flow rate of $Fl = 150$ ml/s, it is assumed that such a *pseudo steady state* exists.

The problem is solved numerically by applying a $D3Q19$ LBM as it is derived in Section 2.1. The pressure and velocity boundary conditions are realised as proposed by Skordos [**124**]. A no-slip condition is set for boundaries representing walls. It is realised by the bounce back rule according to its specification in Section 2.1.4.

The underlying discrete geometry Ω_h for the considered problem is that part of the lungs which could be reconstructed from CT scans according to the procedure decribed in Section 6.1. The person whose lungs have been scanned is the same 46 years old European male who is considered in Section 6.2 for the simulation of intranasal flows. According to the refinement strategy introduced in Subsection 6.1.3 one obtains the discretisation parameter of $h = 0.23$ mm$/N$ for a refinement level $N \in \mathbb{N}$.

To distinguish different boundary regions Γ_h^i material numbers i are introduced as stated in Subsection 6.1.2. Whereby, $i = 2$ denotes the boundary which represents the wall of the bronchial tree, $i = 3$ the boundary at the trachea and $100m+j$ one of the 15 inlets at the considered bronchioles with m representing the generation of the actual bronchiole according to the numbering of Weibel [**135, 136**] and with $j = 1, 2, ...$ serving as counter to distinguish the bronchioles of the same generation. The so obtained material numbers are condensed in the set $B := \{401, 402, 403, 404, 501, 502, ..., 507, 601, 701, 801, 901\}$. In Figure 6.7 the resulting geometry representation of the computational domain Γ_h^i is depicted.

According to the model description for the upper lungs stated in the previous section, a velocity condition is set at the 15 inlets at the bronchioles' ends, and a pressure condition is applied at the boundary at the trachea Γ_h^3. Furthermore, it is stated there, that at all inflow boundaries Γ_h^i $(i \in B)$ at the bronchioles the maximum velocity magnitude U^i_{max} or a flow rate is assumed to be given by the bronchial tree model. Based on that information and Poiseuille's law, the velocity distribution at every single boundary area is computed. In the presented feasibility study for the upper lung sub-model, the maximum velocity magnitudes are assumed to be the same for all 15 inflows, i.e. $U_{max} = U^i_{max}$ for all $i \in B$. Thence, different flow rates are obtained at different bronchioles' ends since the corresponding boundary areas are of different sizes. However, with the given wanted total flow rate of $\mathit{Fl} = 150$ ml/s the maximum velocity magnitude, namely can be computed. It is found to be approximately $U_{max} \approx 5.28$ m/s.

The area of the boundary Γ^3_h at the trachea is found to be approximately $A \approx 0.9065$ cm^2. To reach the desired flow rate of $\mathit{Fl} = 150$ ml/s a mean speed of $U_{mean}(\mathit{Fl}) \approx 1.65$ m/s is needed. This mean speed is considered to be the characteristic macroscopic velocity. The characteristic macroscopic length is fixed to

$$L := \max_{\boldsymbol{r}\in\Gamma^3_h} r_1 - \min_{\boldsymbol{r}\in\Gamma^3_h} r_1 = 0.015 \text{ m} .$$

With it, the kinematic viscosity $\nu = 1.4 \cdot 10^{-5}$ m^2/s and speed of sound $c_s = 343$ m/s, one obtains the Reynold number $\mathit{Re} \approx 1768$ and the Mach number $\mathit{Ma} \approx 0.0048$.

The number of time steps needed to reach a steady state can be reduced by introducing lattice speeds $U_{LB} = 0.01, 0.005, 0.002$ and scaling the problem according to (6.2). This leads to a problem with the same Reynolds number $\widetilde{\mathit{Re}} = \mathit{Re}$ but a different Mach number $\widetilde{\mathit{Ma}} = U_{LB} U_{mean}$. For simplification the as constant assumed density $\rho = 1.225$ kg/m^3 is scaled to $\widetilde{\rho} = 1$. The obtained and in the following subsection presented results are rescaled to the original system of units.

The initial distribution $f_i(t_0)$ is set to be Maxwellian distributed with $\boldsymbol{u} = 0$ and $\widetilde{\rho} = 1$ in $\Omega_h \times Q$. In order to avoid unnatural behaviour as well as numerical artefacts caused by e.g. violation of smoothness, the maximum velocity magnitude is increased linearly in the first $100,000$ time steps from $U_{max} = 0$ to $U_{max} = 5.28$ m/s.

6.3.4. Presentation and Discussion of the Numerical Results. The presentation and discussion of the numerical results obtained for the simulation of an expiration at a fixed flow rate of $\mathit{Fl} = 150$ ml/s in the upper human lungs are the subjects of this subsection. The simulation set-up is described in the last two subsections, more precisely the underlying physical two-scale model in Subsection 6.3.2 and the numerical methods applied in Subsection 6.3.3.

As the numerical experiments for the simulation of intranasal flows, the ones discussed here are executed in parallel either on the *HP XC4000* or on the *JUROPA* high performance computer both described in Section 3.3. It is always taken advantage of the full hybrid parallelisation approach presented in Chapter 3, which is realised using OpenMP and MPI. For example, employing in total $1,024$ cores on 256 nodes on the *HP XC4000*, the execution of the code associated with a problem with $N = 2$ and $U_{LB} = 0.005$ takes about 46 hours and 17 minutes for $1,325,200$ time steps. This problem configuration is typical for the presented results. The referred execution time also includes the needed times to set up the simulation and to write $2,650$ output files. In this context, the discussion of the performance results obtained for similar problem configurations which is presented in Subsection 3.4.2

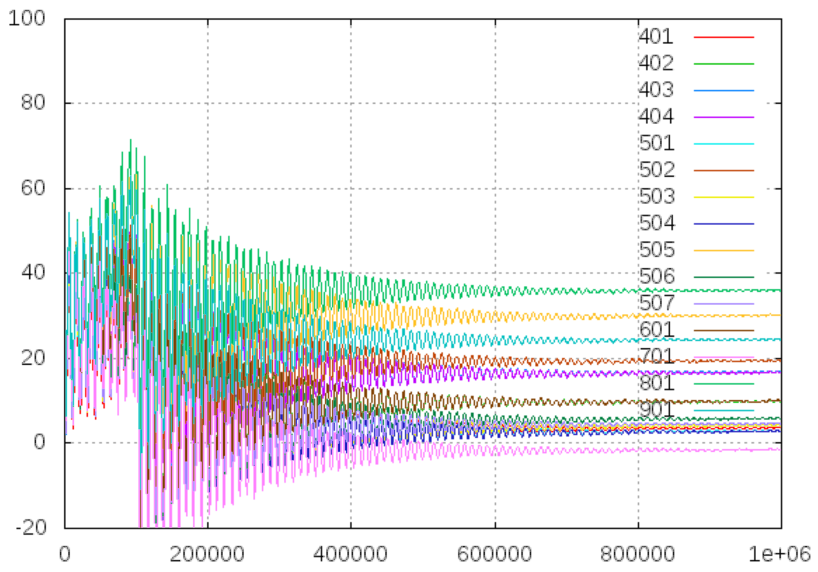


FIGURE 6.16. Time evolution of the total pressure drops $p_{3,i}$ ($i \in B$) at an expiration flow rate of $Fl = 150$ ml/s towards a pseudo steady state. The lattice speed is chosen to be $U_{LB} = 0.005$, and the refinement level is set to $N = 2$.

is to be noted.

The relevant quantity for the coupling of the two models is the obtained pressure at the bronchioles' ends Γ_h^i ($i \in B$). The simulated pressures are captured by the variable p_i ($i \in B \cup \{3\}$) and taken close to the position in the centre of the area Γ_h^i, namely at

$$\boldsymbol{r}^i_{\text{mid}} := \frac{1}{2}\begin{pmatrix} \max_{\boldsymbol{r}\in\Gamma_h^i} r_1 + \min_{\boldsymbol{r}\in\Gamma_h^i} r_1 \\ \max_{\boldsymbol{r}\in\Gamma_h^i} r_2 + \min_{\boldsymbol{r}\in\Gamma_h^i} r_2 \\ \max_{\boldsymbol{r}\in\Gamma_h^i} r_3 + \min_{\boldsymbol{r}\in\Gamma_h^i} r_3 \end{pmatrix} - 2hN\boldsymbol{n} \qquad (i \in B \cup \{3\}) \ .$$

With it, the pressure drops can be computed according to

$$p_{3,i} := p_i - p_3 \qquad (i \in B) \ .$$

It is to be noted, that $p_3 \approx 1{,}013$ hPa due to the boundary condition applied at the boundary which belongs to the outflow area Γ_h^3 at the trachea.

At first, the asymptotic towards a pseudo steady state is studied. Thereto, the evolution of the simulated pressure drops $p_{3,i}$ ($i \in B$) are monitored over a certain number of time steps for different refinement levels $N = 2, 3$ and lattice speeds $U_{LB} = 0.01, 0.005, 0.002$. Exemplary the case which is characterised by $N = 2$ and $U_{LB} = 0.005$ is studied. For all other considered cases a similar behaviour is observed. In Figure 6.16 for all $i \in B$ the corresponding computed mean pressure drops $p_{3,i}$ are plotted as functions of time steps.

For the computed pressure drops p_i ($i \in B$), the convergence characteristics towards pseudo steady states is observed to be similar to those obtained for the simulations of expirations through the human nose, studied in Subsection 6.2.3. Taking the smaller lattice speeds into account which lead to smaller physical times per time step, the decay of the oscillations is also in accordance to the other mentioned results. Yet, they indicate that the start-up phase of $100{,}000$ time steps is

material	$N = 2$		$N = 3$		
number i	$U_{LB} = 0.005$	$U_{LB} = 0.002$	$U_{LB} = 0.01$	$U_{LB} = 0.005$	$U_{LB} = 0.002$
401	3.6	10.9	3.4	3.3	1.3
402	9.9	9.0	5.7	6.9	7.1
403	16.9	23.8	11.0	12.9	17.5
404	16.7	18.2	12.8	12.1	10.1
501	2.9	4.1	4.2	6.4	9.1
502	19.5	32.3	6.4	6.4	9.5
503	4.3	5.0	2.1	2.3	3.7
504	2.9	2.7	3.7	3.5	3.7
505	30.2	23.1	26.6	25.5	29.9
506	5.9	9.7	4.8	5.0	6.5
507	4.6	4.2	3.8	4.1	4.8
601	9.9	16.4	6.7	9.5	20.1
701	-1.5	-8.1	11.4	23.8	57.5
801	36.0	51.3	11.7	14.2	20.4
901	24.4	32.8	24.1	37.0	55.6

TABLE 6.6. Obtained results for the mean pressure drops $\overline{p}_i$ (Pa) for different choices of lattice speeds U_{LB} and refinement levels N.

not sufficient. However, in consideration of the high computational costs a longer start-up phase is hardly feasibly.

In order to obtain well defined results the averaging strategy proposed in Subsection 6.2.3 is similarly applied here. In Figure 6.6 the corresponding computed mean pressure drops $\overline{p}_i$ $(i \in B)$ are stated for differently chosen refinement levels N and lattice speeds U_{LB}.

The obtained results are not all found to be independent of N and U_{LB}. Especially in the bronchioles of higher generations, i.e. $m = 6, 7, 8, 9$, relative deviations of more than 100 per cent are observed. In turn, in other bronchioles the differences are comparatively small. By trend, this is the case when the simulated pressure drops are rather small. Then coupling the two models of the two-scale approach, it is expected that due to the character of the model for the lower parts of the lungs great differences of the pressure drops dissipate. In this respect, a full coupling of the two models seems possible.

In Figure 6.17 the simulated flow field is displayed in form of the velocity magnitude in chosen cross-sections. The underlying refinement level is set to $N = 2$ and the lattice speed to $U_{LB} = 0.005$. In the same picture, a closer look to four planes located around the first bifurcation of the bronchial tree clearly reveals the complex characteristics of the flow field. There, the obtained values for the velocity magnitude are found to be about 2.6 m/s which is between the mean speed $U_{mean} \approx 1.65$ m/s at the outlet at the trachea and the maximum speed $U_{max} = 5.28$ m/s at the inlet at the bronchioles' ends. Relative high velocity magnitudes of up to about 9.07 m/s are observed in the bronchiole which ends in Γ_h^{901}.

Summing up, the simulated pressure drops are found to reach pseudo steady states for several chosen discretisation parameters, i.e. lattice speeds U_{LB} and refinement levels N. However, not all results are observed to approach values independent of U_{LB} and N. Yet, due to the dissipative property of the model for the lower part of the lungs, a realisation of the two-scale approach seems practicable.

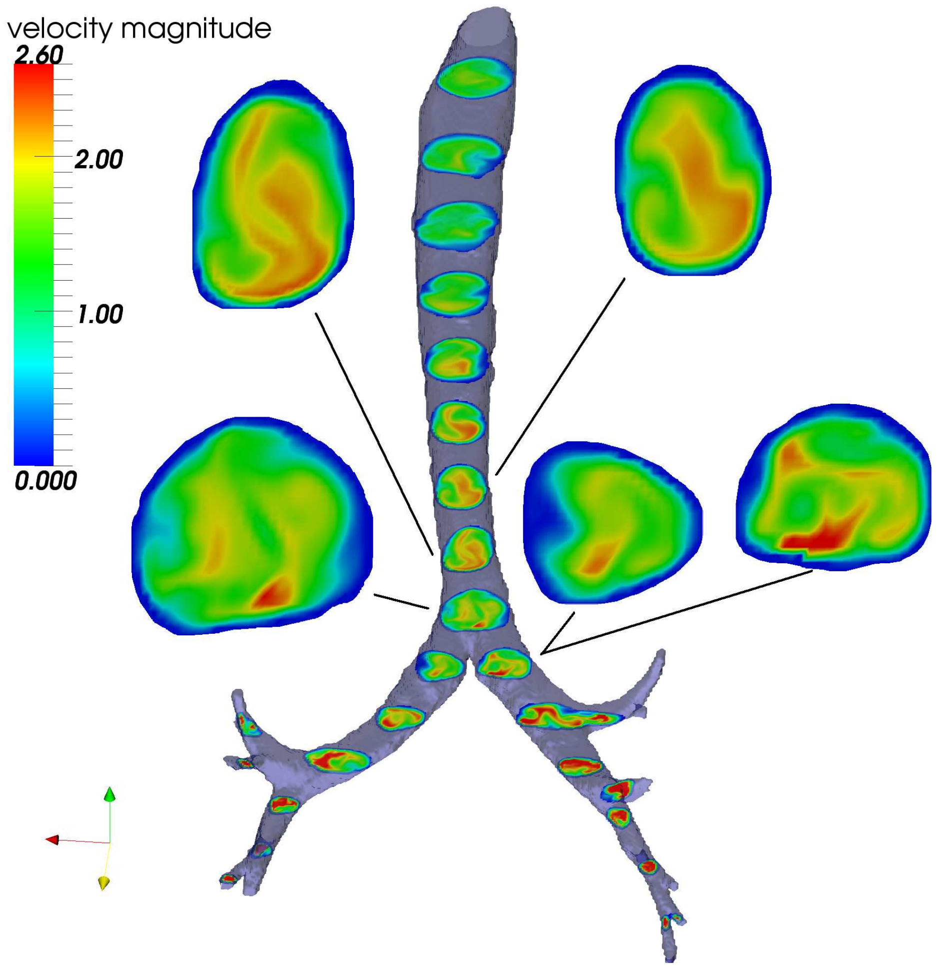


FIGURE 6.17. The simulated flow field for an expiration flow rate of $Fl = 150$ ml/s in the human lungs is shown by mean of various planes. The intersections are coloured according to the computed velocity magnitude which is provided in meter per second. A closer look is provided to four planes around the first bifurcation of the bronchial tree. There, the complexity of the flow dynamics clearly becomes visible. The lattice speed in the LB simulation is set to be $U_{LB} = 0.005$ and the refinement level to $N = 2$. The simulation is stopped after $800,000$ times steps has been performed.

In future works, a better resolution of the extracted geometry and the adoption of other boundary conditions, like the one proposed by Bouzidi et al. [**22**], should also be considered in order to obtain a better boundary approximation and with it potentially more stability. Furthermore, the relative high computational costs might be reduced.

Summary and Outlook

In this last chapter, the content of this thesis as well as the obtained results are summerised and open questions arising in the context of the presented work are addressed chapter by chapter. Then, at the end, an integrative perspective is provided which concludes the work.

In Chapter 1, two physical models which both describe the dynamics of incompressible Newtonian fluids are introduced and compared to each other. The first is a macroscopic model governed by the incompressible Navier-Stokes equation and the second is a mesoscopic model governed by the Boltzmann equation or its simplification, the BGK-Boltzmann equation. The stochastic origin of the mesoscopic model is emphasised by deriving it in an appropriate framework. Taking advantage of stochastic methods, seems to be promising with respect to enable a deeper analysis of the corresponding governing equations. This might help answer the open questions concerning e.g. solvability of the Boltzmann equation, its connection to the incompressible Navier-Stokes equation, and in turn its solvability.

In the following chapter, LBM are introduced as discretisation techniques for families of BGK-Boltzmann equations. A framework is established which enables the derivation of families of consistent LB equations from certain families of BGK-Boltzmann equations. An interesting unanswered question is whether the stability of the considered LB schemes can also be shown by taking advantage of the provided framework. Further, a complete analysis of adequate boundary conditions remains to be provided. The promising results obtained for the two examples considered at the end of Chapter 2 encourage further investigations in these directions.

A hybrid parallelisation concept especially developed for LBM is the subject of Chapter 3. The strategy and its realisation, which allows coping with platforms sharing the properties of both shared and distributed architectures, is illustrated in detail. The performance results for two examples are presented and discussed. The MPI-based approach is found to be efficient for both problems with underlying simple and complex geometries since the computing time can almost be halved even if a relatively large number of nodes employed, e.g. 128, is doubled. It is expected that applying a sophisticated graph-based partitioning algorithm leads to a further improvement of the efficiency. This should be taken into account in future work. Further, it is revealed that the shared memory approach that uses OpenMP cannot compete against the MPI version on SMP nodes. To achieve competitive performance results the memory hierarchy and binding of threads to specific processors on platforms with NUMA have to be taken into account. Since OpenMP does not provide any support for memory hierarchies and affinity control, the programmer is left to use utilities provided by the operating environment which makes the software more complex and less portable. The motivation of the proposed hybrid parallelisation concept is based on the idea that this kind of local partition at the nodal level should automatically be treated by the considered parallelisation paradigm.

Since OpenMP is actually intended to offer such an interface, the results clearly reveal the need of optimising the considered implementation of OpenMP.

Chapter 4 and Chapter 5 introduce strategies to numerically solve fluid flow control and optimisation problems, where the side conditions are governed by a BGK-Boltzmann equation. These strategies are then realiesed and finally applied to certain test cases. Both approaches are suitable for parallel computations and take advantage of the hybrid concept introduced in Chapter 3. Although they are both gradient-based and hence require the computation of certain derivatives, the two approaches differ in the way that the derivatives are obtained.

In the first of the two chapters, the application of a *first-discretise-then-optimise* strategy based on an AD forward mode technique is illustrated. Then, a generic parallel implementation is presented and its realisation is discussed. The obtained numerical results for an example are found to verify the approach. The corresponding performance results reveal a roughly linear growth of the time needed to perform one optimisation step with respect to the dimension of the control space multiplied with the number of unknowns of the underlying discrete fluid flow problem. For the parallel case, this growth rate is observed to decrease which attests the scalability of the approach. Whether an AD backward mode scheme can be realised in order to solve realistic 3D problems more efficiently or not, remains an open question. Difficulties occurring in this context and possible strategies to overcome them are also discussed in this chapter.

The *first-optimise-then-discretise* ansatz is followed in Chapter 5. The approach is based on obtaining the required derivatives by means of solutions of adjoint problems. A first order optimality system as necessary condition for an optimum of the continuous optimisation problem is derived by applying Lagrange's formalism. Establishing a sufficient condition for an optimum and, furthermore, proving its solvability need to be tackled in future works. The *adjoint BGK-Boltzmann equation* is then derived as the governing equation of a prototypical adjoint problem. Afterwards, methods similar to LBM are introduced and applied to discretise equations of this type. This leads to *adjoint lattice Boltzmann equations* whose structure is similar to those of LB equations. Aspects concerning the parallel implementation are also addressed. At the end of the chapter, the distributed control problem formulated in Chapter 4 is considered as a test case. The obtained results are found to numerically verify the realisation. Interesting to note is the convergence behaviour towards a steady state, which indicates an underlying physical meaning that might be founded in a relation between a family of adjoint BGK-Boltzmann equations and an adjoint incompressible Navier-Stokes equation. This encourages further investigations aiming to verify the relation. A test of the parallel performance reveals a similar scalability as for the LB equations tested in Chapter 3. Finally, both the numerical and the performance results are compared to those obtained by applying the AD-based strategy. The computational costs for solving by an ALB scheme are observed to grow proportionally to the number of unknowns of the fluid flow problem and to be independent to the number of control variables. Although the costs are found to be smaller than those measured for the AD-based approach, the easier implementation of an AD-based strategy should not be ignored in practice.

In Chapter 6, the numerical results obtained for simulations of the respiration in an individual human nose as well as in the upper part of a human lungs are presented. The underlying geometry data are obtained by computer tomography (CT) scans of a patient with a diagnosed ventilation disorder. A preprocessing concept, that is to a large extend automated is first presented. This procedure is used to transform the raw data to a form suitable for the simulation of fluid flows in complex geometries with an LB algorithm. The realisation of the strategies is tested by extracting the computational domains of the lungs and nose from the mentioned CT data.

The flow of air in the human nose is simulated for two test configurations both corresponding to realistic everyday situations, i.e. with flow rates of $100 - 333$ ml/s. The first of them is an exhalation through both nostrils at several fixed ventilation rates. Numerical evidence is established for the simulated values for total pressure drops occurring between the nostrils and the nasopharynx. Firstly, the simulation is found to approach pseudo steady states independently of the discretisation parameter and the grid resolution. Secondly, the results are validated by means of a comparison with numerical results as well as experimentally data both obtained by others for similar geometries. Considering the computed flow field, a stenosis at the left inferior meatus is found to be the possible cause for the ventilation disorder. The results encourage further research in this direction. Both inspirations and expirations in geometries of other patients could be simulated with the aim of discovering similar pathologies related in the geometry of the nose. The second test case considers respirations through only one nostril at a time. Comparing the numerical results with measurements from the very patient obtained by rhinomanometry methods, quantitative agreement is observed. In the future, the same test should be repeated based on a wider measurement basis so that also a quantitative agreement can be achieved.

Finally, a feasibility study for the numerical simulation of respirations in the complete human lungs is presented. As part of a two-scale approach, the considered problem is an exhalation at a fixed flow rate of 150 ml/s in the upper part of the human lungs. The computed pressure drops between an upper trachea part and several bronchioles of up to the 9th generation are observed to reach a steady state for different considered discretisation parameters. A full coupling of the two sub-models of the two-scale approach seems practicable and should be taken into account in future works. A finer resolution of the extracted geometry and the adoption of other boundary conditions like the one proposed by Bouzidi et al. [**22**], should also be considered in order to obtain a better boundary approximation and with it potentially more stability. Furthermore, the high computational costs might be reduced.

Last but not least, it seems promising to consider the application of the optimisation strategies proposed in Chapter 4 and Chapter 5 in future works to learn more about human respiratory flows. Firstly, the range of feasible applications widens, encompassing for example patient-individual in advance surgery planning or optimising the medication via asthma sprays. Secondly, these strategies make it possible to establish other techniques like adaptive mesh refinement strategies or optimal experimental design approaches. This, in turn, leads to lower computational costs and more accurate results on the one hand and widens the possible range of applications even further on the other. Thus, completely new aspects related to the *grand challenge* of being able to numerically simulate the full human respiratory system can be tackled in the future.

Bibliography

[1] 3D Systems, Inc., 333 Three D Systems Circle Rock Hill, SC 29730 USA. *Stereolithography Interface Specification*, October 1989.

[2] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6):641–647, 1994.

[3] Mark Ainsworth and John Tinsley Oden. *A posteriori error estimation in finite element analysis*. Pure and applied mathematics, A Wiley-Interscience publication. Wiley, New York, 2000.

[4] Altair Engineering Inc., 1820 E. Big Beaver Rd., Troy, MI 48083-2031 USA. *Altair HyperMesh: The Fastest, Solver Neutral CAE Environment for High Fidelity Modeling*, April 2010.

[5] Santosh Ansumali. *Minimal kinetic modeling of hydrodynamics*. PhD thesis, Technische Wissenschaften, Eidgenssische Technische Hochschule ETH Zrich, 2004.

[6] Pierre Aubert, Nicolas Di Césaré, and Olivier Pironneau. Automatic differentiation in C++ using expression templates and application to a flow control problem. *Computing and Visualization in Science*, 3:197–208, 2001.

[7] Lisa S. Avila, editor. *The VTK users guide : updated for VTK version 5*. Kitware, [New York], 2006. CD-ROM u.d.T.: The visualization toolkit, Version 5.0.4.

[8] Hans Babovsky. *Die Boltzmann-Gleichung*. Teubner Verlag, 2002.

[9] Ivo [Editor] Babuka. Modeling, mesh generation, and adaptive numerical methods for partial differential equations. Number 75 in The IMA volumes in mathematics and its applications, New York, 1995. Springer.

[10] Léonardo Baffico, Celine Grandmont, and Bertrand Maury. Multiscale modelling of the respiratory tract. Research Report, 2008.

[11] C.G. Ball, M. Uddin, and A. Pollard. High resolution turbulence modelling of airflow in an idealised human extra-thoracic airway. *Computers & Fluids*, 37(8):943 – 964, 2008.

[12] C. Bardos, F. Golse, and D. Levermore. Fluid dynamic limits of kinetic equations. i. formal derivations. *Journal of Statistical Physics*, 63:323–344, April 1991.

[13] G.K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1967.

[14] M. Baumann. Numerische Methoden zur Visualisierung instationärer Strmungsvorgänge. Diplomarbeit, Universität Karlsruhe (TH), Fakultät für Mathematik, 2007.

[15] M. Baumann, T. Gengenbach, W. Heppt, V. Heuveline, M.J. Krause, and S. Zimny. United Airways: Numerical Simulation of the Human Respiratory System, 2009. http://www.united-airways.eu.

[16] M. Baumann, E. Giotakis, W. Heppt, V. Heuveline, M.J. Krause, R. Mayer, and P. Weber. United Airways: Numerical Simulation of the Human Respiratory System, 2008. http://www.united-airways.eu.

[17] Fokko Beekhof. CVMLCPP: Common Versatile Multi-purpose Library for C++, Web page. online, April 2010. http://tech.unige.ch/cvmlcpp.

[18] Claus Bendtsen and Ole Stauning. FADBAD, a flexible C++ package for automatic differentiation. Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1996.

[19] P. L. Bhatnagar, E. P. Gross, and M. Krook. A Model for Collision Processes in Gases. 1. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Phys. Rev.*, 94:511–525, 1954.

[20] S. Blazy, S. Nazarov, and M. Specovius-Neugebauer. Artificial Boundary Conditions of Pressure Type for Viscous Flows in a System of Pipes. *Journal of Mathematical Fluid Mechanics*, 9:1–33, 2007.

[21] Hendryk Bockelmann and Vincent Heuveline. Parallel I/O and Checkpointing Strategies for PDE-constrained Optimization. In *accepted for proceedings of PARA 2008, 9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*, 2008.

[22] M'hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13(11):3452–3459, 2001.

[23] J. M. Buick and C. A. Greated. Gravity in a lattice Boltzmann model. *Phys. Rev. E*, 61(5):5307–5320, May 2000.

[24] J.M. Bull. Measuring synchronisation and scheduling overheads in OpenMP. In *European Workshop on OpenMP (EWOMP1999), Lund, Sweden*, 1999.

[25] Carlo Cercignani. *The Boltzmann Equation and its Applications.* Springer-Verlag, New York, 1988.

[26] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald. *Parallel Programming in OpenMP.* Morgan Kaufmann, 2001.

[27] T. P. Chiang, W.H. Sheu, and Robert R. Hwang. Effect of Reynolds number on the eddy structure in a lid-driven cavity. *International Journal for Numerical Methods in Fluids*, 26(5):557–579, 1998.

[28] B. Chopard and M. Droz. *Cellular automata modeling of physical systems.* Cambridge University Press, 1998.

[29] S.E. Churchill, L.L. Shackelford, J.N. Georgi, and M.T. Black. Morphological variation and airflow dynamics in the human nose. *Am. J. Hum. Biol.*, 16:625–638, 2004.

[30] Eugene H. Courtiss and Robert M. Goldwyn. The Effects of Nasal Surgery on Airflow. *Plastic and Reconstructive Surgery*, 72(1):9–19, 1983.

[31] Dominique d'Humieres, Irina Ginzburg, Manfred Krafczyk, Pierre Lallemand, and Li-Shi Luo. Multiple-Relaxation-Time Lattice Boltzmann Models in Three Dimensions. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.

[32] A. Dupuis and B. Chopard. An object oriented approach to lattice gas modeling. *Future Generation Computer Systems*, 16:523–532, 2000.

[33] D. Elad, R. Liebenthal, B. L. Wenig, and S. Einav. Analysis of air flow patterns in the human nose. *Medical and Biological Engineering and Computing*, 31(6):585–592, 1993.

[34] Eduard Feireisl. *Dynamics of Viscous Compressible Fluids*, volume 26 of *Oxford Lecture Series in Mathematics and its Applications.* Oxford Press, 2003.

[35] Eduard Feireisl and Antonín Novotný. *Singular Limits in Thermodynamics of Viscous Fluids.* Advances in Mathematical Fluid Mechanics. Birkhäuser Basel, Basel, 2009. In: Springer-Online.

[36] Miloslav Feistauer. *Mathematical Methods in Fluid Dynamics.* Longman Scientific & Technical, Harlow, England, 1993.

[37] C. Fetita, S. Mancini, D. Perchet, F. Prteux, M. Thiriet, and L.. Vial. An image-based computational model of oscillatory flow in the proximal part of tracheobronchial trees. *Computer Methods in Biomechanics and Biomedical Engineering*, 8(8):279 – 293, 2005.

[38] M. Finck, D. Hänel, and I. Wlokas. Simulation of nasal flow by lattice boltzmann methods. *Comput. Biol. Med.*, 37(6):739–749, 2007.

[39] Rainhill K. Freitas. *Lattice Boltzmann methods for internal flows.* PhD thesis, Aachen Technische Hochschule, 2008.

[40] Rainhill K. Freitas and Wolfgang Schrder. Biofluidmechanik: Simulation der Luftströmung in den Atemwegen. *Deutsches Arzteblatt*, 105(27):A–1512, 2008.

[41] Rainhill K. Freitas and Wolfgang Schrder. Numerical investigation of the three-dimensional flow in a human lung model. *Journal of Biomechanics*, 41(11):2446 – 2457, 2008.

[42] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. *Phys. Rev. Lett.*, 56(14):1505–1508, Apr 1986.

[43] A. Fursikov, M. Gunzburger, L. S. Hou, and S. Manservisi. *Optimal Control Problems for the Navier–Stokes Equations*, pages 143–155. Lectures on Applied Mathematics. Springer, 2000.

[44] Carl Geiger and Christian Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben.* Springer-Lehrbuch. Springer, Berlin, 1999.

[45] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron. Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows. *Comp. Fluids*, (35):888–897, 2004.

[46] T. Gengenbach, T. Henn, W. Heppt, V. Heuveline, M.J. Krause, and S. Zimny. United Airways: Numerical Simulation of the Human Respiratory System, 2010. http://www.united-airways.eu.

[47] Thomas Gengenbach. Modellierung und numerische Simulation der menschlichen Atemwege auf Hochleistungsrechnern. Diplomarbeit, Universität Karlsruhe (TH), Fakultät für Mathematik, April 2009.

[48] Thomas Gengenbach, Mathias J. Krause, and Vincent Heuveline. Numerical Simulation of the Human Lung: A Two-scale Approach. In Olaf Dössel, editor, *accepted for Biomedical Engineering.* De Gruyter, 2010.

[49] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, December 1982.

[50] L. Gifford, R. Gosselink, H.-J. Haas, G. Heesen, M. Van Kampen, T. Reybrouck, H. Schewe, A. Schwab, H. Slater, M. Vieten, U. Wehrstein, and T. Weiß. *Angewandte Physiologie*, volume 2. Thieme, Stuttgart, 2005.

[51] C Grandmont, B Maury, and N Meunier. A viscoelastic model with non-local damping application to the human lungs. *Esaim-Mathematical Modelling And Numerical Analysis-Modelisation Mathematique Et Analyse Numerique*, 40(1):201–224, January 2006.

[52] Andreas Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1:35–54, 1992.

[53] Andreas Griewank. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Frontiers in applied mathematics; 19. SIAM, Philadelphia, 2000.

[54] Andreas Griewank, David Juedes, and Jean Utke. ADOL-C - A Package for the Automatic Differentiation of Algorithms Written in C/C++. *ACM Transactions on Mathematical Software*, 22(2):131167, 1993.

[55] J.-L. Guermond, C. Migeon, G. Pineau, and L. Quartapelle. Start-up flows in a three-dimensional rectangular driven cavity of aspect ration 1:1:2 at Re=1000. *Journal of Fluid Mechanics*, 450:169–199, 2002.

[56] Max D. Gunzburger. *Perspectives in flow control and optimization.* Advances in design and control ; 5. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.

[57] Z. Guo, C. Zheng, and B. Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Phys. Rev. E*, 65:046308, 2002.

[58] Günther Hämmerlin and Karl-Heinz Hoffmann. *Numerische Mathematik.* Springer-Lehrbuch Grundwissen Mathematik. Springer, Berlin, 4., nochmals durchges. aufl. edition, 1994.

[59] Dieter Hänel. *Molekulare Gasdynamik.* Springer, 2004.

[60] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100 – 132, 1985.

[61] J. Hardy, Y. Pomeau, and O. de Pazzis. Time evolution of a two-dimensional model system. i. invariant states and time correlation functions. *Journal of Mathematical Physics*, 14(12):1746–1759, 1973.

[62] X. He and L.-S. Luo. Lattice boltzmann model for the incompressible navier stokes equation. *Journal of Statistical Physics*, 88:927–944, 1997.

[63] Xiaoyi He and Li-Shi Luo. Theory of the lattice boltzmann method: From the boltzmann equation to the lattice boltzmann equation. *Phys. Rev. E*, 56(6):6811–6817, Dec 1997.

[64] Xiaoyi He, Qisu Zou, Li-Shi Luo, and Micah Dembo. Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model. *Journal of Statistical Physics*, 87(1):115–136, 1997.

[65] V. Heuveline and M.J. Krause. OpenLB: towards an efficient parallel open source library for lattice Boltzmann fluid flow simulations. volume 9 of *PARA, International Workshop on State-of-the-Art in Scientific and Parallel Computing*, 2010. accepted for.

[66] V. Heuveline, M.J. Krause, and J. Latt. Towards a hybrid parallelization of lattice Boltzmann methods. *Computers & Mathematics with Applications*, 2009.

[67] V. Heuveline, M.J. Krause, J. Latt, and O. Malaspinas. Open source lattice Boltzmann code (OpenLB). http://www.openlb.org.

[68] V. Heuveline and J. Latt. The OpenLB project: an open source and object oriented implementation of lattice Boltzmann methods. *Int. J. Mod. Phys. C*, 18:627–634, 2007.

[69] V. Heuveline and J.-P. Weiß. A parallel implementation of a lattice Boltzmann method on the ClearSpeed Advance Accelerator Board. *IWRMM-Preprints*, Nr. 07/03, 2007.

[70] M. Hinze. *Optimal and instantaneous control of the instationary Navier-Stokes equations.* Habilitation, Technische Universität Berlin, 2002.

[71] I. Hörschler, Ch. Brcker, W. Schrder, and M. Meinke. Investigation of the impact of the geometry on the nose flow. *European Journal of Mechanics - B/Fluids*, 25(4):471 – 490, 2006.

[72] E. H. Huizing and J. A. M. de Groot. *Functional reconstructive nasal surgery.* Georg Thieme Verlag, Stuttgart, 2003.

[73] Takaji Inamuro, Masato Yoshina, and Fumimaru Ogino. A non-slip boundary condition for lattice Boltzmann simulations. *Phys. Fluids*, 7:2928–2930, 1995.

[74] K. Inthavong, J. Wen, J. Tu, and Z. Tian. From CT scans to CFD modelling fluid and heat transfer in a realistic human nasal cavity. *Engineering Applications of Computational Fluid Mechanics*, 3(3):321–335, 2009.

[75] Michael Junk and Axel Klar. Discretizations for the incompressible navier-stokes equations based on the lattice boltzmann method. *SIAM J. Sci. Comput.*, 22(1):1–19, 2000.

[76] Michael Junk, Axel Klar, and Li-Shi Luo. Asymptotic analysis of the lattice boltzmann equation. *Journal of Computational Physics*, 210(2):676 – 704, 2005.

[77] J. T. Kelly, B. Asgharian, J. Kimbell, and B. Wong. Particle Deposition in Human Nasal Airway Replicas Manufactured by Different Methods. Part I: Inertial Regime Particles. *Aerosol Science and Technology*, 38(11):1063–1071, November 2004.

[78] J. T. Kelly, A. K. Prasad, and A. S. Wexler. Detailed flow patterns in the nasal cavity. *Journal of Applied Physiology*, 89(1):323–337, 2000.

[79] K. Keyhani, P. W. Scherer, and M. M. Mozell. Numerical simulation of airflow in the human nasal cavity. *Journal of Biomechanical Engineering*, 117(4):429–441, 1995.

[80] C. Kleinstreuer and Z. Zhang. Laminar-to-turbulent fluid-particle flows in a human airway model. *International Journal of Multiphase Flow*, 29(2):271 – 289, 2003.

[81] C. Kleinstreuer and Z. Zhang. Airflow and Particle Transport in the Human Respiratory System. *Annual Review of Fluid Mechanics*, 42(1):301, 2010.

[82] J. M. V. A. Koelman. A simple lattice boltzmann scheme for navier-stokes fluid flow. *EPL (Europhysics Letters)*, 15(6):603–607, 1991.

[83] C. Körner, T. Pohl, U. Rüde, N. Thürey, and T. Zeiser. Parallel lattice Boltzmann methods for CFD applications. In *Numerical Solution of Partial Differential Equations on Parallel Computers*, pages 439–465. 2006.

[84] M. Krafczyk and E. Rank. A parallelized lattice-gas solver for transient Navier-Stokes-flow: Implementation and simulation results. *Jour. Num. Meth. Eng.*, 38:1243–1258, 1995.

[85] J. Latt. How to implement your DdQq dynamics with only q variables per node (instead of 2q). Technical report, Tufts University Medford, USA, 2007.

[86] J. Latt. *OpenLB User Guide Associated to Release 0.4 of the Code*. OpenLB, 2008.

[87] J. Latt and B. Chopard. Vladymir – a C++ matrix library for data-parallel applications. *Future Generation Computer Systems*, 20:1023–1039, 2004.

[88] Jonas Latt. *Hydrodynamic limit of lattice Boltzmann equations*. PhD thesis, University of Geneva, Geneva, Switzerland, 2007.

[89] Jonas Latt and Bastien Chopard. Lattice Boltzmann method with regularized non-equilibrium distribution functions. *Math. Comp. Sim.*, 72:165–168, 2006.

[90] Jonas Latt, Bastien Chopard, Orestis Malaspinas, Michel Deville, and Andreas Michler. Straight velocity boundaries in the lattice Boltzmann method. *Phys. Rev. E*, 77:056703, 2008.

[91] Gunther Lehmann. Die Funktion der menschlichen Nase als Staubfilter. *European Journal of Applied Physiology and Occupational Physiology*, 7(2):167–175, 1933.

[92] J Lin, G L Hu, J R Fan, and D Pan. Study on airflow and inhaled particle deposition within realistic human upper respiratory tract. *Journal of Physics: Conference Series*, 147(1):012067, 2009.

[93] Jacques Louis Lions. *Optimal control of systems governed by partial differential equations*. Springer, Berlin, 1971.

[94] Pierre-Louis Lions. *Mathematical topics in fluid mechanics*, volume 2: Compressible models of *Oxford lecture series in mathematics and its applications ; 10*. Clarendon Press, 1998.

[95] X. Y. Luo, J. S. Hinton, T. T. Liew, and K. K. Tan. Les modelling of flow in a simple airway model. *Medical Engineering & Physics*, 26(5):403 – 413, 2004.

[96] F. Massaioli and G. Amati. Achieving high performance in a LBM code using OpenMP. EWOMP 2002, 2002.

[97] K. Mattila, J. Hyväluoma, T. Rossi, M. Aspnäs, and J. Westerholm. An efficient swap algorithm for the lattice Boltzmann method. *Computer Physics Communications*, 176:200–210, 2007.

[98] K. Mattila, J. Hyväluoma, J. Timonen, and T. Rossi. Comparison of implementations of the lattice-Boltzmann method. *Computers & Mathematics with Applications*, 55:1514–1524, 2007.

[99] Guy R. McNamara and Gianluigi Zanetti. Use of the boltzmann equation to simulate lattice-gas automata. *Phys. Rev. Lett.*, 61(20):2332–2335, Nov 1988.

[100] Medical Imaging & Technology Alliance. *Digital Imaging and Communications in Medicine (DICOM), Part 1: Introduction and Overview*. National Electrical Manufacturers Association, 1300 N. 17th Street, Rosslyn, Virginia 22209 USA, 2009.

[101] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. Technical Report UT-CS-94-230, University of Tennessee, Knoxville, TN, USA, May 1994.

[102] Klaus D. Mrike and Eberhard Betz, editors. *Biologie des Menschen: das bewährte Lehr- und Nachschlagewerk*. Quelle und Meyer, Wiebelsheim, 15 edition, 2001.

[103] J. Ni, C.-L. Lin, Y. Zhang, T. He, S. Wang, and B. Knosp. Parallelism of lattice Boltzmann method (LBM) for lid-driven cavity flows. In *High Performance Computing and Applications*

(HPCA2004), Shanghai, China, August 8-10, 2004, accepted and being published in Lecture Note in Computer Science (LNCS). Springer-Verlag Heidelberg, Germany, 2004.

[104] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operation research and financial engineering. Springer, New York, NY, 2. edition, 2006.

[105] Fakir S. Nooruddin and Greg Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205, 2003.

[106] Materialise NV. Mimics Software: The standard in 3D image processing and editing based on CT or MRI data. http://www.materialise.com/mimics.

[107] Roger Peyret, editor. *Handbook of computational fluid mechanics*. Academic Press, London, repr. as paperback with corr. edition, 2000.

[108] G. Pingen, A. Evgrafov, and K. Maute. Topology optimization of flow domains using the lattice Boltzmann method. *Structural and Multidisciplinary Optimization*, 34, 6:507–524, 2007.

[109] G. Pingen, A. Evgrafov, and K. Maute. A parallel Schur complement solver for the solution of the adjoint steady-state lattice Boltzmann equations: application to design optimisation. *International Journal of Computational Fluid Dynamics*, 22, 7:457–464, 2008.

[110] Georg Pingen. *Optimal Design for Fluidic Systems: Topology and Shape Optimization with the Lattice Boltzmann Method*. PhD thesis, Faculty of the Graduate School of the University of Colorado, 2008.

[111] Georg Pingen, Anton Evgrafov, and Kurt Maute. Adjoint parameter sensitivity analysis for the hydrodynamic lattice Boltzmann method with applications to design optimization. *Computers & Fluids*, 38(4):910 – 923, 2009.

[112] Deserno F. Thurey N. Rude U. Lammers P. Wellein G. Zeiser T. Pohl, T. Performance evaluation of parallel large-scale lattice Boltzmann applications on three supercomputing architectures. In *Supercomputing 2004, Proceedings of the ACM/IEEE SC2004 Conference*, page 21, 2004.

[113] A. W. Proetz. Air currents in the upper respiratory tract and their clinical importance. *Ann Otol Rhino Laryngol*, 60:439–467, 1951.

[114] Vanessa Reinefeld. *Rhinoresistometrische und Akustisch-Rhinometrische Untersuchungen zur Effektivität der Nd:YAG-Lasermuschelkaustik*. PhD thesis, Medizinische Fakultt Charit - Universitätsmedizin Berlin, 2003.

[115] Martin Kilian Rheinländer. *Analysis of Lattice-Boltzmann Methods: Asymptotic and Numeric Investigation of a Singularly Perturbed System*. PhD thesis, Universität Konstanz, Universitätsstr. 10, 78457 Konstanz, 2007.

[116] Youcef Saad. *Iterative methods for sparse linear systems*. SIAM, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2. ed. edition, 2003. Includes bibliographical references and index.

[117] Laure Saint-Raymond. From the bgk model to the navier-stokes equations. *Annales Scientifiques de l'cole Normale Suprieure*, 36(2):271 – 317, 2003.

[118] M. Schäfer and S. Turek. Benchmark computations of laminar flow around a cylinder. Preprint 96-03, University Heidelberg, 1996.

[119] P. W. Scherer, I. I. Hahn, and M. M. Mozell. The biophysics of nasal airflow. *Otolaryngologic Clinics of North America*, 22(2):265–78, 1989.

[120] Arne Schffler, Ulrike Hartmann, and Stephan C. Amberg, editors. *Biologie, Anatomie, Physiologie : kompaktes Lehrbuch fr die Pflegeberufe*. Urban und Fischer, Mnchen, 4 edition, 2000.

[121] Tim Schröder. SOMATOM Sensation 64 der schnellste Computertomograph der Welt. In Bestell-Nr. A91100-M-C860-1 CC 65860 ZS 110418., editor, *MEDICAL SOLUTIONS, Changing the Way Healthcare is Delivered*, pages 10–13. Siemens AG, November 2004.

[122] Johann S. Schwegler. *Der Mensch - Anatomie und Physiologie: Schritt fr Schritt Zusammenhänge verstehen*. Thieme, Stuttgart, 2 edition, 1998.

[123] Andreas Siebert. Dynamic region growing. In *In: Vision Interface*, 1997.

[124] P. Skordos. Initial and boundary conditions for the lattice Boltzmann method. *Phys. Rev. E*, 48(6)(6):4823–4842, 1993.

[125] Y. Sone. Asymptotic theory of a steady flow of a rarefied gas past bodies for small knudsen numbers. In R. Gatignol and Soubbaramayer, editors, *Advances in Kinetic Theory and Continuum Mechanics*, page 19. Springer Berlin, 1991.

[126] Maria Specovius-Neugebauer. Approximation of the stokes dirichlet problem in domains with cylindrical outlets. *SIAM J. Math. Anal.*, 30(3):645–677, 1999.

[127] Bjarne Stroustrup. *The C++ programming language*. Addison-Wesley, Reading, Mass., 3. ed., 25. print. edition, 2006.

[128] S. Succi, G. Amati, and R. Benzi. Challenges in lattice Boltzmann computing. *J. Stat. Phys.*, 81:5, 1995.

[129] M. C. Sukop and D. T. Thorne. *Lattice Boltzmann modeling*. Springer, 2006.

[130] D.L. Swift. Inspiratory Inertial Deposition of Aerosols in Human Nasal Airway Replicate Casts: Implication for the Proposed NCRP Lung Model. *Radiation Protection Dosimetry*, 38(1-3):29–34, 1991.

[131] M. M. Tekitek, M. Bouzidi, F. Dubois, and P. Lallemand. Adjoint lattice Boltzmann equation for parameter identification. *Computers & Fluids*, 35:805–813, 2006.

[132] C. Terboven and D. an Mey. OpenMP and C++. In *OpenMP shared memory parallel programming: Second International Workshop on OpenMP (IWOMP 2006)*, pages 300 – 311. Springer Berlin Heidelberg, June 2008.

[133] Fredi Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen: Theorie, Verfahren und Anwendungen*. Vieweg, Wiesbaden, 1 edition, 2005.

[134] J. Valentin. Basic anatomical and physiological data for use in radiological protection: reference values - icrp publication 89. *Annals of the ICRP*, 32:1–277(277), September 2002.

[135] Ewald R. Weibel. *Morphometry of the human lung*. Springer, Berlin, 1963. Bibliography: p. 143-148.

[136] Ewald R. Weibel and Domingo M. Gomez. Architecture of the Human Lung: Use of quantitative methods establishes fundamental relations between size and number of lung structures. *Science*, 137(3530):577–585, 1962.

[137] Ivo Weinhold and Gunter Mlynski. Numerical simulation of airflow in the human nose. *European Archives of Oto-Rhino-Laryngology*, 261(8):452–455, 2004.

[138] Jan-Philipp Weiß. *Numerical analysis of Lattice Boltzmann Methods for the heat equation on a bounded interval*. Universitätsverlag Karlsruhe, Karlsruhe, 2006.

[139] G. Wellein, T. Zeiser, S. Donath, and G. Hager. On the single processor performance of simple lattice Boltzmann kernels. *Computers and Fluids*, 35(8-9):910–919, 2006.

[140] Dieter A. Wolf-Gladrow. *Lattice-Gas, Cellular Automata and Lattice Boltzmann Models, An Introduction*. Lecture Notes in Mathematics. Springer, Heidelberg, Berlin, 2000.

[141] Stefan Zachow, Alexander Steinmann, Thomas Hildebrandt, Rainer Weber, and Werner Heppt. CFD simulation of nasal airflow: Towards treatment planning for functional rhinosurgery. *Int. J. of Computer Assisted Radiology and Surgery*, pages 165–167, 2006.

[142] Eberhard Zeidler. *Applied functional analysis*, volume 2: Main principles and their applications of *Applied mathematical sciences; 109*. Springer, New York, 1995.

[143] T. Zeiser, J. Götz, and M. Stürmer. On performance and accuracy of lattice boltzmann approaches for single phase flow in porous media: A toy became an accepted tool - how to maintain its features despite more and mor complex (physical) models and changing trends in high performance computing!? In N. Shokina et al. M. Resch, Y. Shokin, editor, *Proceedings of 3rd Russian-German Workshop on High Performance Computing, Novosibirsk, July 2007, Springer*, 2008.

[144] T. Zeiser, T. G. Wellein, A. Nitsure, K. Iglberger, U. Rüde, and G. Hager. Introducing a parallel cache oblivious blocking approach for the lattice Boltzmann method. *Progr. in Comp. Fluid Dyn.*, 2006.

[145] Jürgen Zervos-Kopp. *Anatomie, Biologie und Physiologie*. Georg Thieme Verlag KG, Stuttgart, 2007.

[146] Simon Zimny. Numerische Simulation von intranasalen Strömungen mit Lattice-Boltzmann-Methoden. Diplomarbeit, Karlsruhe Institute of Technology (KIT), Universität Karlsruhe (TH), Fakultät für Mathematik, April 2010.

APPENDIX A

The Open Source Library OpenLB

A.1. Project Scope and Overview

The OpenLB project aims at setting up an open source implementation of LBM in an object oriented framework. The code is written in C++ and intended to be used both by application programmers and by developers who may add their own particular dynamics. It supports advanced data structure that take into account complex geometries and parallel program executions. The programming concepts strongly rely on dynamic genericity via the use of object oriented interfaces as well as static genericity by means of templates. This design allows an efficient, straightforward and intuitive implementation of LBM. It is cross-verified for software quality by several reviewers and is presented along with a user guide. To the knowledge of the authors, the OpenLB project is the first attempt to produce a generic platform for LB programming and to share it with the community via a system of open source contributions.

An overview of the computational framework employed in OpenLB and the features of the latest release 0.5 of OpenLB is given in the following:

Overview Computational Framework:
- Platform independent developments
- Modular, extensible C++ code
- Object-oriented programming style
- Template-based genericity
- Open source (GNU General Public License, version 2) .

Overview Features:
- 2D and 3D simulations by means of LBM
- Various LBM, e.g. BGK, MRT, Regularized LB [**89, 88**]
- Various lattice types, e.g. $D2Q9$, $D3Q15$, $D3Q19$, $D3Q27$
- Multi physical models (multiphase, thermal flows)
- Local and non-local boundary condition types, e.g. Inamuro [**73**], He and Zou [**64**], Latt (regularized LB) [**89, 88**]
- Checkpointing for interrupted program executions
- Parallelism for shared and distributed memory platforms
- Visualisation, e.g. based on VTK [**7**]
- Straightforward coupling to external tools for pre- and postprocessing

Furthermore, a documentation for developers as well as user guides for several releases are provided on the OpenLB website `http://www.openlb.org` .

Beside the overview and technical reports, e.g. [**85**], given on the OpenLB website, one finds more details, in particular concerning the computational concepts realised in the OpenLB code, in [**68**]. Ongoing work concerning the hybrid parallelisation strategy (cf. Chapter 3) which has been realised in OpenLB is presented [**66**]. Latt, Malaspinas et al. dedicated their work [**90**] a comparison of various common boundary conditions which are dedicated for LBM. The present tests are performed using the OpenLB library.

A.2. Authors

Several scientists form the basis of the OpenLB project. The names of all participants, both project coordinators and code developers, are arranged in alphabetical order as follows:

Project Coordinators:

- Vincent Heuveline (KIT, Universität Karlsruhe (TH))
- Mathias J. Krause (KIT, Universität Karlsruhe (TH))
- Jonas Latt (Ecole Polytechnique Federale de Lausanne)

Code Developers:

- Mathias J. Krause (KIT, Universität Karlsruhe (TH))
  - OpenMP parallelisation
  - Cuboid data structure for MPI parallelisation
  - Makefiles
- Jonas Latt (Ecole Polytechnique Federale de Lausanne)
  - Development of the OpenLB core
  - Integration and maintenance of added components
- Orestis Malaspinas (Ecole Polytechnique Federale de Lausanne)
  - Alternative boundary conditions (Inamuro [**73**], He and Zou [**64**], Non-linear FD)
  - Alternative LB models (Entropic LB [**5**], MRT [**31**])
- Bernd Stahl (University of Geneva)
  - `MultiBlock` structure for MPI parallelisation
  - Parallel version of (scalar or tensor-valued) data fields
  - *Visualization Toolkit* (VTK) [**7**] output of data
- Simon Zimny (KIT, Universität Karlsruhe (TH))
  - Preprocessing, automated setting of boundary conditions

A.3. Awards

OpenLB has played a major role in the two proposals of the United Airways project (cf. Appendix B.3) for the *Itanium Innovation Awards.* The contest is organised by the *Itanium Solutions Alliance* which is a global consortium of notable hardware, operating system and application vendors.

Both presented solution strategies for numerical simulations of human respiratory flows were awarded in the category of *Humanitarian Impact*

- as *Finalist* in 2007 and
- as *Honorable Mention Finalist* in 2009 .

A detailed presentation of the subject of the two proposals as well as a press release for the 2009 award can be found in Appendix B.3.

APPENDIX B

The United Airways Project

B.1. Project Scope and Challenges

The prospect of the United Airways project is the complete description of the flow behaviour in the human respiratory system by means of highly efficient numerical simulations. The approach followed in the United Airways project relies on an integrative methodology allowing to analyse the interaction of the human nose, paranasal sinuses, larynx and lungs in a coupled way.

On the one hand numerical investigations of respiration is meant to enable fundamental research towards a better understanding of the flow dynamics in the respiratory tracts. On the other hand, it also forms a basis for many applications in the public health sector. Possible applications range from the study of the environmental impact on the human airways or the optimisation of the sprays for rhinitis, sinusitis and asthma treatment to the point of patient-individual simulations of the impact of respiratory tract surgeries.

The challenges that have to be faced then considering fundamental research or such applications are manifold. They encompass dedicated modelling, automated preprocessing (from CT scans to volume meshes), optimal experimental design, mathematical optimisation, real-time simulations, efficient high performance computing, visualisation and many more.

As manifold as the challenges are the research fields which are touched then coping with the challenges. The interdisciplinary framework of the United Airways project relies on the expertise of principle investigation in the following areas

- Medicine (diagnosis/treatment)
- Environmental medicine
- Occupational medicine
- Biotechnology
- Numerical simulation
- Mathematical optimisation
- High performance computing
- Visualisation .

So far, numerical simulations have been performed for the human lungs and the human nose separately. Thereunto, a strategy for a partly automated preprocessing of complex geometries based on CT scans is proposed and realised (cf. Section 6.1, Gengenbach [**47**] and Zimny [**146**]. Further, a hybrid parallelisation concept dedicated for LBM is proposed, implemented in the framework of OpenLB and tested successfully for the geometry of the human lungs (cf. Chapter 3). The obtained numerical results for the human nose and the upper human lungs which are obtained by means of LBM are presented in Chapter 6. A model for the lower generations of the lungs has been implemented, parallelised and tested successfully by Gengenbach. The corresponding numerical and performance results are presented in [**47**].

B.2. Participants

Several scientists, which are mathematicians and physicians, participate in the United Airways project. The names of the current team members with their contributions are listed above in alphabetical order.

- Thomas Gengenbach (KIT, Universität Karlsruhe (TH))
  - Project coordination
  - Modelling (lungs)
  - Numerical simulations based on FEM (lungs)
  - Preprocessing and postprocessing
- Thomas Henn (KIT, Universität Karlsruhe (TH))
  - Automisation of the preprocessing
- Werner Heppt (Städtisches Klinikum Karlsruhe)
  - Consulting, supervising (medical aspects)
- Vincent Heuveline (KIT, Universität Karlsruhe (TH))
  - Consulting, supervising (numerical aspects)
- Mathias J. Krause (KIT, Universität Karlsruhe (TH))
  - Modelling (upper lungs, nose)
  - Numerical simulations based on LBM (lungs, nose)
  - Preprocessing and postprocessing dedicated for LBM
- Simon Zimny (KIT, Universität Karlsruhe (TH))
  - Numerical simulations based on LBM (nose)
  - Preprocessing dedicated for LBM

To be mentioned are also the names of formerly participants, namely Martin Baumann, Evangelos Giotakis, Rolf Maier and Paul Weber. (cf. names is references for 2008 [**16**], 2009 [**15**] and 2010 [**46**]).

B.3. Awards

The United Airways project has been awarded in the contests *Itanium Innovation Awards* 2007 and 2009. The awards are given by the *Itanium Solutions Alliance* in four different categories for approaches of all areas of industry and research developed using *Intel Itanium*-based solutions.

In both years, work related to the United Airways project has been submitted in the category *Humanitarian Impact*.

The focus of the the 2007 proposal has been placed on the development, realisation and obtained perfortance results of the hybrid parallelisation strategy which is presented in *Chapter* 3. The proposal has been awarded as *Finalist* together with others handed in by applicants of e.g. the Stanford University and the Imperial College London.

In 2009, efficient high performance computing for the numerical simulation of the flow in the human lungs form the basis of the application. Performance results obtained for an MPI-based implementation, which is especially designed for complex geometries like the human lungs and which is part of the open source code OpenLB, are presented. Beside, topics regarding an adequate preprocessing and postprocessing are addressed as well. The presented work has earned the title *Honorable Mention Finalist*. In the following, the corresponding press release from the *Intel Corporation* is imprinted:

KARLSRUHE UNIVERSITY EARNS HUMANITARIAN IMPACT HONORABLE MENTION IN THE 2009 ITANIUM SOLUTIONS ALLIANCE INNOVATION AWARDS

PORTLAND, Ore., July 27 - The Itanium Solutions Alliance today announced that Karlsruhe University has been awarded an honorable mention for its entry in the 2009 Innovation Awards program. Karlsruhe University's United Airways project in Germany used Itanium-based HP hardware to analyze the interaction of the human nose, sinuses, larynx and lungs with the goal of compiling a complete numerical simulation of flow behavior in the human respiratory system. The end benefits of this project include optimizing asthma sprays, improving the quality of medical operations and understanding the impact of respirable dust.

The Itanium Solutions Alliance Innovation Awards were designed to recognize and reward end users and developers for outstanding use of Intel Itanium-based servers in their applications. A panel of distinguished judges evaluated submissions on a number of criteria such as difficulty of challenge, results produced, and originality of the solution.

"For our second annual Innovation Awards, we received a large number of very compelling entries from across the globe, showcasing a wide variety of innovative applications backed by Itanium-based technology," said Joan Jacobs, president and executive director of the Itanium Solutions Alliance. "We are pleased to recognize this year's finalists for their groundbreaking work with Itanium-based systems from across the spectrum of industry and research."

The Humanitarian Impact category awards the innovative use of Itanium-based systems to deliver results that benefit humanity through research, social improvements or other humanitarian efforts. Examples include natural disaster modeling and prediction, resource management, health care advances, and biosciences research.

Nomenclature

Abbreviations

AD	Automatic differentiation
ALB	Adjoint lattice Boltzmann
ALBM	Adjoint lattice Boltzmann methods
BGK	Bhatnagar, Gross and Krook
CAD	Computer-aided design
CFD	Computational fluid dynamics
CT	Computer tomography
DICOM	Digital imaging and communications in medicine
FD	Finite difference
FEM	Finite element method
I/O	Input/output
EOC	Experimental order of convergence
GMRES	Generalised minimal residual method
HPC	High performance computing
LB	Lattice Boltzmann
LBM	Lattice Boltzmann methods
LDC	Lid-driven cavity
LGCA	Lattice gas cellular automata
MLUP/s	Million fluid lattice-site updates per second
MLUP/ps	Million fluid lattice-site updates per process and second
MRI	Magnetic resonance imaging
MRT	Multi relaxation time
NUMA	Non-uniform memory access
SMP	Symmetric multiprocessing
UMA	Uniform memory access
VTK	Visualization toolkit

Fluid Constants, Parameters and Variables

T	Absolute temperature	Subsection 1.3.2
L	Characteristic length	Subsection 1.3.5
U	Characteristic speed	Subsection 1.3.5
ρ	Density	Subsection 1.1.1
$\boldsymbol{F}$	Density of volume force	Subsection 1.1.3 (1.12)
μ	Dynamic viscosity	Subsection 1.1.5 (1.30)
ν	Kinematic viscosity	Subsection 1.3.5 (1.71)
Kn	Knudsen number	Subsection 1.3.5 (1.69)
Ma	Mach number	Subsection 1.3.5 (1.70)
$\overline{c}$	Mean absolute thermal velocity	Subsection 1.3.4 (1.65)
l_f	Mean free path	Subsection 1.2.1
ω	Mean free time	Subsection 1.3.4 (1.66)
f	Particle density function	Subsection 1.2.2 (1.35)
σ	Particle diameter (microscopic)	Subsection 1.2.1
p	Pressure	Subsection 1.1.1 (1.2)
Re	Reynolds number	Subsection 1.3.5 (1.73)
c_s	Speed of sound	Subsection 1.3.5
$\boldsymbol{D}$	Strain rate tensor	Subsection 1.1.1 (1.3)
$\boldsymbol{P}$	Stress tensor	Subsection 1.1.3 (1.15)
R	Universal gas constant	Subsection 1.3.2
$\boldsymbol{u}$	Velocity of flow (macroscopic)	Subsection 1.1.1 (1.1)
$\boldsymbol{v}$	Velocity of molecules (microscopic)	Subsection 1.2.1

Norms and Spaces

$\boldsymbol{x} \cdot \boldsymbol{y}$	Standard scalar product for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, n \in \mathbb{N}$
$\|x\|$	Euklidean norm, $\sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}$ for $\boldsymbol{x} \in \mathbb{R}^n, n \in \mathbb{N}$
d	Dimension of the position space Ω, $d = 2, 3$
$\mathbb{R}_{\geq 0}$	Positive real numbers with zero, $x \in \mathbb{R}$ with $x \geq 0$
$\mathbb{R}_{>0}$	Positive real numbers $x \in \mathbb{R}$, with $x > 0$
$\mathbb{R}^d_{\boldsymbol{n}^+}$	Positive half space, $x \in \mathbb{R}^d$ with $\boldsymbol{x} \cdot \boldsymbol{n} > 0$ for $n \in \partial\Omega$
$\mathbb{R}^d_{\boldsymbol{n}^-}$	Negative half space, $x \in \mathbb{R}^d$ with $\boldsymbol{x} \cdot \boldsymbol{n} < 0$ for $n \in \partial\Omega$
h	Discretisation parameter, $h \in \mathbb{R}_{>0}$
Ω	Domain filled by the fluid, $\Omega \subseteq \mathbb{R}^d$
Ω_h	Lattice, uniform mesh with spacing h, $\Omega_h \cong X_h \subseteq \mathbb{N}^d$
$\partial\Omega$	Boundary of Ω, $\partial\Omega \cong X \subseteq \mathbb{R}^{d-1}$
$\partial\Omega_h$	Boundary of Ω_h, $\partial\Omega_h \cong X_h \subseteq \mathbb{N}^{d-1}$
I	Time interval $[t_0, t_1] = I \subseteq \mathbb{R},\ 0 \leq t_0 < t_1 < \infty$
I_h	Discrete time interval, $I_h := \left\{t \in I : t = t_0 + h^2 k, k \in \mathbb{N}\right\}$
$\boldsymbol{n}(\boldsymbol{r})$	Unit outward normal at $\boldsymbol{r} \in \partial\Omega,\ \boldsymbol{n}(\boldsymbol{r}) \in \mathbb{R}^d$
$\boldsymbol{I}_n$	Identity matrix of dimension $n \in \mathbb{N}$
$\|\xi\|_{L^2(X)}$	$L^2(X)$-norm over $X \subseteq \mathbb{R}^d$, $\|\xi\|_{L^2(X)} := (\int_X \xi(x)\ dx)^{1/2}$ for $\xi \in L^2(X)$
$\|\xi^h\|_{L^2(X_h)}$	$L^2(X_h)$-Norm over $X_h \subseteq \mathbb{N}^d$, $\|\xi^h\|_{L^2(X_h)} := (\frac{1}{h^d} \sum_{x \in X_h} \xi(x))^{1/2}$ for $\xi^h \in L^2(X_h)$
$C_0^n(X)$	All $\xi \in C_0^n(X)$ are n-times continuous differentiable in $X \subseteq \mathbb{R}^d$ with compact support in $Y \subseteq X$, $n \in \mathbb{N}$ or $n = \infty$, X open

Differential Operators

$\frac{d}{dx}$	Total derivative
$\frac{\partial}{\partial x}$	Partial derivative
$\boldsymbol{\nabla}_{\boldsymbol{x}}$	Gradient
$\boldsymbol{\nabla}_{\boldsymbol{x}} \cdot$	Divergence
$\boldsymbol{\nabla}_{\boldsymbol{x}} \times$	Curl
$\boldsymbol{\nabla}_{\boldsymbol{x}} \otimes$	Jacobian
$\Delta_{\boldsymbol{x}}$	Laplacian