Lattice Boltzmann Methods for Particulate Flows with Medical and Technical Applications

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN
(Dr. rer. nat.)

von der Fakultät für Mathematik des
Karlsruher Instituts für Technologie (KIT)
genemigte

DISSERTATION

von

Dipl.-Math. techn. Thomas Henn
aus Bretten.

Tag der mündlichen Prüfung:	19. Oktober 2016
Referent:	PD Dr. Gudrun Thäter
Korreferent:	Prof. Dr. Willy Dörfler

VORWORT

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Numerische und Angewandte Mathematik des Karlsruher Instituts für Technologie (IANM).

Ich spreche all jenen meinen Dank aus, die mich unterstützt und zur Entstehung dieser Arbeit beigetragen haben. Speziell danke ich Frau PD Dr. Gudrun Thäter dafür, dass sie die Betreuung meiner Arbeit übernommen und mir zu jeder Zeit den Rücken freigehalten hat. Herrn Prof. Dr. Willy Dörfler danke ich, dass er mich in seine Arbeitsgruppe aufgenommen und die Position des Koreferenten übernommen hat. Herrn Dr. Mathias J. Krause, der vermutlich die meiste Arbeit mit mir hatte, danke ich für unzählige energische Gespräche, die intensive Betreuung, manche Idee und für seine bemerkenswerte Geduld. Herrn Prof. Dr. Andreas Kirsch danke ich für meine Zeit in der Arbeitsgruppe Inverse Probleme, und Herrn Prof. Dr.–Ing. Hermann Nirschl vom Institut für Mechanische Verfahrenstechnik und Mechanik, Bereich Verfahrenstechnische Maschinen (VM), danke ich für die erfolgreiche Zusammenarbeit der Arbeitsgruppen.

Ich danke meinen Kollegen vom IANM und VM. Das angenehme Arbeitsklima und die hervorragende Zusammenarbeit haben sehr zum Gelingen dieser Dissertation beigetragen. Dank gilt auch den Diplomanten und Masterstudierenden des IANM, die meine Forschungen mit ihren Arbeiten vorangebracht haben.

Besonders danke ich meiner Familie, meinen Eltern, Schwestern und meiner Freundin Nathalie für viel Motivation und Unterstützung in den vergangenen fünf Jahren.

ZUSAMMENFASSUNG

Partikelströmungen treten in zahlreichen natürlichen sowie künstlichen Vorgängen auf, beispielsweise als Feinstaub in den menschlichen Atemwegen, als Sediment in Flüssen oder als Feststoff–Fluid Gemisch bei Filtrationen. Simulation von Partikelströmen kommen zum Einsatz, wenn physische Untersuchungen nicht möglich sind. Darüber hinaus können sie Kosten experimenteller Studien verringern.

Es hat sich gezeigt, dass die transiente Simulation von Strömungen mit einer großen Zahl an beliebig geformten Partikeln den herkömmlichen numerischen Methoden insbesondere bei der Parallelisierung Probleme bereitet. In dieser Arbeit wird die Lattice Boltzmann Methode (LBM) als neues Verfahren zur numerischen Simulation von Strömungen auf Partikelströmungen angewendet und ihr Nutzen untersucht.

Am Beispiel von verdünnten Suspensionen, wird ein Euler–Lagrange Ansatz für die LBM vorgestellt. Dabei wird nur die Wirkung der Trägerphase auf die Partikel modelliert und die Rückkopplung vernachlässigt. Der Fokus liegt dabei auf der Parallelisierung der diskreten Partikelphase. Die neu vorgestellte Methode wird validiert und zur Simulation von Feinstaub in einer Verzweigung der Bronchien und der komplexen Geometrie einer patientenspezifischen Nasenhöhle verwendet.
Im Anschluss daran wird ein Euler–Euler Ansatz entwickelt, bei dem beide Phasen als stetig angenommen werden und erstmals mit einer LBM gelöst werden. Die Simulationen der vorhergehenden Methode in den Atemwegen werden wiederholt. Als Resultat zeigt sich eine sehr gute Übereinstimmung.

Das letzte Kapitel der Arbeit behandelt Suspensionen mit hohem Feststoffanteil. In diesem Fall kann die Rückkopplung der Partikel auf das Fluid und teilweise auch die Interaktion der Partikel untereinander nicht mehr vernachlässigt werden. Zuerst werden zwei Methoden zur Simulation vieler kleiner Partikel vorgestellt und untersucht.
Anschließend wir eine Methode zur Simulation bewegter poröser Medien entwickelt und numerisch aufgelöste Partikel mit einem festen Kern und einem glatten porösen Übergang zur reinen Fluidphase modelliert. Die neue Methode wird validiert durch Simulationen und Vergleich der Ergbnisse mit der Literatur. Für das Problem zweier sedimentierender Partikel wird die Konvergenzordnung experimentell bestimmt. Zum Schluss wird die Sedimentation von 24 unterschiedlich geformten Partikeln simuliert.

Ergebnis der Untersuchungen ist, dass durch eine ganzheitliche Verwendung der LBM, das heißt für beide Phasen, die Simulation von Partikelströmungen effizient gestaltet werden kann. Dies führt einen großen Schritt näher an das Ziel der Simulation einer großen Zahl unterschiedlich geformter Partikel.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ADE	Advection–Diffusion Equation
ALE	Arbitrary Lagrangian–Eulerian
BE	Boltzmann Equation
bEM	backward Euler Method
BGK	Bhatnagar–Gross–Krook
CA	Cellular Automata
CFD	Computational Fluid Dynamics
CI	Confidence Interval
CoA	Coarctation of the Aorta
CPU	Central Processing Unit
CT	Computer Tomography
DEM	Discrete Element Method
DKT	Drafting, Kissing, Tumbling
DNS	Direct Numerical Simulation
EDF	Equilibrium Distribution Function
EE	Euler–Euler
EL	Euler–Lagrange
EOC	Experimental Order of Convergence
FDM	Fictitious Domain Method
FEM	Finite Element Method
fEM	forward Euler Method
FVM	Finite Volume Method
HLBM	Homogenised Lattice Boltzmann Method
IBM	Immersed Boundary Method
LB	Lattice Boltzmann
LBE	Lattice Boltzmann Equation
LBM	Lattice Boltzmann Method
LES	Large Eddie Simulation
LGCA	Lattice Gas Cellular Automata
MD	Molecular Dynamics
MEA	Momentum Exchange Algorithm
MPI	Message Passing Interface
MRI	Magnetic Resonance Imaging
NBS	No Binary Search
NSE	Navier–Stokes Equation
ODE	Ordinary Differential Equation
PU	Processing Unit
REV	Representative Elementary Volume

1 INTRODUCTION

Airborne particulates pose a serious health risk. On the one hand, they are known to cause respiratory diseases such as lung cancer or asthma. On the other hand, purposefully used in nasal or asthma sprays they can help to treat such diseases. It is therefore of great interest to gain a deeper understanding of their dynamics. From a physics point of view, airborne particulates are an example of particulate flows as they appear in numerous technical and non-technical processes in our everyday life. They can be found in engineering as well as chemical, biological and medical applications e. g. as the already mentioned respirable dust in the human tracheobronchial system, the movement of sediment in rivers or in filtration devices. Their simulation can reduce the cost of experimental studies and is inevitable whenever a physical investigation is unfeasible.

Often, suspensions are polydisperse, i.e. the solid objects submerged in the fluid differ in size, mass or shape. It is observed experimentally that the non-uniformity crucially influences the dynamics of suspensions, e. g. in nanomedicine applications, where non-spherical nanoparticles improved the ability to target tumours over spherical ones [143]. Many effects are still not fully understood and therefore in the focus of interdisciplinary research. For a deeper understanding of such effects, sophisticated models as well as numerical simulations can deliver highly valuable insights and give a better idea of how to design useful experiments.

A great and so far unsolved challenge is finding an efficient approach that allows to predict the dynamics of thousands or millions of differently and arbitrarily shaped objects. Thereby, especially the calculation of the interaction of the objects with each other as well as with the fluid causes enormous computational costs.

Conventional numerical methods, such as Finite Volume Method (FVM) and Finite Element Method (FEM) can be of limited efficiency for the simulation of non-steady flows with immersed particles. This is partly due to their necessity to recompute a geometry dependent mesh for each time step and partly due to the lack of highly parallelisable algorithms [88]. Since this is inherent to the chosen mathematical model this will probably not be solved in the future.

Nevertheless, in the past decades the Lattice Boltzmann Method (LBM) ascended to be an efficient computational tool for simulation of complex flows. The LBM is a comparably simple explicit algorithm originating of a discretisation of the Boltzmann Equation (BE). It is based on an equidistant grid and the necessity of a geometry–adapted mesh does not arise. The data employed during one time step are local in their position in the physical domain, as well as in their location in the memory. Therefore the LBM demands a parallelisation by a domain decomposition. In fact, LBMs have already been shown to exhibit an almost ideal scaling on more than 9000 cores [47]. Numerical simulation of submerged particles with LBM began with the articles by Ladd in 1994 [113, 114]. However, their usage in this context still poses challenges. For example the parallelisation used for the carrier phase cannot be simply transferred to the particle phase.

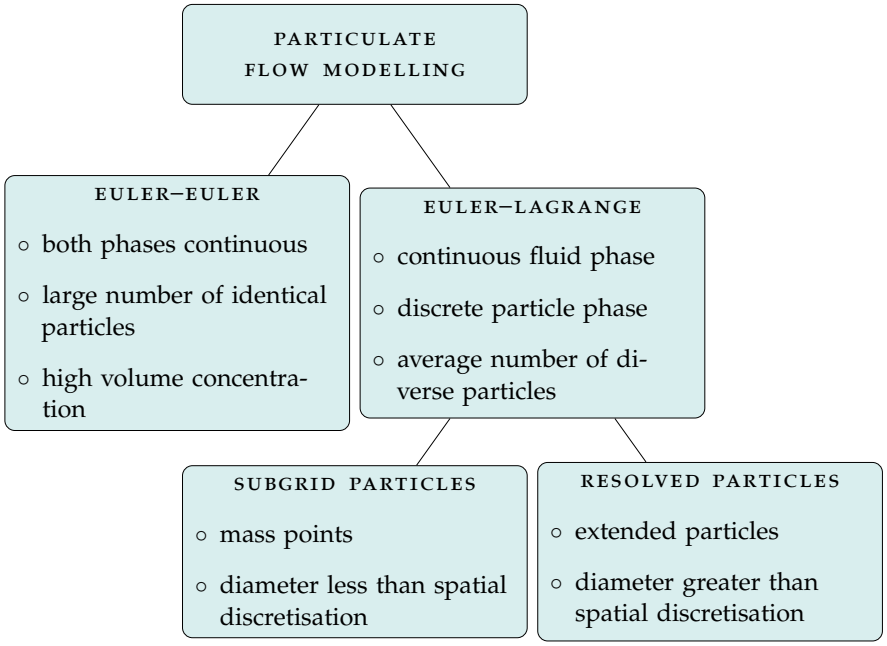


Figure 1.1: Categorisation of particulate flows by the modelling.

Numerical simulation of particulate suspensions requires two parts. One to simulate the carrier phase and one to simulate the particle phase. For both parts one has to decide on how to formulate a model, which in case of the particle phase can be continuous or discrete and then choose a numerical method. Additionally the phase interaction can be modelled in several ways, depending on particle size and concentration. These two aspects of simulation of particulate flows are explained in more detail in the following two sections and are illustrated in Figures 1.1 and 1.2. There is no panacea for simulating particulate flows.

1.1 CATEGORISATION BY FLOW MODEL

Different mathematical models are required for the different regimes of particle sizes. In general, two main classes can be distinguished: firstly, Euler–Euler (EE) methods where the solid phase is described by its particle concentration and continuously modelled by an Advection–Diffusion Equation (ADE) and secondly Euler–Lagrange (EL) methods where the trajectory of each discrete particle is computed according to Newton's law of motion. For both methods the carrier phase is usually modelled as an incompressible Newtonian fluid by a Navier–Stokes Equation (NSE). The categorisation by the flow model is illustrated in Figure 1.1.

The EE approach is usually employed if the number of suspended particles is too large to be computed individually or the exact trajectory of a particle is not of interest. This is mostly the case for a high volume concentration of comparably small particles. The computational costs of an EE scheme scale solely with the numerical resolution of the computational domain and not with the amount of particles.

If the exact trajectories of the particles are of interest, commonly an EL approach is used. In this work two types of the EL approach are used. Firstly methods for *subgrid particles* are proposed. In this case the particles are small in comparison to the discrete spatial step of the system and are therefore

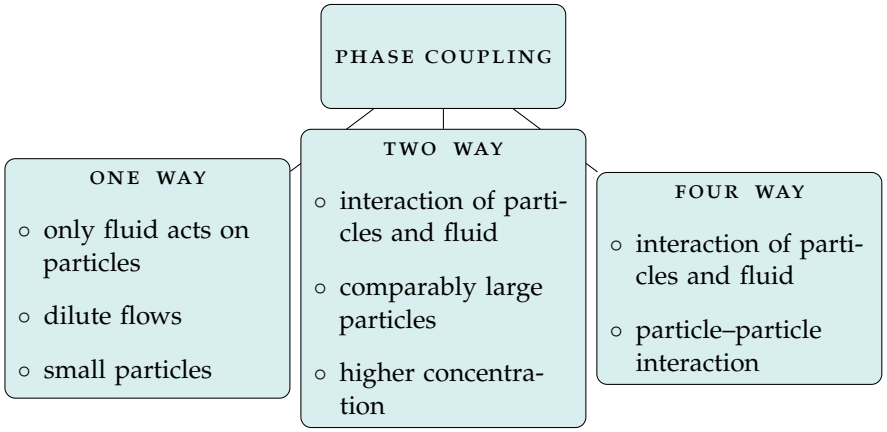


Figure 1.2: Categorisation of particulate flows by the phase coupling.

handled as mass points. Secondly, a method for *resolved particles* that are of the same scale as the characteristic length is proposed.

1.2 CATEGORISATION BY PHASE COUPLING

Simulations of particulate flows can also be classified by the kind of coupling between the particle phase and the fluid phase. In dilute particle flows, when the particles are influenced by the fluid and the feedback is neglected, one speaks of *one way coupling*. *Two way coupling* additionally respects the back coupling of the particles on the fluid, and if additionally particle–particle interaction is considered, the term *four way coupling* is used. The categorisation by the phase coupling is also illustrated in Figure 1.2.

The preferred model depends on the volume fraction of particles $\phi = NV^P/V$, where the number of particles is denoted by N, their volume by V^P and the combined volume of the fluid and the particles by V. For turbulent flows Elghobashi [49] states that for very low values of $\phi < 10^{-6}$ the particles have negligible effect on turbulence and a one way coupling can be applied. For $10^{-6} \leqslant \phi < 10^{-3}$ a two way coupling and for dense suspensions with $\phi \geqslant 10^{-3}$ a four way coupling should be applied.

1.3 THESIS AIMS AND STRUCTURE

The main aim of this thesis is to contribute models and numerical schemes towards an accurate as well as efficient simulation of a huge number of arbitrarily shaped particles. We therefore develop holistic mesoscopic models and simulation approaches using the LBM, that on massively parallel machines efficiently solve a variety of problems of particulate flows

The thesis is divided in three main chapters that with increasing complexity deal with mathematical models for particulate flows and their numerical solution. The chapter following this introduction, Chapter 2, has the aim to introduce the LBM and demonstrate its capabilities as an efficiently parallelisable numerical solver of fluid flows. With this objective, we describe the carrier phase on three different scales, namely the microscopic, mesoscopic and macroscopic scale. While the continuous macroscopic description in form of the NSE leads to the conventional numerical methods, we use the mesoscopic view in form of the BE to derive the LBM. We then introduce special dynamics to handle body forces, turbulence and for the first time

moving porous media. The latter will be called *Homogenised Lattice Boltzmann Method (HLBM)*.
Each logic section is closed by a real world or academic *application* to validate the methods and to demonstrate their potential. Chapter 2 includes a simulation of blood flow through the complex geometry of a patient-specific aortic arch with an coarctation. The simulation is executed for increasing resolution and while for the smaller resolutions a stabilisation in form of a Smagorinsky scheme is necessary, the highest resolution is able to resolve the Kolmogorov scales leading to a Direct Numerical Simulation (DNS). This however requires massively parallel processing on 512 CPUs, demonstrating the excellent scalability of the LBM. From the medical point of view, the focus is on the blood pressure and the pressure drop around the coarctation. The blood pressure in the ascending aorta is computed to 115.48/64 mm Hg, and very precisely matches the measured values of 115/64 mm Hg.

Chapter 3 has the objective of investigating dilute particulate flows. It therefore introduces dilute particles as a second phase that is affected by the carrier phase through a one way coupling. Using an EL approach, we first propose and theoretically analyse two parallelisation strategies for the computation of particle trajectories, that are based on the domain decomposition used for the carrier phase. With regard to simulation of equally distributed particles the algorithm promising higher efficiency is implemented and tested, with the result of a super-linear scaling. In order to provide some kind of verification the Experimental Order of Convergence (EOC) has been computed.
The theoretically discussed EL approach is applied to a simulation of the inhalation of respirable dust. Here, the used computational domain is a patient-specific geometry of a nasal cavity with a peripheral obstructive ventilation disorder and additionally including paranasal sinuses. For the first time two transient respiratory cycles with a repeated particle injection are computed and particle deposition patterns are determined. It is found that for continuous injection the particle escape rates through the nostrils increase during expiration. Most particles deposit in the anterior region and remain in the sinuses after one completed respiratory cycle.
We then turn towards an EE approach and for the first time simulate a continuous particle phase using LBM. Here, it was necessary to develop a new boundary condition and a new stabilisation scheme for the LBM in context of the ADE.

Chapter 4 aims at the back coupling of the particles on the fluid and covers two and four way coupled flows. We start by introducing a tree-based and a grid-based contact detection algorithm. Both algorithms use geometric information to reduce the number of potentially interacting particle pairs. Their run-time is compared for three different particle distributions. The result is, that for structured particle distributions the run-times of both algorithms are comparable, while for a random distribution the grid-based algorithm clearly outruns the tree-based.
Two different methods for two way coupled flows of subgrid particles are newly proposed. The first one utilises the HLBM and the other one a direct forcing approach. Both methods are used to simulate one sedimenting particle and the computed sedimentation velocity is compared to an analytical result. It is found that the HLBM is only of limited usability in this context, but the forced scheme shows promising results.
Finally we turn towards the simulation of numerically resolved particles that are large enough to cover several discrete fluid nodes. Existing meth-

ods for this setting are the Immersed Boundary Method (IBM) and Fictitious Domain Method (FDM), that reintroduce a secondary Lagrangian mesh to approximate the particle domain and require additional computation time. We propose a method that uses the already available lattice. Therefore the particles are modelled as a moving porous medium with a solid center and a smooth transition over the boundary to a pure fluid. The implementation applies the proposed HLBM. The method has been used to revisit academic applications such as *Flow around a cylinder* and *Kissing, Drafting, Tumbling*. The produced results are found to be in excellent agreement with results in the literature. Also, it is found that the method is of linear EOC. Finally the method is used to simulate the sedimentation of 24 particles of different shapes.

The results of our investigations are that the LBM can be efficiently used for simulations of particulate flows. It thereby solves the mathematical models of both phases and also handles their interactions and partly handles interparticle collisions. This is a great leap closer to the aim of simulating fully coupled flows of large numbers of arbitrarily shaped particles.

1.4 METHODS AND APPLICATION

Throughout the thesis several specialisations of the LBM as well as numerical methods are introduced or newly proposed. The following table provides an overview on which methods were used in the applications.

Method \ APPLICATION	AORTA	BIFURCATION EL	NASAL CAVITY	BIFURCATION EE	SUBGRID HLBM	SUBGRID FORCED	RESOLVED HLBM
Force model 2.3.3						×	
Turbulence model 2.3.2	×			×			
HLBM 2.3.5					×		×
Euler method 3.1.2		×	×	×	×		
Verlet method 3.1.2						×	×
Gravitation 3.1.1					×	×	×
Stokes drag 3.1.1		×	×	×	×	×	
p–p interaction 4.1						×	(×)

Table 1.1: Overview of methods

1.5 OPENLB

The algorithms and methods proposed in this thesis have been implemented using and expanding the open source LBM library `OpenLB`[1]. Its development has begun in 2006 as a cooperation of Mathias J. Krause (KIT) and Jonas

1 `http://www.openlb.net`

Latt (Université de Genève). Meanwhile the developer team has grown to ten active developers, supplemented by ten more former developers.

The `OpenLB` repository now contains more than 300.000 lines of code, of which more than 100.000 are publicly available. During development the focus is on an easy utilisation and broad applicability. It can be compiled by a large range of compilers and is executable on all common operating systems.

`OpenLB` provides methods for simulation of non-Newtonian flows, turbulent flows, thermal flows, multi-phase flows and since version 1.0 also particulate flows. It uses the Message Passing Interface (MPI) library for parallelisation and has been shown to efficiently scale up to 256 CPUs [110].

1.6 RELATED PUBLISHED ARTICLES

During the past years several journal articles have been published, that form the basis of parts of this thesis. They are listed below including a description of where they are reproduced in this work. Throughout the thesis we will again point out whenever a particular section has been published before.

- T. Henn, V. Heuveline, M. J. Krause and S. Ritterbusch. "Statistical Atlases and Computational Models of the Heart. Imaging and Modelling Challenges: Third International Workshop, STACOM 2012, Held in Conjunction with MICCAI 2012, Nice, France, October 5, 2012, Revised Selected Papers". In: ed. by O. Camara, T. Mansi, M. Pop, K. Rhode, M. Sermesant and A. Young. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. Chap. Aortic Coarctation Simulation Based on the Lattice Boltzmann Method: Benchmark Results, pp. 34–43. DOI: 10.1007/978-3-642-36961-2_5

  ABSTRACT

  We investigate a patient-specific blood flow simulation through a transverse aortic arch with a moderate thoracic Coarctation of the Aorta (CoA), where particular attention is paid to the blood pressure gradient through the coarctation. The challenge in this context is the complex geometry containing a stenosis, which results in complex flow patterns. The fluid is assumed to be incompressible and Newtonian. Its dynamic is usually described by a NSE with appropriate boundary conditions. Instead, we modelled the problem mesoscopically by a family of BGK-Boltzmann equations those solutions reaches that of a corresponding Navier–Stokes system in a certain limit. For discretisation we take advantage of LBM, which are realised within the open-source library `OpenLB`. A realistic transient flow profile of the cardiac output for a human at rest was used to specify the inflow boundary condition at the aortic root, whereas the outflow at the descending aorta was modelled by a pressure boundary condition. A short introduction to LBM is provided and especially the used boundary conditions are introduced in detail. The exact simulation setup is stated and the obtained results are discussed.

  COMMENT

  This paper results from the participation at the "1st Simulation on Aortic Coarctation Challenge" which was part of the conference "Stat-

istical Atlases and Computational Modelling of the Heart (MICCAI)" in Nice 2012. It is partly reproduced in subsection 2.5.

- T. Henn, G. Thäter, W. Dörfler, H. Nirschl and M. J. Krause. "Parallel dilute particulate flow simulations in the human nasal cavity". In: *Computers & Fluids* 124 (2016), pp. 197–207. DOI: 10.1016/j.compfluid.2015.08.002

ABSTRACT

When simulating time dependent particulate flows, one faces the dilemma that the domain decomposition used for fluid simulation is not optimal for parallel computation of particle trajectories. Therefore, the article proposes and compares two parallelisation strategies for the particle phase based on the fixed domain decomposition approach used in the open source lattice Boltzmann framework OpenLB. The *communication optimal* strategy is found to be more efficient in the case of homogeneously distributed particles. Convergence studies and performance tests are conducted using a simplified geometry of the human lungs and show excellent parallel speedup. The implemented strategy is used to simulate time dependent particulate flows of micro-particles in a patient-specific geometry of a human nasal cavity including paranasal sinuses. Dilute, uniformly distributed particles are released once at the start of inspiration, as well as repeatedly during the entire inspiratory cycle, which leads to a more homogeneous distribution. It is found that the deposition rates vary for the different injection methods.

COMMENT

This article partially forms Sections 3.3 and 3.4.

- R. Trunk, T. Henn, W. Dörfler, H. Nirschl and M. J. Krause. "Inertial dilute particulate fluid flow simulations with an Euler–Euler lattice Boltzmann method". In: *Journal of Computational Science* (2016). DOI: http://dx.doi.org/10.1016/j.jocs.2016.03.013

ABSTRACT

Systems of dilute particulates, affected by inertia, with sizes in the range of 1 µm to 1 mm are of great interest in the design of many mechanical devices. For the simulation of such particle-laden flows most often EL approaches are applied, that yield massive computational costs, if a high accuracy e. g. in the deposition pattern is desired. In contrast to that, EE approaches scale only with the resolution of the chosen discretisation in computational effort. However, the stabilisation required for the considered convection dominated regime and in general the formulation of boundary conditions on this macroscopic scale are more challenging. In this article a stabilised extension to the EE approach is proposed, together with appropriate boundary conditions, to also account for drag forces and yield viable results for a wide range of Péclet and Reynolds numbers. The two-component system is solved using a LBM and the resulting scheme is applied to a simplified geometry of a human lungs bifurcation. The numerical results are

validated by comparison to other works, that apply an EL approach, regarding the deposition and its dependency on the Stokes number. After consideration of the effect of artificial diffusion, the results are found to be in excellent agreement.

COMMENT

This article partially forms Section 3.5.

- T. Henn, F. Klemens, G. Thäter and M. J. Krause. "Particle Flow Simulations with Homogenised Lattice Boltzmann Methods". In: *Submitted to Particuology* (2016)

ABSTRACT

In this article, a new approach for the simulation of arbitrarily shaped particles submersed in viscous fluid is proposed. It is obtained by adapting the velocity parameter of the equilibrium distribution function of a standard LBM. It is validated by comparison of exemplary simulations to results in literature, as well as convergence analysis. Pressure fluctuation occurring in Ladd's approach [113] are greatly reduced. In comparison to the IBM the new approach does not require cost intensive interpolations. The parallel efficiency of LBM remains. An intrinsic momentum transfer is observed during particle–particle collisions. To show the capabilities of the method, sedimentation of particles of several shapes is simulated omitting an explicit particle collision model.

COMMENT

This article partially forms Section 4.3.1.

- H. Mirzaee, T. Henn, M. J. Krause, L. Goubergrits, C. Schumann, M. Neugebauer, T. Kuehne, T. Preusser and A. Hennemuth. "MRI-based computational hemodynamics in patients with aortic coarctation using the lattice Boltzmann methods: Clinical validation study". In: *Journal of Magnetic Resonance Imaging* (2016). DOI: 10.1002/jmri.25366

ABSTRACT

Purpose: To introduce a scheme based on a recent technique in computational hemodynamics, known as the LBM, to noninvasively measure pressure gradients in patients with a CoA. To provide evidence on the accuracy of the proposed scheme, the computed pressure drop values are compared against those obtained using the reference standard method of cauterisation.

Materials and Methods: Pre and posttreatment LBM-based pressure gradients for 12 patients with CoA were simulated for the time point of peak systole using the open source library `OpenLB`. 4D flow-sensitive phase-contrast Magnetic Resonance Imaging (MRI) at 1.5 T was used to acquire flow and to setup the simulation. The vascular geometry was reconstructed using 3D whole-heart MRI. Patients underwent pre and postinterventional pressure cauterisation as a reference standard.

Results: There is a significant linear correlation between the pretreatment catheter pressure drops and those computed based on the LBM

simulation, $r = 0.85$, $p < 0.001$. The bias was -0.58 ± 4.1 mmHg and was not significant ($p = 0.64$) with a 95% Confidence Interval (CI) of -3.22 to 2.06. For the posttreatment results, the bias was larger and at -2.54 ± 3.53 mmHg with a 95% CI of -0.17 to -4.91 mmHg.

Conclusion: The results indicate a reasonable agreement between the simulation results and the catheter measurements. LBM-based computational hemodynamics can be considered as an alternative to more traditional computational fluid dynamics schemes for noninvasive pressure calculations and can assist in diagnosis and therapy planning.

COMMENT

This article is the result of a cooperation with the *Fraunhofer MEVIS, Institute for Medical Image Computing* and a followup article to Henn et al. [80].

2 MODELLING AND SIMULATION OF FLUID FLOWS

This chapter is introducing the Lattice Boltzmann Method (LBM). This method is central in our simulations since it is used to solve the carrier phase of particle laden flows and sometimes also to solve the particle phase. In the first section we start by explaining three differently scaled perspectives of the to-be simulated fluid. We then concentrate on the mesoscopic scale, by introducing the Boltzmann Equation (BE) and its discretisation, which results in the LBM in the third section. It is followed by some aspects concerning its implementation, specifically the structure of the data, parallelisation and the voxelisation process. Finally, we present a medical application of the LBM of blood flow through an patient specific aortic arch with a coarctation.

2.1 FLUID SYSTEMS AT DIFFERENT SCALES

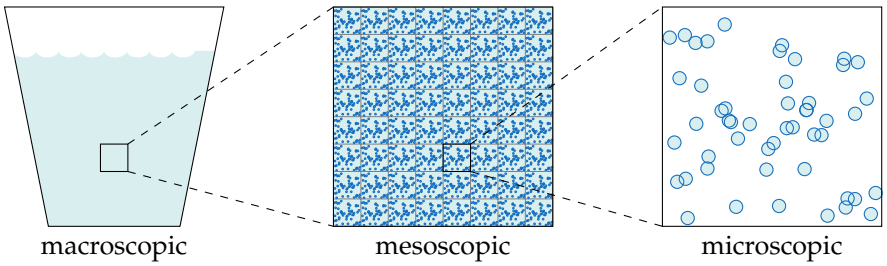


Figure 2.1: Different scaling approaches: Macroscopic, mesoscopic and microscopic.

Mathematical models of fluids can be accomplished in several ways depending on the scale on which the selected fluid is considered. The probably most intuitive choice is to observe the domain e.g: a glass of water "from the outside" (macroscopic, see Figure 2.1) and directly model and compute the observable quantities such as the fluid velocity and pressure. Classical models for this are, e. g. the Navier–Stokes Equations (NSEs), which can be solved inter alia by various Finite Element Methods (FEMs) or Finite Volume Methods (FVMs). In this approach, the fluid is assumed to be continuous, i.e. the composition of microscopic particles is not important.

In contrast, the microscopic approach attempts to track the trajectory of each single water molecule and to compute the interaction between them. This is done by methods such as Molecular Dynamics (MD) [4, 6] or the Discrete Element Method (DEM) [39, 125]. The challenge here is the sheer number of molecules. Even the calculation of only one millilitre of water (approx. 10^{22} molecules) exceeds the performance of all current computers, limiting the methods to very small domains.

Therefore in the last decades, a mesoscopic approach has been developed, combining the advantages of both, the microscopic and the macroscopic worlds. Members of this class are the LBMs. They are based on the BE, which models the evolution of a particle density distribution. This function describes the amount of fluid particles moving in an infinitesimal small period of time and an infinitesimal small domain, while moving in a certain direction. Hence it describes a time, space and velocity averaged state of the fluid

molecules. The BE can be solved by the LBM. The computational complexity of this is significantly lower than the microscopic approach and is approximately equal to that of the FEM or FVM, but has the advantage that it can exploit current parallel microprocessor architectures significantly better, albeit its algorithm is less complex. This will become even more important in the future, as architectures are still changing.

The three different scales form a hierarchy in the sense, that smaller scales are more general than coarser scales. The macroscopic properties velocity and pressure can be computed by the moments of the mesoscopic distribution function, which itself can be computed as average of the microscopic molecular trajectories. However, it is neither possible to compute the (exact) molecular trajectories from the mesoscopic distribution function nor from the macroscopic fluid's velocity.

A rigorous derivation of the mesoscopic Boltzmann equations from microscopic Newtonian particles interacting via a short-range potential can be found in Gallagher, Saint-Raymond and Texier [64]. A formal mathematical transition from the mesoscopic BE to the macroscopic equations (Stokes equations, compressible and incompressible NSEs) can be found in Bardos, Golse and Levermore [10, 11].

2.1.1 *Macroscopic: Navier-Stokes Equations*

Claude-Louis Navier *(1785–1836), French engineer and physicist*

Sir George Gabriel Stokes ℭ *(1819–1903) mathematician, physicist, politician and theologian*

At macroscopic scale, a fluid is seen as a continuous medium, i.e. its molecular structure and interactions are neglected. It is assumed to fill the entire domain of interest and the focus lies on its properties, such as pressure, velocity and temperature. The relations connecting these properties have mostly been developed by observation. In the following we will only discuss *incompressible Newtonian* fluids, which are governed by the *incompressible* NSE. The incompressible approximation is generally valid for liquids and gases with velocity less then 30% of the speed of sound in the respective material [118]. The following derivation strongly follows the lecture notes by Rannacher [151]. We start by stating the following theorem by Reynolds [153].

Osborne Reynolds *(1842–1912) British physicist, the Reynolds number* Re *is named after him*

THEOREM 1 (Reynolds transport theorem). *Let* $\phi : \Omega \times I \to \mathbb{R}$ *sufficiently smooth. Then for each material volume* $V(t) \in \mathbb{R}^d$ *it holds, that*

$$\frac{d}{dt}\int_{V(t)} \phi(\underline{x},t)\ d\underline{x} = \int_{V(t)} \partial_t\phi(\underline{x},t) + \nabla\cdot\left(\Phi\underline{u}^F(\underline{x},t)\right)\ d\underline{x}\,. \tag{2.1}$$

Here $d/dt = \partial_t + \underline{u}^F\nabla$ is the total derivative with respect to time and $\partial_t = \partial/\partial t$ as well as $\nabla = \left(\partial_{x_1},\dots,\partial_{x_d}\right) = (\partial/\partial\underline{x}_1,\cdots,\partial/\partial\underline{x}_d)$ are the partial derivatives with respect to time and position space. We investigate the dynamics of a fluid in a domain $\Omega \subset \mathbb{R}^d,\ d \in \{2,3\}$ over a given time interval $I = [0,T) \subset \mathbb{R}$. The quantities under consideration are the fluid velocity $\underline{u}^F : \Omega \times I \to \mathbb{R}^d$ and pressure $p : \Omega \times I \to \mathbb{R}$.

But let us begin with the mass density $\rho^F : \Omega \times I \to \mathbb{R}$, given within a volume $V = V(t)$ we obtain the contained mass as

$$m(t) = \int_{V(t)} \rho^F(\underline{x},t)\ d\underline{x}\,.$$

The first empirical observation is, that the contained mass is preserved over time

$$\frac{d}{dt}\int_{V(t)} \rho^F(\underline{x},t)\ d\underline{x} = 0\,.$$

Using Reynolds transport theorem for $\phi = \rho^F$ one gets

$$\int_{V(t)} \partial_t \rho^F + \nabla \cdot \left(\rho^F \underline{u}^F \right) \, d\underline{x} = 0 \,,$$

which holds for arbitrary volumes V. Following well-known arguments integration can be dropped and we obtain the *continuity equation* for fluid dynamics

$$\partial_t \rho^F + \nabla \cdot \left(\rho^F \underline{u}^F \right) = 0 \qquad \text{in } \Omega \times I \,. \tag{2.2}$$

Considering only incompressible fluids ($\rho^F = \text{const}$) this reduces to

$$\nabla \cdot \underline{u}^F = 0 \qquad \text{in } \Omega \times I \,. \tag{2.3}$$

Sir Isaac Newton ☾ *(1642–1726) English physicist and mathematician, author of "Philosophiæ Naturalis Principia Mathematica"*

The second major observation is *conservation of momentum*. According to Newton's second law of motion, the time derivative of linear momentum of a mass volume equals the forces exerted on the same volume

$$\frac{d}{dt} \int_V \rho^F \underline{u}^F \, d\underline{x} = \underline{F} \,.$$

The force $\underline{F}$ can be divided into body forces (gravity, electrostatic force) and surface forces (pressure and stress)

$$\frac{d}{dt} \int_V \rho^F \underline{u}^F \, d\underline{x} = \int_V \rho^F \underline{f} \, d\underline{x} + \int_{\partial V} \underline{n} \cdot \underline{\underline{\sigma}} \, do \,,$$

where $f : \Omega \times I \to \mathbb{R}^d$ is the density of volume force, $\underline{n}$ the unit outer normal on the boundary $\partial\Omega$ and $\underline{\underline{\sigma}} : \Omega \times I \to \mathbb{R}^{(d \times d)}$ the *cauchy stress tensor*. Using Gauss theorem we can transform the surface integral into a volume integral

Carl Friedrich Gauß ☾ *(1777–1855) German mathematician, discoverer of the normal distribution and face of the 10DM bill*

$$\frac{d}{dt} \int_V \rho^F \underline{u}^F \, d\underline{x} = \int_V \rho^F \underline{f} + \nabla \cdot \underline{\underline{\sigma}} \, d\underline{x} \,.$$

On the other hand using Reynold's theorem for $\phi = \rho^F \underline{u}^F$ we see

$$\frac{d}{dt} \int_V \rho^F \underline{u}^F \, d\underline{x} = \int_V \partial_t \left(\rho^F \underline{u}^F \right) + \nabla \cdot \left(\rho^F \underline{u}^F \otimes \underline{u}^F \right) \, d\underline{x} \,,$$

using the outer product of vectors $\underline{u}^F \otimes \underline{u}^F = (u_i u_j)_{i,j=1}^d$. Therefore

$$\partial_t (\rho^F \underline{u}^F) + \nabla \cdot \left(\rho^F \underline{u}^F \otimes \underline{u}^F \right) = \rho^F \underline{f} + \nabla \cdot \underline{\underline{\sigma}}$$

and expanding the partial derivatives

$$\begin{aligned} \partial_t \left(\rho^F \underline{u}^F \right) &= \rho^F \partial_t \underline{u}^F + \underline{u}^F \partial_t \rho^F, \\ \nabla \cdot \left(\rho^F \underline{u}^F \otimes \underline{u}^F \right) &= \underline{u}^F \nabla \cdot \left(\rho^F \underline{u}^F \right) + \rho^F \underline{u}^F \cdot \nabla \underline{u}^F \,, \end{aligned}$$

one obtains

$$\rho^F \partial_t \underline{u}^F + \underline{u}^F \left(\partial_t \rho^F + \nabla \cdot \left(\rho^F \underline{u}^F \right) \right) + \rho^F \underline{u}^F \cdot \nabla \underline{u}^F = \rho^F \underline{f} + \nabla \cdot \underline{\underline{\sigma}} \,.$$

Keeping Eq. (2.2) in mind this results in the *Cauchy momentum equation* in its non-conservative form

Baron Augustin-Louis Cauchy ☾ *(1789–1857) French mathematician, pioneer of analysis*

$$\rho^F \partial_t \underline{u}^F + \rho^F \left(\underline{u}^F \cdot \nabla \right) \underline{u}^F = \rho^F \underline{f} + \nabla \cdot \underline{\underline{\sigma}} \,. \tag{2.4}$$

For a volume $V = V(t)$ the angular momentum is defined by

$$\underline{L}(V) := \int_V \underline{x} \times \left(\rho^F \underline{u}^F\right) d\underline{x}$$

and the torque by

$$\underline{D}(V) := \int_V \underline{x} \times \left(\rho^F \underline{f}\right) d\underline{x} + \int_{\partial V} \underline{x} \times \left(\underline{n} \cdot \underline{\underline{\sigma}}\right) d\underline{o} \,, \tag{2.5}$$

where $\underline{a} \times \underline{b}, \quad \underline{a}, \underline{b} \in \mathbb{R}^3$ is the cross product. The *conservation of angular momentum* states that the temporal change in angular momentum equals the applied torque

$$\partial_t \underline{L}(V) = \underline{D}(V) \,,$$

and is the rotational equivalent to the conservation of momentum. Using Reynold's transport theorem for $\Phi_i = \rho^F \epsilon_{ijk} x_j u_k$ $i, j, k \in 1, 2, 3$ and Einstein summation convention we find

Albert Einstein [C] *(1879–1955) German-born theoretical physicist, introduced Einstein notation, where a double index variable implies summation*

$$d_t \int_V \rho^F \epsilon_{ijk} x_j u_k d\underline{x} = \int_V \partial_t \left(\rho^F \epsilon_{ijk} x_j u_k\right) + \nabla \cdot \left(\rho^F \epsilon_{ijk} x_j u_k \underline{u}^F\right) d\underline{x} \,,$$

where x_j and u_k are the components of $\underline{x}$ and $\underline{u}^F$ and ϵ_{ijk} is the Levi–Civita symbol, which is defined as

$$\epsilon_{ijk} := \begin{cases} 1, & \text{even} \\ -1, & \text{odd permutation of } \{1,2,3\} \\ 0 & \text{no} \end{cases} .$$

With the derivatives

$$\partial_t \left(\rho^F \epsilon_{ijk} x_j u_k\right) = \epsilon_{ijk} x_j v_k \partial_t \rho^F + \epsilon_{ijk} x_j \partial_t v_k$$

and

$$\begin{aligned} \nabla \cdot \left(\rho^F \epsilon_{ijk} x_j u_k \underline{u}^F\right) &= \partial_l \left(\rho^F \epsilon_{ijk} x_j u_k u_l\right) \\ &= \epsilon_{ijk} x_j u_k \nabla \cdot \left(\rho^F \underline{u}^F\right) \\ &\quad + \rho^F \epsilon_{ijk} x_j u_l \partial_l u_k + \rho^F \epsilon_{ijk} u_j u_k \,. \end{aligned}$$

and since $\epsilon_{ijk} a_j b_k = (\underline{a} \times \underline{b})_i$ and therefore $\epsilon_{ijk} u_j u_k = 0$ one obtains

$$\begin{aligned} \frac{d}{dt} \underline{L}(V) = \int_{V(t)} &\left(\underline{x} \times \underline{u}^F \partial_t \rho^F + \rho^F \underline{x} \times \partial_t \underline{v} + \left(\underline{x} \times \underline{u}^F\right) \nabla \cdot \left(\rho^F \underline{u}^F\right)\right. \\ &\left. + \rho^F \underline{x} \times \left(\underline{u}^F \cdot \nabla\right) \underline{u}^F\right) d\underline{x} \,. \end{aligned}$$

Using the continuity equation (2.2) one can simplify to

$$\frac{d}{dt} \underline{L}(V) = \int_V \underline{x} \times \left(\rho^F \partial_t \underline{u}^F + \rho^F \left(\underline{u}^F \cdot \nabla\right) \underline{u}^F\right) d\underline{x} \,.$$

Using Gauss theorem on the right addend of the right hand side of (2.5)

$$\begin{aligned} (D(V))_i &= \int_V \left(\underline{x} \times \left(\rho^F \underline{f}\right)\right)_i d\underline{x} + \int_{\partial V} \left(\underline{x} \times \left(\underline{n} \cdot \underline{\underline{\sigma}}\right)\right)_i d\underline{o} \\ &= \int_V \left(\underline{x} \times \left(\rho^F \underline{f}\right)\right)_i d\underline{x} + \int_{\partial V} \epsilon_{ijk} x_i n_l \sigma_{lk} d\underline{o} \\ &= \int_V \left(\underline{x} \times \left(\rho^F \underline{f}\right)\right)_i d\underline{x} + \int_V \epsilon_{ijk} \sigma_{jk} + \epsilon_{ijk} \partial_l \left(x_j \sigma_{lk}\right) d\underline{x} \\ &= \int_V \left(\underline{x} \times \left(\rho^F \underline{f} + \nabla \cdot \underline{\underline{\sigma}}\right)\right)_i + \left(\underline{\underline{\epsilon}} : \underline{\underline{\sigma}}\right)_i d\underline{x} \end{aligned}$$

which leads to

$$\int_V \left(\underline{x} \times \left(\rho^F \partial_t \underline{u}^F + \rho^F \left(\underline{u}^F \cdot \nabla\right) \underline{u}^F - \rho^F \underline{f} - \nabla \cdot \underline{\underline{\sigma}}\right)\right) d\underline{x} = \int_V \underline{x} \times \left(\underline{\underline{\epsilon}} : \underline{\underline{\sigma}}\right) d\underline{x} \,.$$

Keeping in mind the momentum equation (2.4) this shows

$$0 = \int_V \underline{x} \times \left(\underline{\underline{\epsilon}} : \underline{\underline{\sigma}}\right) d\underline{x}$$

and eventually the symmetry of the stress tensor ($\sigma_{ij} = \sigma_{ji}$). The derived equations of state arose from general principles of conservation and are valid for almost all fluids and gases. In the following we will state some restrictions by relating $\underline{\underline{\sigma}}$ to the flow variables according to properties of the respective material, resulting in the NSE for incompressible Newtonian fluids.

For Stokes fluids the stress tensor takes a spherically symmetric form in rest i.e. only normal stresses appear

$$\underline{\underline{\sigma}}|_{\underline{u}^F = 0} = -p \, \underline{\underline{Id}} \,,$$

with the scalar hydro-static pressure $p : \Omega \times I \to \mathbb{R}$ and the identity matrix $\underline{\underline{Id}} \in \mathbb{R}^{(d \times d)}$. For a moving fluid the (symmetric) *deviatoric stress* tensor $\underline{\underline{\tau}} \in \mathbb{R}^{(d \times d)}$ is added

$$\underline{\underline{\sigma}} = -p \, \underline{\underline{Id}} + \underline{\underline{\tau}} \,.$$

Usually $\underline{\underline{\tau}}$ is described by a constitutive equation $\underline{\underline{\tau}} = F(\underline{\underline{S}})$, where $\underline{\underline{S}} = \frac{1}{2}(\nabla \underline{u}^F + (\nabla \underline{u}^F)^\top)$ is the *strain rate* tensor. It is assumed that $\operatorname{tr}(\underline{\underline{\tau}}) := \tau_{ii} = 0$, otherwise $\operatorname{tr}(\underline{\underline{\tau}})$ can be accumulated to the pressure. This implies that $p = -\frac{1}{3}\sigma_{ii}$. Fluids, that obey the constitutive equation $F(\underline{\underline{S}}) = 2\mu \underline{\underline{S}} + \lambda \ \operatorname{tr}(\underline{\underline{S}}) Id$ are called *Newtonian* fluids (for a derivation see e.g. [13]). Here $\lambda \in \mathbb{R}_0^+$ and $\mu \in \mathbb{R}_0^+$ are material dependent parameters. λ is the first *Lamé constant* or *volume viscosity* and μ is the second *Lamé constant* or *dynamic viscosity*. As $\underline{\underline{\tau}}$ makes zero contribution to the mean normal stress,

Gabriel Lamé [1795–1870] *French physicist and mathematician*

$$\operatorname{tr}(\underline{\underline{\tau}}) = 2\mu \partial_i u_i + 3\lambda \partial_i u_i = 0 \,,$$

which implies that

$$2\mu + 3\lambda = 0 \,.$$

Hence

$$\underline{\underline{\sigma}} = -p \, \underline{\underline{Id}} + \mu\left(\nabla \underline{u}^F + (\nabla \underline{u}^F)^T\right) - \frac{2}{3}\mu\left(\nabla \cdot \underline{u}^F\right)\underline{\underline{Id}}$$

holds. Inserting the last equation into (2.4) and assuming μ to be constant one obtains the *compressible* NSE equations:

$$\partial_t \rho^F + \nabla \cdot (\rho^F \underline{u}^F) = 0 \qquad \text{in } \Omega \times I \,, \tag{2.6a}$$

$$\rho^F \partial_t \underline{u}^F + \rho^F (\underline{u}^F \cdot \nabla) \underline{u}^F = \rho^F \underline{f} - \nabla p + \mu \Delta \underline{u}^F + \frac{1}{3}\mu \nabla (\nabla \cdot \underline{u}^F) \qquad \text{in } \Omega \times I \,. \tag{2.6b}$$

Gases at velocities less then 30% of their speed of sound and liquids in general can be assumed to be incompressible [118]. Therefore keeping ρ^F constant over time and space reveals the NSE for *incompressible* fluids:

$$\nabla \cdot \underline{u}^F = 0 \qquad \text{in } \Omega \times I \,, \tag{2.7a}$$

$$\partial_t \underline{u}^F + (\underline{u}^F \cdot \nabla) \underline{u}^F = \nu \Delta \underline{u}^F - \frac{1}{\rho^F} \nabla p + \underline{f} \qquad \text{in } \Omega \times I \,. \tag{2.7b}$$

Here $\nu := \mu/\rho^F$ is the *kinematic viscosity*. The units of Equation (2.7b) are the units of acceleration $[m/s^2]$. The left hand side contains the fluid accelerations due to inertia, while the right hand side contains acceleration due to friction and external forces.

Often the NSE are written in dimensionless notation. This can be done by introducing the *characteristic length* $L > 0$ and *characteristic velocity* $U > 0$ and defining the dimensionless quantities

$$\text{position } \underline{x}^* := \frac{\underline{x}}{L}\,, \qquad \text{velocity } \underline{u}^{F*} := \frac{\underline{u}^F}{U}\,,$$
$$\text{time } t^* := \frac{t}{L/U}\,, \qquad \text{pressure } p^* := \frac{1}{U^2}\frac{p}{\rho^F}\,,$$

and dimensionless operators

$$d_t^* := \frac{d}{dt^*}\,, \qquad \nabla^* = \frac{d}{d\underline{x}^*}\,.$$

Substituting the scales and differential operators one obtains the *dimensionless* NSEs

$$\nabla^* \cdot \underline{u}^{F*} = 0 \qquad \text{in } \Omega \times I\,, \tag{2.8a}$$

$$\partial_{t^*}\underline{u}^{F*} + (\underline{u}^{F*} \cdot \nabla^*)\underline{u}^{F*} = \frac{1}{Re}\Delta^*\underline{u}^{F*} - \nabla^* p^* + \frac{L}{U^2\rho^F}\underline{f} \qquad \text{in } \Omega \times I\,, \tag{2.8b}$$

with the *Reynolds number* $Re = \frac{UL}{\nu}$. The Reynolds number compares inertial and viscous forces and is used to characterise the flow behaviour of the system. In low Reynolds number flows viscous forces dominate which results in a more laminar flow, while in high Reynolds number flows the inertial forces dominate, leading to more turbulent and chaotic behaviour. If $\underline{f}$ denotes gravitational acceleration $\underline{f} = \underline{g}$ the force term can be substituted by the Froude number $Fr = U^2/Lg$, which denotes the ration of inertial and gravitational forces.

William Froude
(1810–1879) English engineer

For small Reynolds numbers the viscous term becomes more significant and the inertia terms can be neglected. If no external forces exist, one obtains the incompressible, time dependent, dimensionless *Stokes equations*

$$\nabla^* \cdot \underline{u}^{F*} = 0 \qquad \text{in } \Omega \times I\,, \tag{2.9a}$$

$$\frac{1}{Re}\Delta^*\underline{u}^{F*} - \nabla^* p^* = 0 \qquad \text{in } \Omega \times I\,. \tag{2.9b}$$

2.1.2 *Microscopic: Newton's Law*

Looking closely at an assumed continuous fluid one finds that it is constituted of microscopic atoms or molecules. Assuming that these atoms are all spherical in shape, of the same mass $M > 0$ and radius $R > 0$ and obey the classic laws of mechanics, therefore neglecting quantum mechanic effects, one could try to simulate their time dependent motion and draw conclusions as to the characteristics of the composed fluid. Bridging the gap between the atomic structure of matter and its continuum-like behaviour

at a macroscopic level is a basic problem of statistical mechanics. For mass points the equations of motion are

$$\frac{d}{dt}\underline{U}(t) = \frac{\underline{F}(t)}{M} \qquad \text{in } I\,, \tag{2.10}$$

$$\frac{d}{dt}\underline{X}(t) = \underline{U}(t) \qquad \text{in } I\,, \tag{2.11}$$

where $\underline{U} : I \to \mathbb{R}^d$ is the molecule velocity and $\underline{x} : I \to \mathbb{R}^d$ the molecule position. Furthermore, $\underline{F} : I \to \mathbb{R}^d$ is a force acting on a particle which is usually a sum of several forces i. a. particle interaction. For $N \in \mathbb{N}$ molecules this leads to $6N$ differential equations in $6N$ unknowns and $6N$ initial conditions. This is problematic for at least two reasons: Firstly, water as a common fluid has a molar mass of 18.01528 g/mol, which leads to approximately $3.35 \cdot 10^{22}$ molecules in one cubic centimetre. Therefore even on current computers this is a limitation to extremely small domains. Secondly, according to Heisenberg's uncertainty principle complementary variables, such as velocity and position can not be identified simultaneously. Hence it is impossible to find precise initial conditions for this model.

Werner Karl Heisenberg *(1901–1976) German physicist, holds an honorary doctorate of KIT*

2.2 BOLTZMANN EQUATION

Instead of trying to characterise single atoms or molecules, in mesoscopic modelling one concentrates on the probabilistic average behaviour of a set of atoms in a small volume in phase space which is assumed to be large in comparison to the size of single atoms and small in comparison to the characteristic length of the observed system. The formal connection of the microscopic and mesoscopic world is known as the Boltzmann–Grad limit [70]. In it, formally the particle number N tends to infinity, the particle mass m and radius R tend to 0, while Nm and NR^2 remain finite and NR^3 tends to 0.

Ludwig Boltzmann ✆ *(1844–1906) Austrian physicist and philosopher*

Harold Grad *(1923–1986) American applied mathematician*

The single particle distribution function $f : \Omega \times \Xi \times I \to \mathbb{R}^+$ describes the particle mass density of a fluid depending on position $\underline{x} \in \Omega$, microscopic velocity $\underline{\xi} \in \Xi$ and time $t \in I = [0, T)$. Usually the combination of the domain $\Omega \subset \mathbb{R}^d$ and velocity space $\Xi \subset \mathbb{R}^d$ is called the phase space $\Omega \times \Xi$. As a first interpretation $f(\underline{x}, \underline{\xi}, t)\,d\underline{x}\,d\underline{\xi}$ gives the mass of particles contained in an infinitesimal volume $d\underline{x}$ centred at $\underline{x}$ having a velocity in an infinitesimal small velocity space volume $d\underline{\xi}$ centred around $\underline{\xi}$ at time t. Therefore, f is often called a (mass) density distribution function in literature [9, 172].

We define the moments $\mathcal{M}_\chi : \Omega \times I \to \mathbb{R}^n, n \in \mathbb{N}$ of f as

$$\mathcal{M}_\chi(\underline{x}, t) = \int_\Xi \chi(\underline{\xi}) f(\underline{x}, \underline{\xi}, t)\; d\underline{\xi}\,.$$

For specific values of χ one obtains the macroscopic quantities of interest:

- $\chi = 1$, mass density ρ^F

$$\rho^F(\underline{x}, t) = \int_\Xi f(\underline{x}, \underline{\xi}, t)\; d\underline{\xi}\,, \tag{2.12}$$

- $\chi = \underline{\xi}$, velocity $\underline{u}^F$

$$\rho^F(\underline{x}, t)\underline{u}^F(\underline{x}, t) = \int_\Xi \xi f(\underline{x}, \underline{\xi}, t)\; d\underline{\xi}\,, \tag{2.13}$$

- $\chi = \left\|\underline{\xi} - \underline{u}(\underline{x}, t)\right\|_2^2$, temperature T

$$\frac{3k_B}{m}\rho^F(\underline{x}, t)T(\underline{x}, t) = \int_\Xi \left\|\underline{\xi} - \underline{u}(\underline{x}, t)\right\|_2^2 f(\underline{x}, \underline{\xi}, t)\, d\underline{\xi} \,, \tag{2.14}$$

where in the last equations k_B denotes the Boltzmann constant.

The evolution of the particle distribution function is described by the Boltzmann Equation (BE)

$$\partial_t f + \underline{\xi} \cdot \nabla_{\underline{x}} f + \underline{a} \cdot \nabla_{\underline{\xi}} f = J(f) \qquad \text{in } \Omega \times \Xi \times I \,, \tag{2.15}$$

where ∂_t, $\nabla_{\underline{x}}$ and $\nabla_{\underline{\xi}}$ are the partial derivatives with respect to time, space and microscopic velocity. The BE is a balance equation, that relates changes in f due to transportation to changes due to collisions

$$\left.\frac{Df}{Dt}\right|_{\text{transport}} = \left.\frac{Df}{Dt}\right|_{\text{collision}} .$$

The acceleration $\underline{a} : \Omega \times I \to \mathbb{R}^d$ caused by a body force $\underline{F} = M\underline{a}$ acting on the fluid particles, is assumed to be equal to zero for the time being. The right hand side $J(F)$ of Equation (2.15) is called collision term and contains changes in f due to interactions between particles.

For the original Boltzmann collision operator it is assumed, that collisions involving more than two particles are negligible [14] and that the collisions between pairs of particles are uncorrelated [14]. The collision term can be written as [189]

$$J(f) = \int_\Xi \int_S B(\underline{\theta}, \underline{\xi}, \underline{\xi}_1)(f' f_1' - f f_1)\, d\underline{\theta} d\underline{\xi} \qquad \text{in } \Omega \times \Xi \times I \,, \tag{2.16}$$

with $f_1 = f(\underline{x}, \underline{\xi}_1, t), f' = f(\underline{x}, \underline{\xi}', t)$ and $f_1' = f(\underline{x}, \underline{\xi}_1', t)$. The domain of integration is the velocity space Ξ and the unit sphere S. The collision kernel $B(\underline{\theta}, \underline{\xi}, \underline{\xi}_1)$ cannot be expressed in terms of elementary functions [26]. However, it essentially holds the probability density that a particle moving with velocity $\underline{\xi}$ and interacts with a particle moving with velocity $\underline{\xi}'$, is deflected in direction $\underline{\theta} = (\underline{\xi}_1 - \underline{\xi})/\|\underline{\xi}_1 - \underline{\xi}\|_2$. It is assumed, that two particles interact by a perfectly elastic collision, that conserves mass and momentum. Therefore, the velocities after collision can be computed by

$$\underline{\xi}' = \underline{\xi} + \underline{\theta\theta}\,(\underline{\xi}_1 - \underline{\xi}) \,,$$
$$\underline{\xi}_1' = \underline{\xi}_1 + \underline{\theta\theta}\,(\underline{\xi}_1 - \underline{\xi}) \,.$$

Prabhu Lal Bhatnagar *(1912–1976) Indian mathematician*
Eugene P. Gross *(1926–1991) American physicist*
Max Krook *(1913–1985) American mathematician famous for their collision operator*

The collision operator can also be interpreted as a balance term for particles incoming into a infinitesimally small volume element $d\underline{x}$ due to collisions outside of $d\underline{x}$ and particles outgoing of $d\underline{x}$ due to collisions inside $d\underline{x}$ [74]. It is valid in this form for a dilute mono-atomic gas. A detailed derivation of the Boltzmann collision operator can be found in the literature [26, 74, 158].

2.2.1 *Collision Invariance and Equilibrium*

As the collision term is of a rather complex form, there exist several approximations that allow an easier numerical computation. Throughout this thesis the Bhatnagar–Gross–Krook (BGK) [16] Method is used. Besides multi-relaxation time schemes [46] are common. All of them have to maintain some important properties, which are shown next. The BGK approximation then follows naturally.

DEFINITION 1. A locally integrable function $\phi : \Xi \to \mathbb{R}$ is called collision invariant of the BE with collision operator $J(f)$, if for all functions f integrable in Ξ and integrable $\phi \cdot f$, it holds:

$$\int_{\Xi} \phi(\underline{\xi}) J(f)(\underline{\xi}) d\underline{\xi} = 0 \, .$$

The *elementary collision invariants* are

$$\phi_0(\underline{\xi}) = 1 \, , \qquad \phi_i(\underline{\xi}) = \underline{\xi}_i \, , i = 1, \ldots, 3 \, , \qquad \phi_4(\underline{\xi}) = \left\| \underline{\xi} \right\|_2^2 \, ,$$

and represent mass, momentum and kinetic energy. General collision invariants are linear combinations of the elementary collision invariants.

THEOREM 2. *Every continuous function* $\phi : \Xi \to \mathbb{R}$ *is a* collision invariant, *iff*

$$\phi(\underline{\xi}) = a + \underline{b}\underline{\xi} + c \left\| \underline{\xi} \right\|_2^2 \, , \tag{2.17}$$

for $a, c \in \mathbb{R}$ *and* $\underline{c} \in \Xi$.

Proof. See Babovsky [9, Satz 2.20] □

DEFINITION 2. A continuous and integrable function $f : \Omega \times \Xi \times I \to \mathbb{R}^+$, for which $\ln(f) \cdot J(f)$ is integrable in Ξ is called *equilibrium solution* of the BE if $J(f) = 0$.

By the definition above $\ln(f)$ is collision invariant and

$$\ln(f) = a + \underline{b}\xi + c\|\xi\|_2^2$$

holds. It can be shown, that all possible equilibrium solutions can be written in the form

$$M(\underline{x}, \underline{\xi}, t) = \frac{\rho^F(\underline{x}, t)}{(2\pi k_B T(\underline{x}, t))^{d/2}} \exp\left(- \frac{\left\| \underline{\xi} - \underline{u}^F(\underline{x}, t) \right\|_2^2}{2 k_B T(\underline{x}, t)} \right) \, , \tag{2.18}$$

where $M : \Omega \times \Xi \times I \to \mathbb{R}^+$ is only implicitly depending on the position $\underline{x}$ and the time t, we therefore also write $M[\rho^F, \underline{u}^F, T](\underline{\xi}) := M(\underline{x}, \underline{\xi}, t)$. Functions of this type are called *Maxwell functions* or Equilibrium Distribution Function (EDF). The fluid density ρ^F, fluid velocity $\underline{u}^F$ and fluid temperature T are connected to the function $M[\rho^F, \underline{u}^F, T]$ by

James Clerk Maxwell *(1831–1879)* ☾ *Scottish physicist, developed the electromagnetic equations*

- mass density:

$$\rho^F(\underline{x}, t) = \int_{\underline{\Xi}} M[\rho^F, \underline{u}^F, T](\underline{x}, \underline{\xi}, t) \, d\underline{\xi} \, , \tag{2.19}$$

- velocity:

$$\rho^F(\underline{x}, t) \underline{u}^F(\underline{x}, t) = \int_{\Xi} \underline{\xi} M[\rho^F, \underline{u}^F, T](\underline{x}, \underline{\xi}, t) \, d\underline{\xi} \, , \tag{2.20}$$

- temperature:

$$\rho^F(\underline{x}, t) T(\underline{x}, t) = \frac{1}{3k_B} \int_{\Xi} \left\| \underline{\xi} - \underline{u}(\underline{x}, t) \right\|_2^2 M[\rho^F, \underline{u}^F, T](\underline{x}, \underline{\xi}, t) \, d\underline{\xi} \, . \tag{2.21}$$

Unfortunately the Maxwell distribution is not a general solution of the BE as only the right hand side $J(M)$ is equal to 0, but the left hand side $D_t M$ usually is not. However, we will show that for undisturbed fluids the distribution function f always tends towards an equilibrium solution. We therefore introduce the quantity H as

$$H(\underline{x}, t) = \int_{\Xi} f(\underline{x}, \underline{\xi}, t) \ln f(\underline{x}, \underline{\xi}, t) \, d\underline{\xi} \, ,$$

which is connected to negative entropy and state the

THEOREM 3 (H-theorem). *Providing f satisfies (2.15) for $\underline{F} = 0$ and $\ln(f)J(f)$ is integrable in Ξ then the following holds*

1. *$H(\underline{x}, t)$ is monotonically decreasing in t.*
2. *$H(\underline{x}, t)$ is constant in t iff f is a Maxwell distribution.*

Proof. A physical interpretation of this theorem is, that the entropy of a closed system is continuously increasing with time and only keeps a certain level at the time it reaches an equilibrium state. We show a simplified proof of part one and a closed system, in which spatial change is zero and no external forces are applied. In this situation Boltzmann's Equation (2.15) takes the form

$$\partial_t f = J(f) \, . \tag{2.22}$$

The time derivative of H is

$$\partial_t H = \int_{\Xi} \partial_t (f \ln f) \, d\underline{\xi} = \int_{\Xi} (1 + \ln f) \partial_t f \, d\underline{\xi} \, .$$

Substituting $\partial_t f$ by (2.22) leads to

$$\partial_t H = \int_{\Xi} (1 + \ln f) J(f) \, d\underline{\xi} \, .$$

As the primary and inverse particle collisions, as well as the participating particles are indistinguishable, the functions f, f_1, f', and f'_1 are interchangeable in $J(f)$, without changing $|J(f)|$. Summation of $J(f) + J(f_1) + J(f') + J(f'_1)$ and the fact that 1 is a collision invariant, allows to make a statement concerning the sign of $\partial_t H$,

$$\partial_t H = \frac{1}{4} \int_{\Xi} \int_{\Xi} \int_{S} B(\underline{\theta}, \underline{\xi}, \underline{\xi}_1)(f' f'_1 - f f_1) \ln \left(\frac{f' f'_1}{f f_1} \right) \, d\theta \, d\underline{\xi}_1 \, d\underline{\xi} \leqslant 0 \, .$$

The inequality can be easily seen, as $(x - y) \ln(\frac{x}{y}) \leqslant 0$. A more detailed and more general proof can again be found in Cercignani [27]. □

A consequence of the H-Theorem is, that from reversible microscopic particle collisions an irreversible process evolves. This was surprising and was a source of heavy discussions. Another results is, that the particle distribution of an undisturbed fluid statistically tends towards the well defined equilibrium distribution (2.18), only depending on the local density, velocity and temperature of the fluid. This leads to the introduction of a simplified collision term.

2.2.2 BGK *Collision Operator*

With the knowledge of the last section, in particular that the particle distribution tends towards an equilibrium distribution, Bhatnagar, Gross and Krook [16] proposed a simplified approximation of the collision term $J(f)$. The so-called Bhatnagar–Gross–Krook (BGK) *collision operator* is defined by

$$Q(f) = -\frac{1}{\tau^*}(f - M) \qquad \text{in } \Omega \times \Xi \times I\,, \tag{2.23}$$

with a Maxwellian $M := M[\rho^F, \underline{u}^f, T](\underline{\xi})$ and the *mean free time* between collisions $\tau^* \in \mathbb{R}$. The BGK collision operator $Q(f)$ has the same collision invariants as the traditional collision operator $J(F)$. For $\phi(\underline{\xi})$ as in (2.17) this can be shown by

$$\begin{aligned}
\int_\Xi \phi(\underline{\xi}) Q(f) d\underline{\xi} &= -\frac{1}{\tau^*} \int_\Xi \phi(\underline{\xi})\,(f - M)\ d\underline{\xi} \\
&= -\frac{1}{\tau^*} \int_\Xi \left(a + \underline{b}\underline{\xi} + c\left\|\underline{\xi}\right\|_2^2\right)(f - M)\ d\underline{\xi} \\
&= -\frac{1}{\tau^*}\left(a\left(\int_\Xi f\, d\xi - \int_\Xi M\, d\xi\right) + \dots\right) \\
&= -\frac{1}{\tau^*}\left(a\left(\underline{u}^F - \underline{u}^F\right) + \dots\right) \\
&= 0\,.
\end{aligned}$$

using Eqs. (2.12)–(2.14) and Eqs. (2.19)–(2.21). Krause [110] states that the homogeneous BGK-BE also satisfies the H-Theorem. The BE with BGK collision term is called *BGK–Boltzmann Equation (BE)*

$$\partial_t f + \underline{\xi} \cdot \nabla_{\underline{x}} f = \frac{1}{\tau^*}(f - M) \qquad \text{in } \Omega \times \Xi \times I\,. \tag{2.24}$$

2.2.3 *Discretisation of the Boltzmann Equation*

The discretisation of the BGK-BE and derivation of the Lattice Boltzmann Method (LBM) follows roughly the description in [110, Sec 1.3-2.1]. Other discretisations that obtain the same result can be found in Abe [1] and He and Luo [78].

Introducing a discretisation parameter $h \in \mathbb{R}^+$ we define the *speed of sound*

$$c_s = \sqrt{3 k_B T} := \frac{1}{h}\,. \tag{2.25}$$

In most literature ,e. g. He and Luo [78], the speed of sound is defined as the ratio of the spatial step and the time step $c_S = \Delta_x/\Delta_t$. By choosing $c_S = 1/h$ we implicitly assume diffusive scaling, where the time step is proportional to the square of the spatial step. The *mean free time* is set to

$$\tau^* = 3\nu h^2\,,$$

where $\nu > 0$ is the kinematic viscosity. Inserting Equation (2.25) into Equation (2.18) the Maxwellian distribution becomes independent of the temperature T

$$M^*[\rho^F, \underline{u}^F](\underline{\xi}) = \rho^F \left(\frac{3}{2\pi}\right)^{d/2} h^d \exp\left(-\frac{3}{2}(h\underline{\xi} - h\underline{u})^2\right)\,, \tag{2.26}$$

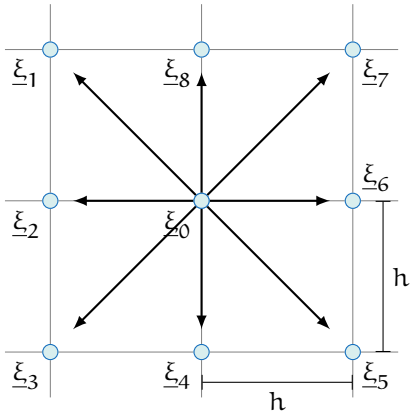


Figure 2.2: Illustration of a *D2Q9* lattice.

where $d \in \{2, 3\}$ is the space dimension. The BGK-Boltzmann Equation results in

$$h^2 \frac{d}{dt} f = -\frac{1}{3\nu} (f - M^*) \qquad \text{in } \Omega \times \Xi \times I \,. \tag{2.27}$$

The phase space $\Omega \times \Xi$ is also discretised using the parameter h. The domain Ω is approximated by a set of equidistant *nodes* with spacing h. The resulting discrete set of points is called the *lattice* Ω_h. The velocity space is reduced to a small number $q \in \mathbb{N}$ of selected velocities $\underline{\xi}_i, i = 0, \dots, q-1$. The discrete velocity space is denoted by $\Xi_h := \{\underline{\xi}_i \in \Xi : i = 0, \dots, q-1\}$. The time interval I is discretised by $I_h := \{t \in I : t = h^2 k, k \in \mathbb{N}\}$. Individual LBM are denoted by *DdQq*, with common models being *D2Q9* and *D3Q19*. For the *D2Q9* the discrete velocities are given by (see also Figure 2.2)

$$\begin{aligned}
&\underline{\xi}_0 = (0,0)\,, \\
&\underline{\xi}_1 = \tfrac{1}{h}(-1,1)\,, \quad \underline{\xi}_2 = \tfrac{1}{h}(-1,0)\,, \\
&\underline{\xi}_3 = \tfrac{1}{h}(-1,-1)\,, \quad \underline{\xi}_4 = \tfrac{1}{h}(0,-1)\,, \\
&\underline{\xi}_5 = \tfrac{1}{h}(1,-1)\,, \quad \underline{\xi}_6 = \tfrac{1}{h}(1,0)\,, \\
&\underline{\xi}_7 = \tfrac{1}{h}(1,1)\,, \quad \underline{\xi}_8 = \tfrac{1}{h}(0,1)\,,
\end{aligned}$$

and for the *D3Q19* model the velocities are (see also Figure 2.3)

$$\begin{aligned}
&\underline{\xi}_0 = (0,0,0) \\
&\underline{\xi}_1 = \tfrac{1}{h}(0,-1,0)\,, \quad \underline{\xi}_2 = \tfrac{1}{h}(-1,0,0)\,, \quad \underline{\xi}_3 = \tfrac{1}{h}(0,0,-1)\,, \\
&\underline{\xi}_4 = \tfrac{1}{h}(-1,-1,0)\,, \quad \underline{\xi}_5 = \tfrac{1}{h}(-1,1,0)\,, \quad \underline{\xi}_6 = \tfrac{1}{h}(-1,0,-1)\,, \\
&\underline{\xi}_7 = \tfrac{1}{h}(-1,0,1)\,, \quad \underline{\xi}_8 = \tfrac{1}{h}(0,-1,-1)\,, \quad \underline{\xi}_9 = \tfrac{1}{h}(0,-1,1)\,, \\
&\underline{\xi}_{10} = \tfrac{1}{h}(1,0,0)\,, \quad \underline{\xi}_{11} = \tfrac{1}{h}(0,1,0)\,, \quad \underline{\xi}_{12} = \tfrac{1}{h}(0,0,1)\,, \\
&\underline{\xi}_{13} = \tfrac{1}{h}(1,1,0)\,, \quad \underline{\xi}_{14} = \tfrac{1}{h}(1,-1,0)\,, \quad \underline{\xi}_{15} = \tfrac{1}{h}(1,0,1)\,, \\
&\underline{\xi}_{16} = \tfrac{1}{h}(1,0,-1)\,, \quad \underline{\xi}_{17} = \tfrac{1}{h}(0,1,1)\,, \quad \underline{\xi}_{18} = \tfrac{1}{h}(0,1,-1)\,.
\end{aligned}$$

Brook Taylor ☾ *(1685–1731) British mathematician*

The numbering is in accordance to the numbering used in `OpenLB`. One can see, that the discrete velocities are chosen such that an imaginary fluid particle can only move from one lattice node to an immediate neighbour within one timestep or remain on its current node. To obtain velocities independent of h we define $\widetilde{\underline{\xi}}_i := \underline{\xi}_i h$. Assuming a solution f_h of (2.27) with moments

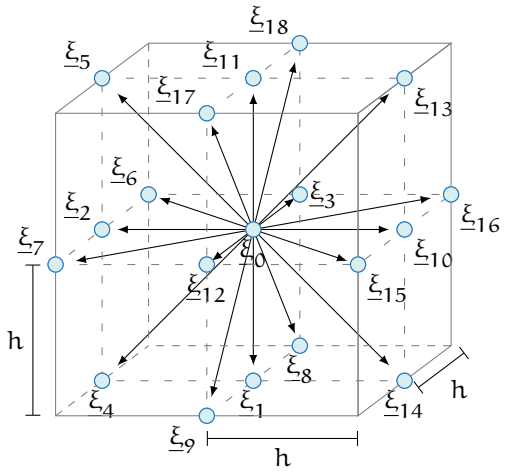


Figure 2.3: Illustration of a *D3Q19* lattice.

ρ_h^F and $\underline{u}_h^F$, the temperature independent Maxwellian $M^* := M^*[\rho, \underline{u}^F]$ can be further expanded in a Taylor series

$$\begin{aligned}
M^* &= \rho^F \left(\frac{3}{2\pi}\right)^{\frac{d}{2}} h^d \exp\left(-\frac{3}{2}(\underline{\tilde{\xi}}_i^2 - h\underline{u})^2\right) \\
&= \rho^F \left(\frac{3}{2\pi}\right)^{\frac{d}{2}} h^d \exp\left(\frac{3}{2}\underline{\tilde{\xi}}_i^2\right) \exp\left(3h\underline{\tilde{\xi}}_i \cdot \underline{u}^F - \frac{3}{2}\left(h\underline{u}^F\right)^2\right) \\
&= \rho^F \left(\frac{3}{2\pi}\right)^{\frac{d}{2}} h^d \exp\left(\frac{3}{2}\underline{\tilde{\xi}}_i^2\right) \\
&\qquad \left(1 + 3h\underline{\tilde{\xi}}_i \cdot \underline{u}^F - \frac{3}{2}\left(h\underline{u}^F\right)^2 + \frac{9}{2}h^2\left(\underline{\tilde{\xi}}_i \cdot \underline{u}^F\right)^2\right) + \mathcal{O}(h^{3+d}) \\
&=: M_h[\rho^F, \underline{u}^F](\underline{\tilde{\xi}}_i) + \mathcal{O}(h^{3+d}) \,.
\end{aligned}$$

The integral moments of $M_h(\underline{\xi}_i) := M_h[\rho^F, \underline{u}^F](\underline{\xi}_i)$ can be evaluated exactly by a Gaussian-type quadrature

$$\rho_h^F = \int_\Xi M_h(\underline{\xi})\, d\underline{\xi} = \sum_{i=0}^{q-1} w_i\, M_h(\underline{\xi}_i)\,, \tag{2.28}$$

$$\rho_h^F \underline{u}_h^F = \int_\Xi h\underline{\xi} M_h(\underline{\xi})\, d\underline{\xi} = \sum_{i=0}^{q-1} w_i \underline{\xi}_i\, M_h(\underline{\xi}_i)\,. \tag{2.29}$$

For $w = \rho^{F^{-1}} (\frac{2\pi}{3})^{d/2} h^{-d} \exp\left(\frac{3}{2}\underline{\tilde{\xi}}_i^2\right)^{-1}$ the weights w_i of the *D2Q9* lattice have been evaluated by He and Luo [78] as

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

whereas the weights for the *D3Q19* lattice are

$$
\begin{aligned}
w_0 &= \frac{1}{3} w \,, \\
w_i &= \frac{1}{8} w \quad \text{for } i \in \{1,2,3,10,11,12\} \,, \\
w_i &= \frac{1}{36} w \quad \text{for } i \in \{4,\dots 9,13,\dots,18\} \,.
\end{aligned}
$$

Assuming f_h is a solution of (2.27), it is known, see e. g. [110], that its integral moments can be approximated by

$$
\begin{aligned}
\int_\Xi f_h(\underline{x},\underline{\xi},t)\, d\underline{\xi} &= \sum_{i=0}^{q-1} w_i f_h(\underline{x},\underline{\xi}_i,t) + \mathcal{O}(h^2) \,, \\
\int_\Xi \underline{\xi} f_h(\underline{x},\underline{\xi},t)\, d\underline{\xi} &= \sum_{i=0}^{q-1} w_i \underline{\xi}_i f_h(\underline{x},\underline{\xi}_i,t) + \mathcal{O}(h) \,.
\end{aligned}
$$

Introducing the abbreviations

$$
f_i(\underline{x},t) := w_i\, f^h(\underline{x},\underline{\xi}_i,t) \,, \tag{2.30}
$$

$$
\rho_h^F(\underline{x},t) := \sum_{i=0}^{q-1} f_i(\underline{x},t) \,, \tag{2.31}
$$

$$
\underline{u}_h^F(\underline{x},t) := \frac{1}{\rho_h^F(\underline{x},t)} \sum_{i=0}^{q-1} \underline{\xi}_i f_i(\underline{x},t) \tag{2.32}
$$

$$
M_i := w_i M_h(\underline{\xi}_i) \tag{2.33}
$$

and multiplying Eq. (2.27) by w_i, we obtain the *velocity discrete* BGK*–Boltzmann Equations* as a set of q equations for $i = 0,\dots,q-1$:

$$
h^2 \frac{d}{dt} f_i = -\frac{1}{3\nu}\,(f_i - M_i) \qquad \text{in } \Omega_h \times \Xi_h \times I \,. \tag{2.34}
$$

We continue to discretise the differential operator by a central difference approximation

$$
\begin{aligned}
h^2(\partial_t + \underline{\xi}_i \nabla_{\underline{x}}) f_i(\underline{x} + \frac{h^2}{2}\underline{\xi}_i, t + \frac{h^2}{2}) &= \\
= f_i(\underline{x} + h^2, t + h^2) - f_i(\underline{x},t) + \mathcal{O}(h^4)&
\end{aligned} \tag{2.35}
$$

and additionally expand $f_i^h(t + h^2/2)$ in a Taylor series

$$
\begin{aligned}
f_i(\underline{x} + \frac{h^2}{2}\underline{\xi}_i, t + \frac{h^2}{2}) &= \\
&= \sum_{n=0}^{\infty} \left(\frac{h^2}{2}\partial_t + \frac{h^2}{2}\underline{\xi}_i \nabla_{\underline{x}} \right)^n f_i(\underline{x},t) \\
&= f_i(\underline{x},t) + \frac{h^2}{2}\left(\partial_t + \underline{\xi}_i \nabla_{\underline{x}}\right) f_i(\underline{x},t) + \mathcal{O}(h^4) \,.
\end{aligned} \tag{2.36}
$$

Approximating $\left(\partial_t + \underline{\xi}_i \nabla_{\underline{x}}\right) f_i(\underline{x},t)$ by a forward difference operator leads to

$$
\begin{aligned}
f_i(\underline{x} + \frac{h^2}{2}\underline{\xi}_i, t + \frac{h^2}{2}) &= \\
= f_i(\underline{x},t) + \frac{1}{2}\Big(f_i(\underline{x} + h^2\underline{\xi}_i, t + h^2) - f_i(\underline{x},t) \Big) + \mathcal{O}(h^4) \,.&
\end{aligned}
$$

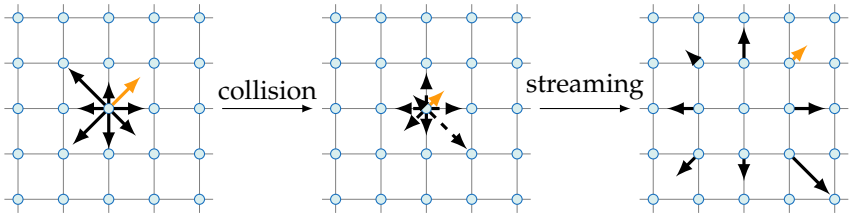


Figure 2.4: Illustration of the collision and streaming steps

Beginning with a shifted velocity discrete BGK-Boltzmann Equation (2.27)

$$h^2 \frac{d}{dt} f_i(\underline{x} + \frac{h^2}{2}\underline{\xi}_i, t + h^2/2) = \\ = \frac{1}{3\nu}\left(f_i(\underline{x} + \frac{h^2}{2}\underline{\xi}_i, t + h^2/2) - M_h(\underline{x} + \frac{h^2}{2}\underline{\xi}_i, t + h^2/2)\right)$$

and substituting the left hand side by (2.35) and $f_i(\underline{x} + h^2/2\underline{\xi}_i, t + h^2/2)$ on the right hand side by (2.36) one obtains

$$(3\nu + 1/2)\left(f_i(\underline{x} + h^2\underline{\xi}_i, t + h^2) - f_i(\underline{x}, t)\right) + O(h^4) = \\ = -\left(f_i(\underline{x}, t) - M_i(\underline{x} + \frac{h^2}{2}\underline{\xi}_i, t + h^2/2)\right) .$$

Finally approximating $M_i(\underline{x} + h^2/2\underline{\xi}_i, t + h^2/2) = M_i(\underline{x}, t) + O(h^2)$ reveals the *Lattice Boltzmann Equation (LBE)*

$$f_i(\underline{x} + h^2\underline{\xi}_i, t + h^2) - f_i(\underline{x}, t) = \\ = -\frac{1}{3\nu + 1/2}\left(f_i(\underline{x}, t) - M_i(\underline{x}, t)\right) + O(h^2) \qquad \text{in } \Omega_h \times I_h , \tag{2.37}$$

which is consistent of order 2 to the BGK-Boltzmann Equation (2.24) [110].

2.3 LATTICE BOLTZMANN METHOD

This section introduces the Lattice Boltzmann Method (LBM) and shows its connection to the macroscopic Navier–Stokes Equation (NSE). This is followed by enhancements to the basic LBM to deal with body forces, turbulence and porous media. The section is closed by the introduction of boundary conditions.

In the previous section we derived the Lattice Boltzmann Equation (LBE) via a discretisation of the Boltzmann Equation (BE). Splitting the Lattice Boltzmann Equation (LBE) (2.37) in two steps, with an intermediate value $\widetilde{f}_i$ one obtains the Lattice Boltzmann Method (LBM)

$$\widetilde{f}_i(\underline{x}, t) = -\frac{1}{\tau}\left(f_i(\underline{x}, t) - M_i(\underline{x}, t)\right) , \tag{2.38}$$

$$f_i(\underline{x} + h^2\underline{\xi}_i, t + h^2) = \widetilde{f}_i(\underline{x}, t) , \tag{2.39}$$

with the *relaxation time* $\tau = 3\nu + 1/2$. Equation (2.38) is called the *collision step* and (2.39) is called *propagation* or *streaming step*. Both steps are illustrated in Figure 2.4 for a *D2Q9* model. During the collision step the magnitudes of the f_is are changed, which is represented by the length of the arrows. During the collision step the populations f_i move to the next lattice node

Listing 2.1: Basic LBM algorithm

```
for t ∈ I_h {
  for x ∈ Ω_h {
    for i = 0,...,q−1 {
      f̃_i(x,t) = −(1/τ)(f_i(x,t) − M_i(x,t))
      f_i(x + h²ξ_i, t + h²) = f̃_i(x,t)
    }
  }
}
```

in direction $\underline{\xi}_i$. Listing 2.1 shows a basic LBM algorithm, which will be occasionally expanded throughout this thesis.

George Boole ☾ *(1815–1864) English mathematician, philosopher and logician*

Historically the LBM does not originate from a discrete BE, but from Cellular Automata (CA). CA use one Boolean variable on a each node of a lattice, which changes its state during each time step according to the state of the neighbouring nodes. The best known two-dimensional CA is probably *Conway's Game of Life*. It follows two simple rules

1. If a node is alive, it stays alive if it has two or three alive neighbours,
2. If a node is dead, it awakes if it has three alive neighbours.

Later Lattice Gas Cellular Automata (LGCA) used a small number of Boolean variables on each node which have already been connected to their direction of travel. Representatives of this class of CA are the HPP model [76] and its successor the FHP model [61]. The HPP model uses an orthogonal mesh and lacks rotational invariance, which is the reason why it does not lead to the NSE in the macroscopic limit [171, Chapter 2]. The FHP model however, employs an hexahedral mesh, which overcomes the shortcoming of the former model. Finally, the transition from Boolean to real variables eliminates statistical noise and has been done in the article by McNamara and Zanetti [128], which is usually stated as the first LBM paper. An excellent historical derivation of the LBM can be found in the book by Wolf–Gladrow [190].

STRENGTHS AND WEAKNESSES

A general strength of the LBM is its simplicity. As we have seen, the entire algorithm can be summarised to two lines. This simplicity can be transferred to the implementation, meaning that it is easy to code.

In comparison to conventional Computational Fluid Dynamics (CFD) methods, such as FEM or FVM, the LBM has overcome the necessity and the limitations of a geometry dependent mesh, by using a fixed, non-adaptive grid. Therefore, cost-intensive re-meshing especially during simulations of submerged particles are unnecessary, as we will see later-on.

The LBM is intrinsically parallelisable. The collision step (2.38) can be computed on one lattice node and does not need any other information, while the streaming step (2.39) only depends on its immediate neighbours. This properties demand for a parallelisation by domain decomposition. Naturally, it has been shown to scale almost linearly on up to 256 processing nodes [55, 110] and more.

LBM computations are extremely fast per cell and timestep and in comparison to e. g. the computation of an FEM cell. However, especially for fluid computations at high Reynolds numbers, when small scales are involved a

huge number of cells is necessary. Additionally, due to its explicit character also small time steps are necessary. This issue may be solved by local grid refinement techniques in the near future.

The LBM enters a slight compressibility regime to solve the pressure equation of the fluid. As all physical materials are slightly compressible, solutions obtained by the LBM may be a more realistic approximation of physical processes, than solutions of a strictly incompressible solver. However this issue is open for discussion.

Due to its explicit time-stepping algorithm the LBM is unable to solve stationary equations. However, if the solution to an in-stationary equation is naturally stationary the LBM is able to compute it.

LATTICE DIMENSIONS

The LBM is based on the simplicity of the equidistant lattice. Its introduction allows to quickly retrieve the data on a lattice node. Usually the data is stored in a multidimensional array that allows direct access. Opposing to that the cell data of FEM implementations has to be accessed via mappings or a walk through linked lists. Due to the arbitrary shape of the FEM cells each local computation has to be mapped on a reference cell and back, which is costly. As all LBM lattice cells are identical this extensive task can be circumvented by introducing so called lattice units (LU) defined such that the lattice spacing in LU equals unity.

Latt [116] introduced a two step approach, first transforming SI units into dimensionless units and in a second step transforming dimensionless units into lattice units. The first transformation is carried out by introducing characteristic quantities as explained at the end of Section 2.1.1. The transition from dimensionless units to lattice units is done in a similar manner by introducing the discrete lattice spacing $\Delta_x > 0$ and the discrete time step $\Delta_t > 0$. In the previous Section 2.2.3 the parameter h was introduced as the inverse ratio of the speed of sound $c_s = \Delta_x/\Delta_t = 1/h$ to implicitly obtain a diffusive scaling. For now however, we will understand Δ_x and Δ_t as independent variables.

Keeping in mind, that in dimensionless quantities both the characteristic length L^* and the characteristic velocity U^* equal unity, we can define the lattice spacing and the discrete time step as

$$\Delta_x := \frac{L^*}{N} = \frac{1}{N}\,, \qquad \Delta_t := \frac{L^*}{U^* N_T} = \frac{1}{T}\,,$$

where $N \in \mathbb{N}$ and $N_T \in \mathbb{N}$ are the number of spatial steps and the number of time steps, respectively. With it we can convert the dimensionless variables, marked by an asterisk $(\cdot)^*$ to the lattice system

$$t_{LU} = \frac{1}{\Delta_t} t^* \qquad \underline{x}_{LU} = \frac{1}{\Delta_x} \underline{x}^*$$
$$\underline{u}^F_{LU} = \frac{\Delta_t}{\Delta_x} \underline{u}^{F*} \qquad \rho^F_{LU} = \frac{\Delta_x^3}{\Delta_m} \rho^{F*}$$
$$p_{LU} = \frac{\Delta_t \Delta_x^2}{\Delta_m} p^*$$

and obtain the NSE in lattice units

$$\nabla_{LU} \cdot \underline{u}^F_{LU} = 0 \qquad \text{in } \Omega \times I\,,$$
$$d_{t\,LU}\, \underline{u}^F_{LU} + (\underline{u}^F_{LU} \cdot \nabla_{LU})\, \underline{u}^F_{LU} = \frac{1}{\rho^F_{LU}} \nabla_{LU}\, p_{LU} + \frac{1}{Re} \frac{\Delta_t}{\Delta_x^2} \Delta_{LU}\, \underline{u}^F_{LU} \qquad \text{in } \Omega \times I\,.$$

We see that one can naturally define the kinematic viscosity ν_{LU} in lattice units as

$$\nu_{LU} := \frac{1}{Re}\frac{\Delta_t}{\Delta_x^2} .$$

Of course the diversion via the dimensionless system is mathematically not necessary. However, it is useful for the solution of academic applications or benchmark computations that do not have a physical representation.

2.3.1 *Transition to Macroscopic Equations*

Variants of the LBM have been shown to solve a number of partial differential equations. As it has been derived above by a discretisation of the BGK-BE, it can be used to find solutions of the BGK-BE. Besides that, variation of the LBE have been used to solve the radiative transport equation [130], the compressible and incompressible Navier–Stokes Equation (NSE) [29, 149], Burgers Equation [5], Advection–Diffusion Equation (ADE) [60, 169] and the Poisson Equation [28] and maybe more.

Jan Burgers *(1895–1981) Dutch physicist*

Siméon Denis Poisson ☾ *(1781–1840) French mathematician and physicist, leading opponent of the wave theory of light*

Sydney Chapman ☾ *(1888–1970) British mathematician*

David Enskog *(1884–1947) Swedish mathematical physicist*

There are several possibilities to perform the transition from the LBE to the NSE in the literature. We begin with He and Luo [77], who use a Chapman–Enskog approach to derive the compressible NSE.

The second order Taylor expansion of the distribution function

$$\begin{aligned} f_i(\underline{x}+h^2\underline{\xi}_i, t+h^2) &= \sum_{n=0}^{\infty} \frac{h^{2n}}{n!} D_i^n f_i(\underline{x}, t) \\ &= f_i(\underline{x}, t) + h^2 D_i f_i(\underline{x}, t) + \frac{h^4}{2} D_i^2 f_i(\underline{x}, t) + \mathcal{O}(h^6) , \end{aligned}$$

with $D_i = (\partial_t + \underline{\xi}_i \cdot \nabla_{\underline{x}})$ yields

$$D_i f_i + \frac{h^2}{2} D_i^2 f_i = \frac{1}{\tau h^2}(f_i - M_i) + \mathcal{O}(h^4) .$$

Substituting the following multi-scale expansions of an assumed solution f_i of the LBE

$$\begin{aligned} f_i &= \sum_{n=0}^{\infty} \epsilon^n f_i^{(n)}, && \text{for } i = 0, \dots, q-1 , \\ \partial_t &= \epsilon \partial_{t_0} + \epsilon^2 \partial_{t_1} \\ \partial_{x_\alpha} &= \epsilon \partial_{0\alpha}, && \text{for } \alpha = 1, \dots, d , \end{aligned}$$

in the previous equation and comparing the coefficients one obtains

$$f_i^{(0)} = M_i , \tag{2.40}$$

$$D_i^{(0)} f_i^{(0)} = -\frac{1}{\tau h^2} f_i^{(1)} , \tag{2.41}$$

$$\partial_{t_1} f_i^{(0)} + \left(\frac{2\tau - 1}{2\tau}\right) D_i^{(0)} f_i^{(1)} = -\frac{1}{\tau h^2} f_i^{(2)} , \tag{2.42}$$

where $D_i^{(0)} = \partial_{t_0} + \underline{\xi}_i \cdot \nabla_0$ and ϵ is assumed to be of the order of the Knudson number Kn. In the multi-scale expansions t_0 represents the fast

convective scale and t_1 the slower diffusive scale. The assumed solution f_i is constraint by

$$\sum_{i=0}^{q-1} f_i^{(0)} = \rho_h^F, \qquad \sum_{i=0}^{q-1} \underline{\xi}_i f_i^{(0)} = \rho_h^F \underline{u}^F ,$$
$$\sum_{i=0}^{q-1} f_i^{(n)} = 0, \qquad \sum_{i=0}^{q-1} \underline{\xi}_i f_i^{(n)} = 0, \quad \text{for } n > 0 .$$

Then the first and second moments of (2.41) lead to the compressible Euler Equations

$$\partial_{t_0} \rho_h^F + \nabla_0 \cdot (\rho_h^F \underline{u}_h^F) = 0 , \tag{2.43}$$
$$\partial_{t_0} (\rho_h^F \underline{u}_h^F) + \nabla_0 \cdot \underline{\underline{\pi}}^{(0)} = 0 . \tag{2.44}$$

Leonhard Euler ☾ *(1707–1783) Swiss mathematician, physicist, astronomer, logician and engineer, face of the old Swiss 10 franc bill*

Here $\pi_{\alpha,\beta}^{(0)} = \sum_{i=0}^{q-1} \xi_{i,\alpha} \xi_{i,\beta} M_i = \rho u_\alpha u_\beta + p \delta_{\alpha,\beta}$, $\alpha, \beta = 1, \ldots, d$ is the zeroth-order momentum flux tensor with the pressure $p = c_S \rho_h^F$. In a similar manner, taking the first and second order moments of (2.42) leads to

$$\partial_{t_0} \rho_h^F = 0 , \tag{2.45}$$
$$\partial_{t_1} (\rho_h^F \underline{u}) + \left(1 - \frac{1}{2\tau}\right) \nabla_0 \cdot \underline{\underline{\pi}}^{(1)} = 0 , \tag{2.46}$$

where $\pi_{\alpha,\beta}^{(1)} = \sum_{i=0}^{q-1} \xi_{i,\alpha} \xi_{i,\beta} f_i^{(1)}$. The tensor $\pi_{\alpha,\beta}^{(1)}$ can be evaluate by taking the second order momentum of (2.41), yielding

$$\pi_{\alpha,\beta}^{(1)} = -\tau \rho_h^F (\partial_{0,\alpha} u_\beta + \partial_{0,\beta} u_\alpha) .$$

Substituting $\underline{\underline{\pi}}^{(1)}$ in (2.46) leads to

$$\rho_h^F \partial_{t_1} \underline{u}_h^F - \rho_h^F \left(\tau - \frac{1}{2}\right) \nabla_0 \left(\nabla_0 \underline{u}_h^F + (\nabla_0 \underline{u}_h^F)^\top\right) = 0 .$$

Multiplication of the last equation with ϵ^2 and adding the product of ϵ and (2.43) and (2.44) allows the re-substitution of the differential operators and one obtains the compressible NSE

$$\partial_t \rho_h^F + \nabla \cdot (\rho_h^F \underline{u}_h^F) = 0 ,$$
$$\rho_h^F \partial_t \underline{u}_h^F + \rho_h^F \underline{u}_h^F \nabla \underline{u}_h^F = -\nabla p + \rho_h^F \nu \nabla \cdot \left(\nabla \underline{u}_h^F + (\nabla \underline{u}_h^F)^\top\right) ,$$

with the kinematic viscosity $\nu = (\tau - 1/2)$. In the small Mach number limit $Ma \to 0$, which has already been used in the expansion of the discrete EDF in Section 2.2.3, the compression can be omitted and one further obtains the incompressible NSE

$$\nabla \cdot \underline{u}_h^F = 0 ,$$
$$\partial_t \underline{u}_h^F + \underline{u}_h^F \cdot \nabla \underline{u}_h^F = -\frac{1}{\rho_h^F} \nabla p + \nu \nabla^2 \underline{u}_h^F ,$$

with an accuracy of $O(Ma^2)$ in the continuity equation and $O(Ma^3)$ in the momentum equation. However, the Chapman–Enskog analysis is often criticised with regard to its lack of mathematical rigour [86, 190].

Besides the introduced Chapman–Enskog expansion, there are other ways to discover the NSE in the LBE. Using a diffusive scaling $\underline{x} \to \underline{x}/\epsilon$, $t \to t/\epsilon^2$

which describes the small Knudson number and small Mach number limit of kinetic equations and a rescaling of the fluid velocity $\underline{u}^F \to \epsilon \underline{u}^F$, Junk and Klar [101] link an assumed expanded solution of certain Lattice Boltzmann (LB) schemes directly to the incompressible NSE and analyse the asymptotic behaviour.

Finally Holdych et al. [86] found an expression of f_i which only depends on M_i by recursively applying the LBE to itself

$$f_i(\underline{x}, t) = \frac{1}{\tau} \sum_{n=1}^{\infty} \left(1 - \frac{1}{\tau}\right)^{n-1} M_i(\underline{x} - nh^2 \underline{\xi}_i, t + nh^2) \, .$$

With it they recover the governing partial differential equations and provide the associated truncation errors. The article by Chen and Doolen [30] provides a good review on the state of the art of the LBM for fluid flows in the year 1998.

2.3.2 *Turbulence Scheme*

Flows can be categorised as laminar and turbulent. One speaks of laminar flow if the fluid moves in parallel layers, which happens at low Reynolds numbers. Laminar flows prove stable towards disturbances or at least merge into a new laminar flow as reaction to a disturbance [48].

The two most important characteristics of turbulence are randomness and a wide range of space and time scales [124]. Randomness appears e. g. in the fluid velocity fluctuations around an averaged value. It can also be observed in the apparently arbitrary occurrence of eddies and vortices. How to correctly model turbulende is still an open field of research.

In turbulent flows energy is transferred through a cascade of interacting scales, from large to small. Large scale turbulence involves a high turbulence reynolds number $Re_L = \overline{U}\overline{L}/\nu$, with $\overline{L} > 0$ the length scale representing the order of size of the large scales, e. g. the maximal length between two correlated probes of the flow and $\overline{U} > 0$ characterises the overall turbulent velocity fluctuation. At high Reynolds numbers the viscosity term in the NSE becomes small compared to the non-linear convective term, implying little viscous effects. However the viscous diffusive term becomes important at small, so called Kolmogorov scales where kinetic energy is dissipated into heat.

Andrey Nikolaevich Kolmogorov *(1903–1987) Russian mathematician*

The magnitude of turbulence can be determined by splitting the fluid velocity into the time mean velocity

$$\overline{\underline{u}}^F(\underline{x}) = \lim_{T \to \infty} \frac{1}{T} \int_0^T \underline{u}^F(\underline{x}, t) dt$$

and the fluctuation $\widehat{\underline{u}}^F$ around it

$$\underline{u}^F(\underline{x}, t) = \overline{\underline{u}}^F(\underline{x}) + \widehat{\underline{u}}^F(\underline{x}, t) \, .$$

The turbulence intensity is then defined by the ratio of the mean square root of the velocity fluctuations and the mean velocity [48]

$$I(\underline{x}, t) = \frac{\left\| \widehat{\underline{u}}^F(\underline{x}, t) \right\|_2}{\sqrt{2} \left\| \overline{\underline{u}}^F(\underline{x}) \right\|_2} \, .$$

If I is only a few percent the flow is of low turbulence. For I greater then 10% one speaks of highly turbulent flows. The turbulence intensity depends

on position and time, therefore the flow characteristic can change. Parts of a flow can be laminar, while the flow shows turbulent behaviour in other areas.

As turbulence occurs on almost all scales, its computation is challenging. While vortices on large scales can be simulated directly, turbulence on small scales can cause numerical simulations to diverge. A theoretical solution to this issue is increasing the resolution until all vortices can be resolved. This computationally expensive approach is called Direct Numerical Simulation (DNS) and usually excels available compute power, which makes it unfeasible for practical problems. A second approach is to only resolve vertices up to a certain level and smoothen so-called subgrid scale vertices by a turbulence model. This common approach is called Large Eddie Simulation (LES).

A common representative of LES tools is the Smagorinsky scheme [165], which has been applied to LBM by Hou et al. [87]. This scheme smoothens subgrid-scale vortices by adding an eddy viscosity ν_t, using

Joseph Smagorinsky *(1924–2005) American meteorologist, devised first simulations on climate change due to carbon dioxide*

$$\nu_t = (C_S h)^2 S,$$

to replace the relaxation time τ in the collision step (2.38) by

$$\widetilde{\tau} = 3(\nu_t + \nu) + \frac{1}{2}\,.$$

Here

$$S = \sqrt{2}\,\|\underline{\underline{S}}\|_F = \sqrt{2\sum_{\alpha=1}^{d}\sum_{\beta=1}^{d}|S_{\alpha,\beta}|^2} \tag{2.47}$$

is the Frobenius norm $\|\cdot\|_F$ of the strain rate tensor $\underline{\underline{S}}$ multiplied by $\sqrt{2}$, given by

$$S_{\alpha,\beta} = \frac{1}{2}\left(\frac{\partial \underline{u}_\alpha}{\partial \underline{x}_\beta} + \frac{\partial \underline{u}_\beta}{\partial \underline{x}_\alpha}\right), \quad \alpha,\beta \in \{1,\dots,d\}\,. \tag{2.48}$$

Further $\underline{u}_\alpha$ denotes the α-th component of the fluid velocity vector and $\underline{x}_\beta$ the β-th spatial coordinate. The problem specific Smagorinsky constant $C_S \in \mathbb{R}^+$ is chosen according to the flow configuration. In the LBM framework the strain rate tensor can be computed from the distribution functions using the non-equilibrium stress tensor $\underline{\underline{\Pi}}$, given by

$$\Pi_{\alpha,\beta} = \sum_{i}^{q-1} \underline{\xi}_{i,\alpha}\underline{\xi}_{i,\beta}(f_i - f_i^{eq}), \quad \alpha,\beta \in \{1,\dots,d\}\,. \tag{2.49}$$

Applying the condition

$$S_{\alpha\beta} = -\frac{h^2}{2\rho\tau_*}\Pi_{\alpha,\beta}\,, \tag{2.50}$$

the new relaxation time can be computed according to Yu et al. [198].

2.3.3 *Forcing Scheme*

Up to now we assumed that body forces were non-existent. We therefore have to reintroduce body forces to the LB scheme somewhat artificially,

which is not a trivial task. Guo and Shu [72] provide a good overview on schemes available in the literature, including residual terms of the resulting hydrodynamic equations.

The first proposition to include a force in the LBE goes back to Shan and Chen [160] in the year 1993. For a force $\underline{F}(\underline{x},t) = \rho^F(\underline{x},t)\underline{a}(\underline{x},t)$, with the force acceleration $\underline{a}: \Omega \times I \to \mathbb{R}^3$, they adapted the fluid velocity occurring in the EDF by

$$\underline{u}^*(\underline{x},t) = \underline{u}_h^F(\underline{x},t) + \underline{a}(\underline{x},t)\tau\Delta_t \tag{2.51}$$

$$= \underline{u}_h^F(\underline{x},t) + \frac{\underline{F}(\underline{x},t)}{\rho^F(\underline{x},t)}\tau\Delta_t\,, \tag{2.52}$$

where $\Delta_t > 0$ denotes the time step. Developing the first order moment of the collision step (2.38) and using Equations (2.29) and (2.32) determines the momentum change on a lattice node during one collision

$$\begin{aligned}
\sum_i \widetilde{f}_h^i &= \sum_i f_h^i \underline{\xi}_i - \sum_i \frac{1}{\tau}\left(f_i - M_i[\rho_h^F, \underline{u}^*]\right)\underline{\xi}_i \\
\widetilde{\rho}_h^F \widetilde{\underline{u}}_h^F &= \rho_h^F \underline{u}_h^F - \frac{1}{\tau}\left(\rho_h^F \underline{u}_h^F - \rho_h^F(\underline{u}_h^F + \frac{\underline{F}}{\rho^F}\tau\Delta_t)\right) \\
\widetilde{\rho}_h^F \widetilde{\underline{u}}_h^F - \rho_h^F \underline{u}_h^F &= \Delta_t \underline{F}\,.
\end{aligned}$$

Here, $\widetilde{\rho}_h^F \widetilde{\underline{u}}_h^F$ denotes the fluid density and velocity after the collision. More modern schemes [62, 79, 112, 121] use an additional forcing term $F_i: \mathbb{R}^3 \times \Omega_h \times I_h \to \mathbb{R}$ in the LBE

$$f_i(\underline{x} + \Delta_t\underline{\xi}_i, t + \Delta_t) = f_i(\underline{x},t) - \frac{1}{\tau}\left(f_i(\underline{x},t) - M_i[\rho_h^F, \underline{u}^*](\underline{x},t)\right) + F_i(\underline{F},\underline{x},t)\,. \tag{2.53}$$

We present a scheme originally proposed by Guo, Zheng and Shi [71], which is implemented in `OpenLB`. In it, the equilibrium velocity $\underline{u}^*$ as well as the fluid velocity $\underline{u}^F$ are defined as

$$\rho_h^F \underline{u}_h^* := \rho_h^F \underline{u}_h^F := \sum_i \underline{\xi}_i f_i + \frac{h^2}{2}\underline{F}\,.$$

Moreover the forcing term is defined as

$$F_i(\underline{F},\underline{x},t) := \left(1 - \frac{1}{2\tau}\right) w_i \left(h(\underline{\xi}_i - \underline{u}_h^F(\underline{x},t)) + h^2 \underline{\xi}_i \cdot \underline{u}_h^F(\underline{x},t)\underline{\widetilde{\xi}}_i\right) \cdot \underline{F}(\underline{x},t)\,. \tag{2.54}$$

Applying a Chapman–Enskog expansion to the forced LBE (2.53), Guo and Shu [72] obtain the following hydrodynamic equations

$$\begin{aligned}
\partial_t \rho^F + \nabla \cdot (\rho^F \underline{u}^F) &= R_\rho\,, \\
\partial_t(\rho^F \underline{u}^F) + \nabla \cdot (\rho^F \underline{u}^F \underline{u}^F) &= -\nabla \rho^F + \nabla \cdot (2\rho^F \nu \underline{\underline{S}}) + \underline{F} + \underline{R}_u\,,
\end{aligned}$$

with $\underline{\underline{S}}$ given by (2.48) and R_ρ and $\underline{R}_u$ are residual terms that show the deviations of the derived equations from the desired mass and momentum conservation equations. For the method by Shan and Chen [160], the residuums are

$$\begin{aligned}
R_\rho &= -\frac{\Delta_t}{2}\nabla \cdot \underline{F}\,, \\
\underline{R}_u &= -\frac{\Delta_t}{2}\partial_t \underline{F} + \Delta_t\left(\tau - \frac{1}{2}\right)\nabla \cdot (\underline{u}^* \underline{F} + \underline{F}\underline{u}^*)\,,
\end{aligned}$$

while for the method by Guo, Zheng and Shi [71] the residuums are equal to zero. For a constant force the residuums of both methods vanish.

2.3.4 *Porous Media Scheme*

Simulation of flow through porous media is a common tool in the oil and gas industry as well as chemical engineering. It can be seen as a special fluid–solid two phase flow where the solid phase is fixed. Its simulation can be conducted on different scales. Firstly at pore scale, where the geometry of the porous medium is resolved and the flow can be simulated as a solution of the NSE in a domain representing the fluid part and according boundary conditions. This provides detailed information of the flow through the pores.

However in some cases only averaged values, e. g. the pressure drop over a given distance is of interest. In this case the Representative Elementary Volume (REV) scale is preferable. On the REV scale apparent fluid quantities are averaged over a small volume $V(\underline{x}) \subset \Omega$. The volume has to be chosen large enough to obtain meaningful values, but still considerably smaller than the entire domain Ω. For example the porosity γ is defined as ratio of the volume of fluid $V_f(\underline{x})$ contained in the entire REV

$$\gamma(\underline{x}) = \frac{V_f(\underline{x})}{V(\underline{x})} ,$$

where the porosity is assumed to be constant over time. Similarly other fluid quantities ϕ can be defined at the REV scale by

$$\langle \phi(\underline{x}, t) \rangle := \frac{1}{V(\underline{x})} \int_{V(\underline{x})} \phi(\underline{x}', t) \mathrm{d}\underline{x}' .$$

One popular empiric "law" relating fluid variables with the surrounding media is *Darcy's law*, connecting the pressure gradient and the fluid velocity by

Henry Darcy *(1803–1858) French engineer*

$$\nabla p = -\frac{\mu}{K} \underline{u}^F , \tag{2.55}$$

with the material specific permeability $K > 0$. It can be expanded by a viscous term to become the *Brinkman* [20] form of Darcy's law

$$\nabla p = -\frac{\nu}{K} \underline{u}^F + \nu_e \Delta \underline{u}^F ,$$

with an effective viscosity $\nu_e > 0$.

Spaid and Phelan Jr [168] adopted Brinkman's approach and proposed an LBM based method to model fluid flow in heterogeneous porous media. In their work they adapt the fluid velocity in EDF similar to the forcing scheme by Shan and Chen [160]

$$\underline{u}^* = \underline{u}^F_h - s\tau\beta \underline{u}^F_h \qquad \text{in } \Omega_h \times I_h ,$$

where τ is the relaxation time and the Boolean $s = s(\underline{x})$ is a switch depending on whether a lattice node is in the porous domain or not. For a porous node ($s(\underline{x}) = 1$) the velocity simplifies to

$$\underline{u}^* = \underline{u}^F_h (1 - \tau\beta) .$$

If $\beta = \nu/K$ the Brinkman equation is represented by this model. The scheme was used by Pingen, Evgrafov and Maute [147] as well as Krause [110] to simulate topology optimisation of flows.

Guo and Zhao [73] proposed another scheme, which asymptotically solves a generalised NSE for flow through a porous medium

$$\nabla \cdot \underline{u}^F = 0 \qquad \text{in } \Omega \times I\,,$$
$$\partial_t \underline{u}^F + (\underline{u}^F \cdot \nabla)\left(\frac{\underline{u}^F}{\gamma}\right) = -\frac{1}{\rho}^F \nabla(\gamma p) + \nu_e \Delta \underline{u}^F + \underline{F}_\gamma \qquad \text{in } \Omega \times I\,,$$

and therefore allows simulations of higher Reynolds numbers. Here, $\underline{F}^\gamma$ represents the total body force due to the presence of a porous medium, given by

$$\underline{F}^\gamma = -\frac{\gamma\nu}{K}\underline{u}^F - \frac{\gamma F^\gamma}{\sqrt{K}}\|\underline{u}^F\|_2 \underline{u}^F\,.$$

Philipp Forchheimer *(1852–1933) Austrian engineer*

On the right hand side of the above equation, the first and the second terms are the linear Darcy and non-linear Forchheimer drags. Here the geometric function F_γ and the permeability K are given by the experimentally founded expressions

$$F_\gamma = \frac{1.75}{\sqrt{150\gamma^3}}\,, \quad K = \frac{\gamma^3 d_P^2}{150(1-\gamma)^2}\,,$$

where d_P is the diameter of particles constituting the porous medium. The continuous model is introduced in the LBM, altering the EDF by including the porosity in the non-linear terms

$$M_i = w_i\left(1 + 3h\underline{\tilde{\xi}}_i \cdot \underline{u}^F - \frac{3}{2}\frac{h^2}{\gamma}\left(\underline{u}^F\right)^2 + \frac{9}{2}\frac{h^2}{\gamma}\left(\underline{\tilde{\xi}}_i \cdot \underline{u}^F\right)^2\right)\,.$$

Additionally, the force F_γ is implemented by the forcing scheme in Guo and Zhao [73] and introduced in the previous section.

Later, Fattahi et al. [54] compared transport phenomena at the porous interface on multiple scales. Simulating flow through a cubical domain, which is partly filled by a porous medium, the porous medium is firstly formed by resolved spherical particles at rest, and secondly by the proposed homogenised porous media scheme. A smooth transition zone based on the porosity between the free flow and the porous model was applied and a simple porosity-dependent rescaling of the viscosity in the interface layer allowed to reproduce the results obtained by averaging the pore-scale solution.

2.3.5 *Homogenised Lattice Boltzmann Method*

The porous media schemes introduced in the previous section only apply to porous media in rest. They can be used to simulate e. g. flow through sand and rocks as they appear in petroleum reservoirs. Our interest is to model moving particles by a porous media approach. The idea is to assume the particles to be solid at their centre and featuring a boundary layer that smoothly changes to a pure fluid.

We propose a method, that uses a convex combination of the fluid velocity $\underline{u}_h^F$ and the velocity of a rigid body $\underline{u}^B : \Omega \times I \to \mathbb{R}^3$, which is explained in more detail in Section 4.3.1. The velocity in the EDF is set to

$$\underline{u}^*(\underline{x}, t) = d(\underline{x}, t)\underline{u}_h^F(\underline{x}, t) + (1 - d(\underline{x}, t))\,\underline{u}^B(\underline{x}, t)\,, \tag{2.56}$$

where $d(\underline{x}, t) : \Omega_h \times I_h \to [0, 1]$ is a level-set function related to the porosity.

Investigating special cases for $d = d(\underline{x}, t)$ we find:

- For $d = 1$, one obtains $\underline{u}^* = \underline{u}^F$, which is the classic BGK–LBM. In this case the collision step is momentum conserving and it has been shown that a Chapman–Enskog procedure recovers the incompressible NSE.
- For $d = 0$, one obtains $\underline{u}^* = \underline{u}^B$. In this case the momentum conservation of the collision step is violated and one obtains a relaxation towards $\underline{u}^B$. The amount of momentum change happening during the collision step is later transferred to the porous medium, such that the total momentum in the system is conserved.
- For $d \in (0,1)$ and $\underline{u}^B = \underline{0}$ one obtains $\underline{u}^* = d\underline{u}^F$, which has been covered by Spaid and Phelan [168]. For $d = 1 - \nu\tau/K$ this models the flow of a fluid with kinematic viscosity ν through a porous media with permeability K. This approach also removes a specific amount of momentum from the fluid and it was shown to recover the Brinkman equation.
- In the Homogenised Lattice Boltzmann Method (HLBM) one uses a $d \in [0,1]$ and arbitrary but known $\underline{u}^B$. This can be interpreted as flow through a moving porous media.

Taking the first moment of Equation (2.38), the momentum change at one lattice node during the collision step is

$$\sum_i \underline{\xi}_i \widetilde{f}_i - \sum_i \underline{\xi}_i f_i = -\sum_i \frac{1}{\tau}\left(f_i - M_i[\rho_h^F, \underline{u}^*]\right)\underline{\xi}_i \,, \tag{2.57}$$

$$\widetilde{\rho}_h^F \widetilde{\underline{u}}_h^F - \rho_h^F \underline{u}_h^F = -\frac{1}{\tau}\left(\rho_h^F \underline{u}_h^F - \rho_h^F (d\underline{u}_h^F + (1-d)\underline{u}^B)\right) \,, \tag{2.58}$$

$$\widetilde{\rho}_h^F \widetilde{\underline{u}}_h^F - \rho_h^F \underline{u}_h^F = -\frac{\rho_h^F}{\tau}(1-d)(\underline{u}_h^F + \underline{u}^B) \,. \tag{2.59}$$

Here $\widetilde{\rho}_h^F$ and $\widetilde{\underline{u}}_h^F$ denote the mass density and velocity after the collision step. Using simple algebra we reorder the equilibrium velocity

$$\begin{aligned}\underline{u}^* &= d\underline{u}_h^F + (1-d)\,\underline{u}^B \\ &= \underline{u}_h^F + (d-1)\underline{u}_h^F + (1-d)\,\underline{u}^B \\ &= \underline{u}_h^F + (1-d)(\underline{u}^B - \underline{u}_h^F) \,,\end{aligned}$$

which according to the model by Shan and Chen and Equation (2.51) leads to a force

$$\underline{F} = \frac{\rho_h^F}{\tau \Delta_t}(1-d)(\underline{u}^B - \underline{u}_h^F)$$

acting on the fluid. Again, one can see, that for a pure fluid, when $d = 1$ no momentum change occurs.

2.3.6 *Boundary Conditions*

We will now introduce boundary conditions for the LBM. They differ significantly from boundary conditions used in continuous simulation techniques such as FEM in the sense that it is not possible to set macroscopic scales, but mesoscopic scales have to be used. The next paragraphs introduce bounce-back boundary conditions which correspond to no-slip boundary conditions in macroscopic scales and boundary conditions proposed by Zou and He

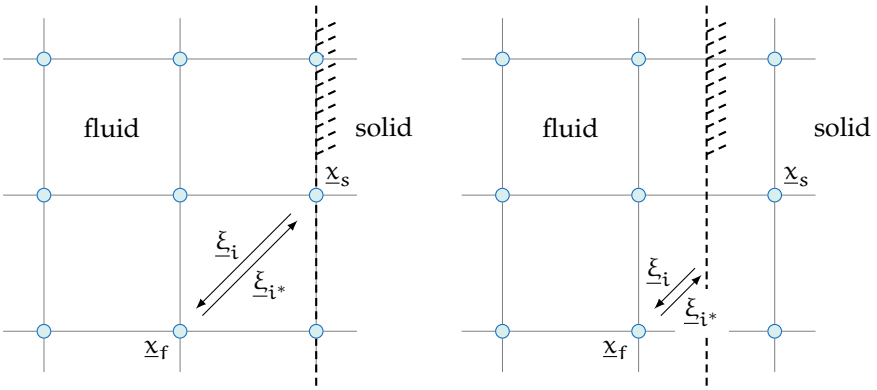


Figure 2.5: Bounceback boundary conditions. The left side displays the situation for a full-way bounceback scheme, the right side for a half-way bounceback scheme.

[204] which allow the definition of the flow velocity and pressure on a lattice node. As a last example, boundary conditions proposed by Bouzidi, Firdaouss and Lallemand [19] that interpolate the boundary between lattice nodes are presented.

BOUNCEBACK

Bounceback boundary conditions are based on the idea that fluid molecules hitting the boundary are reflected in the direction from which they came, due to the roughness of the boundary. Therefore a particle distribution f_i which is transported from a fluid node $\underline{x}_F$ to a boundary node $\underline{x}_B$ is returned to the initial node. If the component of f_i tangential to the normal vector of the boundary remains, this results in free slip, whereas if it is negated this results in a no-slip boundary. For the first method we recommend Hänel [74] or Inamuro, Yoshina and Ogino [94].

There are several implementations of the no-slip bounceback scheme. For the standard or *full-way* bounceback scheme the post-collision function at the boundary node $\underline{x}_b$ is computed by

$$\widetilde{f}_{i^*}(\underline{x}_b, t) = \widetilde{f}_i(\underline{x}_f, t) \, ,$$

where $\underline{\xi}_{i^*} = -\underline{\xi}_i$, see the left side of Figure 2.5. In the standard bounceback the collision step is not executed on the boundary node. However if the collision step is executed, the scheme is called *modified* bounceback.

The right side of Figure 2.5 outlines the situation for the *half-way* bounceback, where the wall is placed in the middle of the fluid and solid nodes $\underline{x}_w = (\underline{x}_s + \underline{x}_f)/2$. The post-collision particles of node $\underline{x}_f$ in direction of $\underline{\xi}_i$ arrive at the wall after half a time step $h^2/2$, are reflected with a reversed velocity and return to $\underline{x}_f$ after the full time step

$$f_{i^*}(\underline{x}_f, t + h^2) = f_i(\underline{x}_f, t + h) \, .$$

As half-way bounceback boundary conditions assume the boundary to be always exactly in the middle between two nodes of the lattice they are prone to errors. According to Zou and He [204] this method produces results of second-order accuracy. Gallivan et al. [65] developed the mean square velocity error for flow around two different cylinders using half-way bounceback boundary conditions. Bouzidi, Firdaouss and Lallemand [19] embraced this

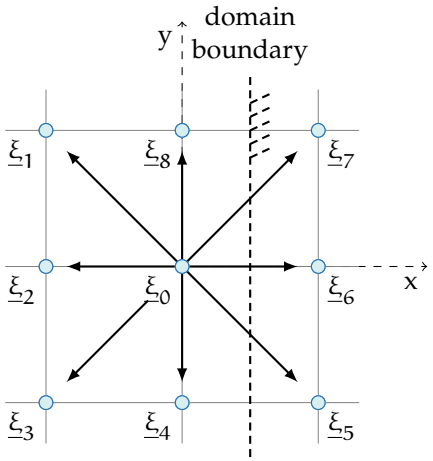


Figure 2.6: Lattice node with directions.

challenge and developed a more robust approach by respecting the exact position of the wall. The full-way bounceback is only of first order accuracy and is therefore less accurate than the half-way bounceback scheme. Ladd and Verberg [112] introduce a method called *continuous bounceback*, where they specify a continuous parameter on each node, that represents the fluid volume fraction associated with the respective node. Details on this method can be found in [180].

ZOU–HE

Zou–He boundary conditions are the equivalent to Dirichlet boundary conditions in continuous flow simulations, which specify macroscopic values like pressure or velocity on the boundary. The Zou–He boundary conditions [204] are based on the half-way bounceback boundary conditions. We will first derive the conditions for a given velocity $\underline{u}_h^F$ and then have a look at those for a given pressure.

Peter Gustav Lejeune Dirichlet ☾ *(1805–1859) German mathematician*

Lets assume the macroscopic velocity $\underline{u}_h^F = (u_x^F, u_y^F)$ is specified at the boundary. Figure 2.6 shows the transfer operations for such a node. After the streaming step $f_0, f_4, f_5, f_6, f_7, f_8$ are known, f_1, f_2, f_3 have yet to be determined. Using Equations (2.31) and (2.32) for the density and the velocity one obtains:

$$f_1 + f_2 + f_3 = \rho_h^F - \sum_{n=4}^{8} f_n \,, \tag{2.60}$$

$$f_1 - f_3 = \rho_h^F u_y^F - (f_8 - f_4 - f_7 + f_5) \,, \tag{2.61}$$

$$f_1 + f_2 + f_3 = \rho_h^F u_x^F + (f_5 + f_6 + f_7) \,. \tag{2.62}$$

$$\tag{2.63}$$

By equating lines (2.61) and (2.63) the density is revealed as

$$\rho_h^F = \frac{1}{1 - u_x^F}\left(f_0 + f_8 + f_4 + 2(f_5 + f_6 + f_7)\right) \,.$$

Lets assume that the bounceback rule is still valid for the non-equilibrium part of the particle distribution normal to the boundary and

$$f_2 - M_2 = f_6 - M_6 \,.$$

The EDFs can be computed by Equation (2.18) and

$$f_2 = f_6 + \frac{2}{3}\rho_h^F u_x^F \, .$$

With it the last missing distribution functions can be calculated as follows

$$\begin{aligned} f_1 &= f_5 - \frac{1}{2}(f_8 - f_4) + \frac{1}{6}\rho_h^F u_x^F + \frac{1}{2}\rho_h^F u_y^F \, , \\ f_3 &= f_7 - \frac{1}{2}(f_8 - f_4) + \frac{1}{6}\rho_h^F u_x^F - \frac{1}{2}\rho_h^F u_y^F \, . \end{aligned}$$

Turning to the pressure boundary condition, one takes the mass density ρ_h^F as defined in (2.31) as a reference value for the pressure p. The two state variables are connected by the relation

$$p = \rho c_S^2 \, ,$$

where c_S denotes the speed of sound in the fluid. The pressure boundary condition is treated in the same way as velocity boundary condition. With reference to Figure 2.6, we assume a given pressure ρ_h^F and a given velocity $u_y^F = 0$ for a flow through the boundary in direction of the x-axis. Then, after the streaming step, the distribution functions f_2, f_3, f_4, f_5, f_6 are known, leading to

$$\begin{aligned} f_1 + f_2 + f_3 &= \rho_h^F - (f_0 + f_4 + f_5 + f_6 + f_7 + f_8) \, , \\ f_1 + f_2 + f_3 &= -\rho_h^F u_x^F + (f_5 + f_6 + f_7) \, , \\ f_1 - f_3 &= \rho_h^F u_y^F - f_4 + f_5 - f_7 + f_8 \, , \end{aligned}$$

for the unknown u_x^F, f_1, f_2, f_3. This system of linear equations can be solved for u_x^F:

$$u_x^F = \frac{f_0 + f_4 + f_8 + 2(f_5 + f_6 + f_7)}{\rho_h^F} - 1.$$

Again one unknown (f_2) is gained by applying the bounceback rule to the particle distribution normal to the inlet.

$$f_2 - M_2 = f_6 - M_6.$$

Hence f_1 and f_3 are obtained by

$$\begin{aligned} f_2 &= f_6 - \frac{2}{3}\rho_h^F u_x^F \, , \\ f_1 &= f_5 + \frac{1}{2}(f_4 - f_8) - \frac{1}{6}\rho_h^F u_x^F \, , \\ f_3 &= f_7 - \frac{1}{2}(f_4 - f_8) - \frac{1}{6}\rho_h^F u_x^F \, . \end{aligned}$$

In both cases, the velocity and pressure boundary, the collision step is applied to the boundary nodes as well as to the fluid nodes.

In their paper Zou and He [204] also derive this boundary condition for the *D2Q9i* model, which is using an adapted equilibrium distribution function, and the *D3Q15* model. They also state that this method cannot be used in complex geometries and extrapolation schemes, such as the following, should be used instead.

INAMURO

Inamuro, Yoshina and Ogino [94] proposed another boundary scheme for a no-slip condition at moving walls. It is assumed, that the unknown distribution functions are an equilibrium function of a counter slip velocity, such that the fluid at the wall is equal to the wall velocity.

Going back to Figure 2.6 the wall moves with known velocity $\underline{u}^W = (u_x^W, u_y^W)$. And the distributions f_1, f_2, f_3 are unknown. It is further assumed, that the fluid velocity normal to the wall is equal to the wall velocity $u_x^F = u_x^W$. The difference between the velocities in y direction is denoted by $u' \in \mathbb{R}$, such that

$$u_y^F = u_y^W + u' \, .$$

With that the unknown f_i, $i \in \{1,2,3\}$ can be calculated using the equilibrium distribution $M_i = M_i[\rho', \underline{u}^F]$ by

$$\begin{aligned}
f_1 =& \frac{1}{36}\rho'\big(1 + 3(u_y^W + u' - u_x^W) - \frac{3}{2}((u_y^W + u')^2 + (u_x^W)^2) \\
&+ \frac{9}{2}(u_y^W + u' - u_x^W)^2\big) \, , \\
f_2 =& \frac{1}{9}\rho'\big(1 - 3u_x^W + 3(u_x^W)^2 - \frac{3}{2}(u_y^W + u')^2\big) \, , \\
f_3 =& \frac{1}{36}\rho'\big(1 - 3(u_y^W + u' + u_x^w) - \frac{3}{2}((u_y^W + u')^2 + (u_x^W)^2) \\
&+ \frac{9}{2}(u_y^W + u' + u_x^W)^2\big) \, ,
\end{aligned}$$

with unknown $\rho' \in \mathbb{R}$. Including the equation for the density ρ_h^F, this system of equations can be solved for the unknown ρ_h^F, ρ' and u' and one obtains

$$\begin{aligned}
\rho_h^F &= 1 + u_x\big(f_0 + f_4 + f_8 + 2(f_5 + f_6 + f_7)\big) \, , \\
\rho' &= 6\big(-\rho_h^F u_x + \frac{f_5 + f_6 + f_7}{1 + 3u_y + 3u_y^2}\big) \, , \\
u' &= \frac{1}{1 - 3u_x} + \Big(6\big(\rho_h^F u_y + \frac{f_4 + f_5 - f_7 - f_8}{\rho'}\big) - u_y + 3u_x u_y\Big) \, .
\end{aligned}$$

The boundary conditions was used to compute two dimensional Poiseuille and Couette flows and experimentally found to be second order convergent.

BOUZIDI

Bouzidi, Firdaouss and Lallemand [19] proposed boundary conditions that handle curved boundaries by interpolation between lattice nodes. For the one-dimensional situation shown in Figure 2.7, the position of the boundary is given by

$$d = \frac{\|\underline{x}_1 - \underline{x}_w\|_2}{h} \, .$$

Particles, represented by their distribution function f_i, leave the last fluid node at $\underline{x}_1$ in direction of the first solid node at $\underline{x}_s$ are reflected at $\underline{x}_w$. However they can only reach a fluid node for d equal to $0, 1/2$ or 1. Therefore the distribution function at $\underline{x}_1$ is unknown. Bouzidi, Firdaouss and Lallemand [19] propose the following scheme to solve this situation:

- For $d < 1/2$ interpolate the distribution function at point $\underline{x}_P$ from the information given in the fluid. These particles will travel to $\underline{x}_1$ after reflection at the wall.

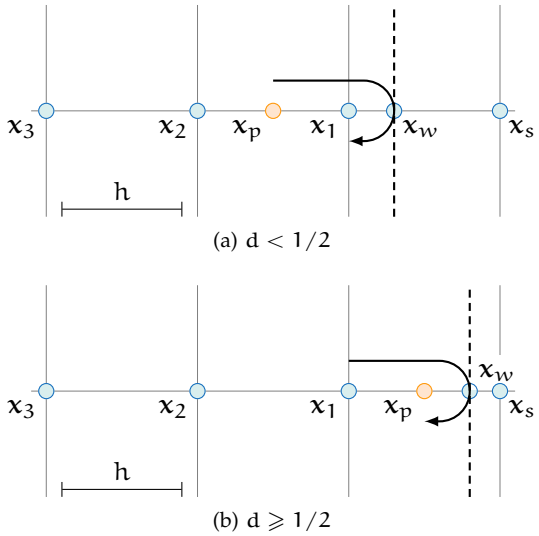


Figure 2.7: Illustration of Bouzidi's boundary conditions. $\underline{x}_i (i = 1,2,3)$ are fluid nodes. $\underline{x}_s$ is a solid node, $\underline{x}_w$ and $\underline{x}_p$ the positions of the wall and a particle.

- For $d \geqslant 1/2$ the information on the particles leaving $\underline{x}_1$ and arriving in $\underline{x}_p$ together with the situation after the propagation step at nodes $\underline{x}_2$ and $\underline{x}_3$ is used to compute the unknown quantities at $\underline{x}_1$.

Let $\underline{\xi}_i$ be the direction from $\underline{x}_1$ to $\underline{x}_w$ and $\underline{\xi}_{i^*} = -\underline{\xi}_i$. Using linear interpolation, one defines

$$f_{i^*}(\underline{x}_1, t + h^2) := \begin{cases} 2d\widetilde{f}_i(\underline{x}_1, t) + (1-2d)\widetilde{f}_i(\underline{x}_1 - h^2\underline{\xi}_i, t), & d < \frac{1}{2}\,, \\ \frac{1}{2d}\widetilde{f}_i(\underline{x}_1, t) + \left(1 - \frac{1}{2d}\right)\widetilde{f}_{i^*}(\underline{x}_1, t), & d \geqslant \frac{1}{2}\,, \end{cases}$$

where $\widetilde{f}_i$ denotes the intermediate states of the distribution function after the collision step but before the streaming step. In their article Bouzidi, Firdaouss and Lallemand [19] also adapt this method for moving boundaries and carry out numerical tests.

2.4 IMPLEMENTATIONAL ASPECTS

The aim of this section is to introduce two practical aspects that occurred during the implementation of `OpenLB`. The first is an efficient method to voxelise highly complex surfaces, obtained e. g. by Magnetic Resonance Imaging (MRI) or Computer Tomography (CT). The second is the used data structure and parallelisation, as it is the basis of the parallelisation of the Lagrangian particle phase later in this work.

2.4.1 *Voxeliser*

One advantage of the LBM is that there is no need for a complex mesh. This saves a lot of time during the preprocessing of simulations in complex flow geometries. However the lattice in LBM is created on-the-fly at the beginning of each simulation. It is therefore essential that the creating algorithm is fast. We call this process *voxelisation*.

Let $\Omega \in \mathbb{R}^3$ be the computational domain and $\mathrm{Mat} : \mathbb{Z}_h^3 \to \mathbb{N}$ be a three dimensional map of material numbers with $\mathbb{Z}_h := \{i\,h \,|\, i \in \mathbb{Z}\} \subset \mathbb{R}$ and

Listing 2.2: Class Octree

```
Class Octree {
  Octree child[8];

  double x[3], L_d;
  int d;
  Triangle T_d[];
  bool insideGeomtry;
}
```

$h \in \mathbb{R}^+$ is the intended spatial step. The aim of this section is to develop an algorithm that efficiently constructs the map $\mathrm{Mat}(\underline{x})$ holding the geometric information of the lattice

$$\mathrm{Mat}(\underline{x}) = \begin{cases} 1 & \text{for } \underline{x} \in \Omega \cap \mathbb{Z}^3_h \\ 0 & \text{else} \end{cases} .$$

Usually the domain Ω is obtained as an STL-file containing a triangulation $\mathscr{T}$ of the boundary $\partial\Omega$. For each triangle $T \in \mathscr{T}$ only the coordinates of the corners are stored, meaning that it does not contain any information of the connectivity of the triangles. Points shared by several triangles are stored multiple times. However, it is used by a range of applications and can be handled by a wide range of software. In `OpenLB` the map Mat is created from STL data via an octree structure.

The voxelisation works as follows. We start with a cube containing the entire triangulation Ω and repeatedly divide the cube into eight identical sub-cubes. The division process ends if the resulting cubes do not contain a part of the triangulation or the final size is reached.

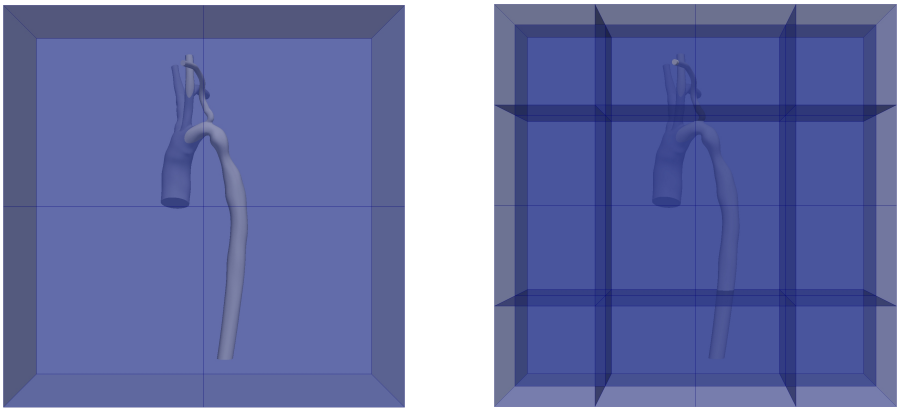
A *graph* is a pair $G = (V, E)$, with a set of vortices V and a set of edges $E \subset V^2$. Each connected graph G without cycles is called a *tree*. A tree for which each internal vortex has eight children is called an *octree*.

Each vortex $v \in V$ of the octree is an instantiation of class `Octree`, containing its position $\underline{x} \in \mathbb{Z}^3_h$ and edge length L_d to represent a cube $C_d = L_d^3 \subset \mathbb{R}^3$, see Listing 2.2 for a sketch. Additionally it stores all triangles $T_d = \{t \in \mathscr{T} | t \cap C \neq \emptyset\}$ intersecting C_d and its distance to the root, called depth $d \in \{0, \dots, d^*\}$. The maximal depth $d^* \in \mathbb{N}$ of the octree is chosen such that

$$L_0 = 2^{d^*} h > \mathrm{diam}\,(\Omega)$$

holds. The position of the root vortex with edge length L_0 is chosen such that the cube C_0 contains the entire triangulation of $\partial\Omega$. It is then recursively divided into eight vortices of edge length $L_{d+1} = L_d/2$ until either $T_d = \emptyset$, i.e. the cube C_d does not contain any intersection with a triangle of the geometry or its depth equals the maximal tree depth ($d = d^*$). Figure 2.8 illustrates the voxelisation process of an exemplary geometry of an aortic arch.

A vortex $v \in V$ is marked as *inside* the geometry if its centre $\underline{x} \in \mathbb{N}$ is inside the geometry. Its material number is then set to one, $\mathrm{Mat}(\underline{x}) := 1$, or zero otherwise $\mathrm{Mat}(\underline{x}) := 0$. To determine its state, a ray stabbing technique is used. Therefore a ray with origin in the centre of v is sent in an arbitrary direction, counting the number of intersections with $\mathscr{T}$. For an



(a) $d = 1$ (b) $d = 2$

(c) $d = 3$ (d) $d = 8$

Figure 2.8: The voxelisation process of an aortic arch at different depths d.

odd number of intersections ν is inside the geometry, for an even number outside. During the segmentation of CT or MRI data, often non-watertight geometries are created. This method can also deal with such geometries, by using several rays in different directions and introducing an appropriate voting procedure.

In additional steps, a boundary layer can be added by defining

$$\mathrm{Mat}^*(\underline{x}) = \begin{cases} 2 & \text{for } \underline{x} \notin \Omega \cap \mathbb{Z}_h^3 \text{ and } \underline{y} \in \Omega \cap \mathbb{Z}_h^3 \\ 1 & \text{for } \underline{x} \in \Omega \cap \mathbb{Z}_h^3 \\ 0 & \text{else} \end{cases},$$

where $\|\underline{x} - \underline{y}\|_2 = h$. If needed, the *material numbers* can be changed to mark lattice nodes for special dynamics, such as porous media dynamics or boundary dynamics. Some boundary conditions such as Bouzidi's boundary condition (see Section 2.3.6) need an exact distance of the lattice node $\underline{x}$ to the domain's boundary $\partial\Omega$ in each lattice direction. Using an octree this distance can be efficiently computed, as all triangles in the neighbourhood of $\underline{x}$ can be determined in $\log(|\mathscr{T}|)$ steps. This greatly reduces the number of potential point-triangle distances that have to be computed. The implementation of the newly proposed voxeliser immensely reduced the preprocessing time, which was essential for the simulations executed in Mirzaee et al. [131].

2.4.2 *Data Structure and Parallelisation*

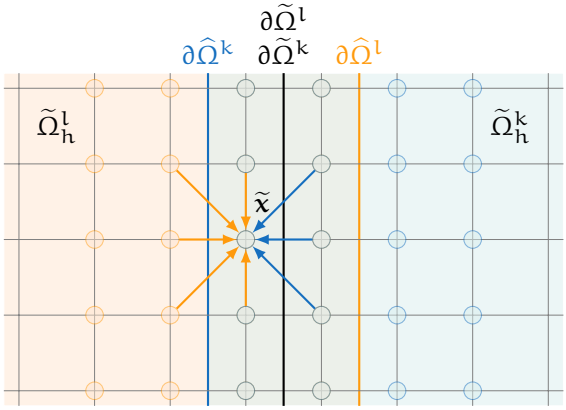
Efficient and simple parallelisation is a key feature of the LBM. The most time demanding steps in LB simulations are collision and streaming. Since the collision step is purely local and the streaming step only requires data of the neighbouring nodes, parallelising by domain decomposition leads to low communication costs and is therefore efficient. This has been discussed by many researchers, e.g. in [83, 123, 136, 148, 201], recently implementations for graphic processing units have become possible [15, 195]. In the following we introduce the parallelisation concept implemented in `OpenLB` [58, 80, 84].

In `OpenLB`, each lattice node is represented by an instantiation of the class `Cell`, which allocates memory for the q values of the discrete distribution function f_i. Additionally it holds a pointer to an so-called external field, which is used by porous media or forcing schemes, as well as a pointer to a dynamics object, which defines the behaviour of the `Cell` during simulation. An array of `Cells` is stored as a member variable in class `BlockLatticeXD` and represents a rectangular set of lattice nodes. The array is stored as consecutive data in memory, which allows direct access. Additionally the Central Processing Unit (CPU) can pre-cache data during the main loop over the cells. Several instances of type `BlockLatticeXD` are again combined into one instance of class `SuperLatticeXD`.

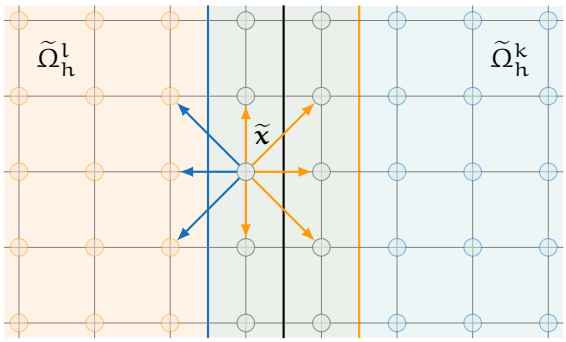
The parallelisation concept in `OpenLB` follows the classical approach of partitioning data according to the geometrical origin. The considered discrete domain Ω_h is divided into $n \in \mathbb{N}$ disjoint, preferably cube-shaped sub-lattices Ω_h^k $(k = 0, 1, ..., n-1)$, of almost equal sizes. This becomes feasible by extending Ω_h to a cuboid-shaped lattice $\widetilde{\Omega}_h$ through the introduction of ghost cells. Ghost cells are cells situated outside of Ω_h, which do not contribute to the simulation, but simplify data organisation. Then, $\widetilde{\Omega}_h$ is split into $m \in \mathbb{N}$ disjoint, always cuboid-shaped sub-lattices $\widetilde{\Omega}_h^l$ $(l = 0, 1, ..., m-1)$ of as similar size as possible (Figure 2.10a). Afterwards, sub-lattices $\widetilde{\Omega}_h^l \cap \Omega_h = \emptyset$ consisting exclusively of ghost cells are neglected (Figure 2.10b). The remaining sub-lattices $\widetilde{\Omega}_h^l$ are shrunk to fit Ω_h as closely as possible (Figure 2.10c). Finally the $\widetilde{\Omega}_h^l$ are expanded in all directions by a layer of lattice nodes, such that neighbouring cuboids overlap. The resulting enlarged discrete cuboid is denoted by $\widehat{\Omega}_h^l$. After each collision and streaming step overlapping lattice nodes are communicated between CPUs. This means that the distribution functions $f_i(\widetilde{x})$ of a node $\widetilde{x} \in \widetilde{\Omega}_h^l \cap (\hat{\Omega}_h^k \backslash \widetilde{\Omega}_h^k)$ computed by $\widetilde{\Omega}_h^l$ and in the overlap of $\hat{\Omega}_h^k$ is send to the respective node on $\widehat{\Omega}_h^k$ and vice versa, see Figure 2.9. The updated LBM algorithm can be found in Listing 2.3.

2.5 APPLICATION: AORTA

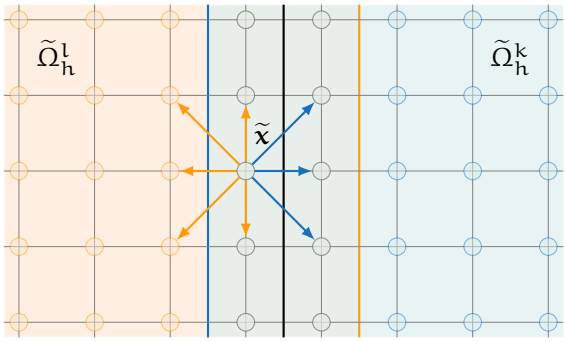
To close this chapter a medical application of the introduced LBM follows. The text has been taken mainly from Henn et al. [80] and been altered to fit the notation in this thesis. It is a result of the participation at the first CFD challenge *Simulation of Hemodynamics in a Patient-Specific Aortic Coarctation Model*, which took place in the context of the conference for *Statistical Atlases and Computational Modelling of the Heart (STACOM)* in 2012. Its aim was the simulation of the blood flow in a patient specific model of an aortic coarctation. The main objective was to compute the pressure drop around



(a) Before streaming

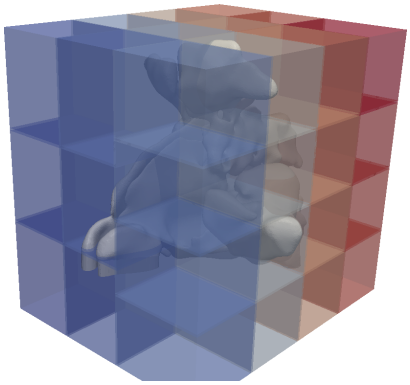


(b) After streaming

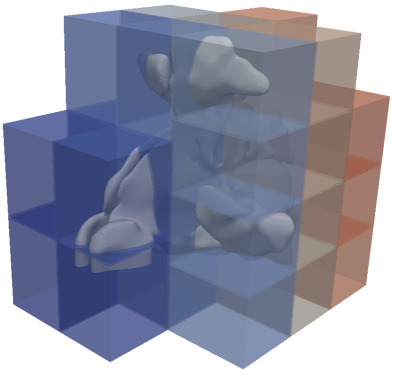


(c) After communication

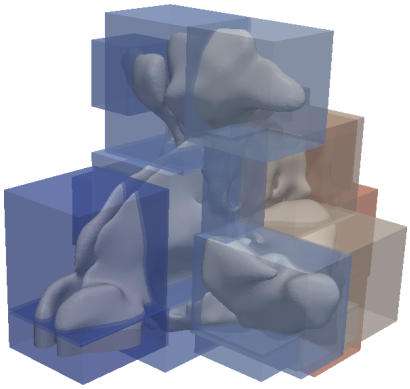
Figure 2.9: Illustration of the communication between neighbouring cuboids with one node overlap.



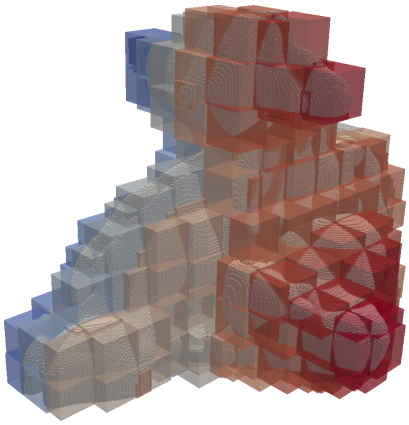
(a) Extended domain $\widetilde{\Omega}_h$.



(b) Extended domain $\widetilde{\Omega}_h$ after neglecting all $\widetilde{\Omega}_h^l$ not containing fluid cells.



(c) Extended domain $\widetilde{\Omega}_h$ after shrinking remaining $\widetilde{\Omega}_h^l$.



(d) A decomposition consisting of 452 subdomains $\widetilde{\Omega}_h^l$ used for productive runs.

Figure 2.10: Illustration of the development of Ω_h for 48 subdomains $\widetilde{\Omega}_h^l$ (a-c) and an exemplary decomposition for a productive simulations (d).

Listing 2.3: Parallel LBM algorithm

```
for t ∈ I_h {
  for x ∈ Ω_h {
    for i = 0,...,q−1 {
      f̃_i(x, t) = −(1/τ)(f_i(x, t) − M_{h,i}(x, t))
      f_i(x + h²ξ_i, t + h²) = f̃_i(x, t)
    }
  }
  communicate overlap
}
```

the contraction, that can be clearly observed in Figure 2.11 centred between the two planes π_1 and π_2.

Patient-specific numerical simulation of human organs opens new opportunities for medical diagnosis and therapy. They are even more advantageous, if they do not require additional radiating screenings, but are based on computer tomography imaging from standard procedures. In the case of the human respiratory system, numerical simulations of air flow have already proven to be accurate in the United Airways project [109]. Therefore, an adaption of the concept to human blood flow is highly appreciated.

An anomaly in the human cardiovascular system, such as a coarctation of the aorta, obstructs the body's supply of nutrients and stresses the heart. In particular, the contraction can lead to an intense drop in pressure, which directly affects the health of the patient. This pathological case accounts approximately one out of ten of all congenital heart defects, and is usually corrected surgically or by use of a catheter.

A conventional measurement of the pressure drop under resting conditions is an easy task for a clinician, but measuring the pressure gradient under exercise conditions is more challenging. Usually artificial stress is created by administering a drug to increase heart rate and contractibility. As this may have unwanted side-effects, it opens up a range of applications for CFD techniques. By simulation of a section of the blood system mimicking the real situation, the health-endangering measurements can be shifted into a virtual model. In this paper we investigate to what extent the LBM is of significance for the present medical case of an eight year old female patient. It is worth mentioning that the used model is likely to be expanded by an elastic model of the aorta geometry to achieve more realistic results.

2.5.1 *Simulation Setup*

A given surface $\partial\Omega$ of an aorta with a moderate thoracic aortic coarctation is voxelised using 5 different resolutions, reaching from $235 \times 118 \times 402$ to $1168 \times 582 \times 2002$ voxels (cf. Figure 2.11). Recorded data of 20 measurements of the ascending aortic flow is interpolated by cubic splines with periodic boundary conditions. A smooth start-up phase is added to suppress undesired pressure fluctuation. The resulting function is illustrated in Figure 2.12. A velocity boundary condition, as introduced by Skordos in [164] with a Poiseuille flow profile reflecting the measured flow volumes is set at the ascending aortic opening. The blood flow through the upper branch vessels was experimentally measured as a percentage of the ascending aortic flow. Therefore the flow through the left carotid artery is set to be 11.3 % of

Jean Léonard Marie Poiseuille *(1797–1869) French physiologist and physicist*

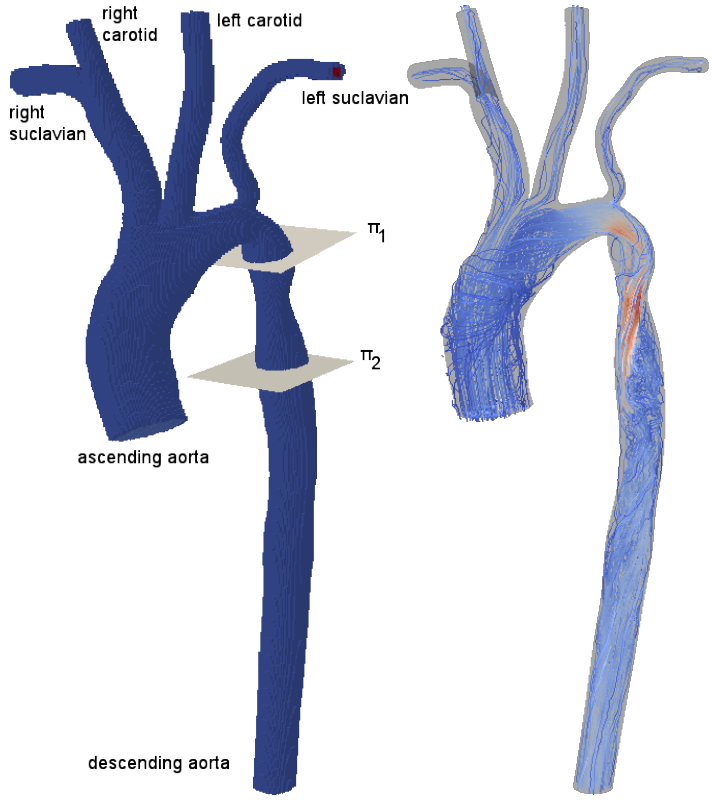


Figure 2.11: Voxelised geometry Ω_h of the aortic arc with spacial resolution of $235 \times 118 \times 402$ voxels and flow visualisations at the point of time of highest flow rate. Colour indicates the flow velocity $\underline{\|u\|_2}^F$.

the aortic flow, whereas the flow through the left subclavian artery is set to be 4.26 %. For the right carotid and the right subclavian only a combined value of 25.6 % was available. Hence the flow through these two arteries was calculated depending on the areas of the openings and set to 10.9 % and 14.7 % of the aortic flow, respectively. The boundary condition at the descending aorta is set as pressure condition, i.e. the pressure is fixed to 0 mmHg for all times. A full-way bounce back condition is assigned to the remaining surface. The fluid is assumed to be Newtonian with a density of $\rho^F = 0.001\,\mathrm{gr/mm^3}$ and a dynamic viscosity of $\mu = 0.004\,\mathrm{gr/mm/s}$. A *D3Q19* BGK-LBM, supported by a Smagorinsky turbulence model with constant $C_S = 0.12$, was used. The simulations were executed using the open-source library `OpenLB`[1]. Computation times varied between approx. 1.5 hours on 64 Intel Xeon X5650@2.67GHz cores for the smallest resolution and 6 days occupying 512 AMD Opteron@2.6GHz cores for the highest resolution.

2.5.2 *Results*

Two cardiac cycles have been simulated and the pressure drop at the aortic coarctation was determined by calculation of the spatial pressure average in two planes π_1 and π_2 (see Figure 2.11). The absolute pressure over time at the ascending aorta and the pressure drop around the coarctation are shown

1 http://www.openlb.org

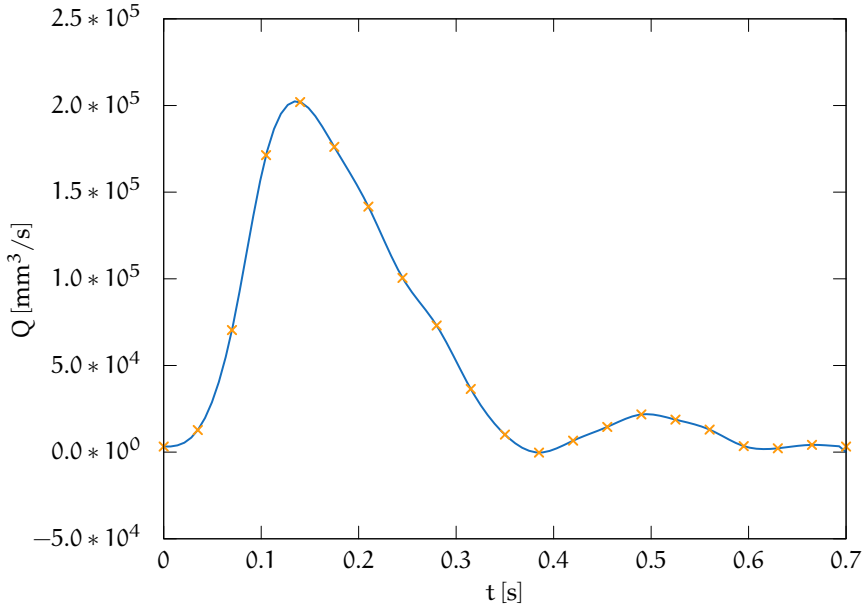


Figure 2.12: Cubic spline interpolation of the provided flow information at the ascending aorta [ml/s]. The marked points represent the measured data.

in Figures 2.13 and 2.14 for different spatial resolutions. It is found that with increasing resolution the resulting curves become less smooth. This effect is more pronounced in the graphs of the pressure drop and may be due to the fact that with increasing resolution small turbulences become more significant, which otherwise are smoothed.

Figure 2.11 shows streamlines representing the flow and velocity $\underline{u}^F$ at the time of highest inflow. As expected the velocity increases in the coarctation. The flow is predominantly laminar except for turbulence in the areas before and after the coarctation.

In Table 2.2 the peak and mean pressure difference around the coarctation, the flow splits through the upper arteries and the descending aorta as a percentage of the flow through the ascending aorta and systolic and diastolic pressure in the ascending aorta are listed for all simulations. As the pressure is only determined up to an additive constant when solving the NSE, the values of the absolute pressure at the ascending aorta have been shifted to fit the measured systolic and diastolic pressure of 115 mmHg and 65 mmHg. It is found that the pressure is decreasing with increasing spatial and time discretisation.

Assuming the measured systolic pressure of 115 mmHg as solution, we obtain the Experimental Order of Convergence (EOC) for $0 < h_1 < h_2$, which is defined by

$$\mathrm{EOC}(h_1, h_2) := \frac{\ln\left(\mathrm{Err}_{h_1}/\mathrm{Err}_{h_2}\right)}{\ln\left(h_1/h_2\right)},$$

where $\mathrm{Err}_h = p_h^{sys} - 115$ is the error of the computed systolic pressure with respect to the measured value for a given spacing h. The results are listed in Table 2.1.

From this table we see that this LBM yields a systolic pressure p_h^{sys} of linear order. For the highest spatial and time resolution we obtain $p_h^{sys} = 116.97\,\mathrm{mmHg}$.

h_2	h_1	EOC
1/402	1/802	1.413
1/802	1/1202	1.436
1/1202	1/1602	1.261
1/1602	1/2002	1.008

Table 2.1: Experimental Order of Convergence (EOC) of the systolic pressure.

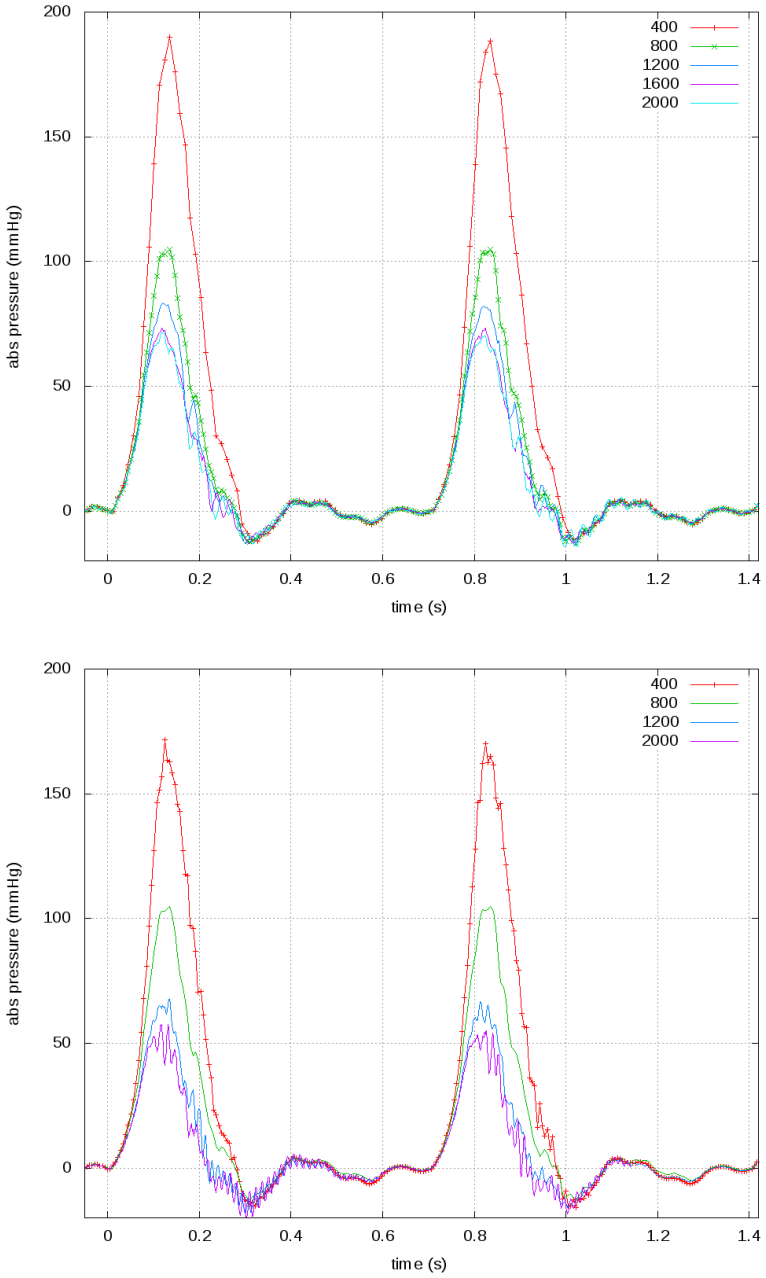


Figure 2.13: The absolute pressure over time at the ascending aorta is shown for different spacial resolutions. The upper graph shows the resulting curve for the higher time resolution. The according time-step sizes can be found in Table 2.2.

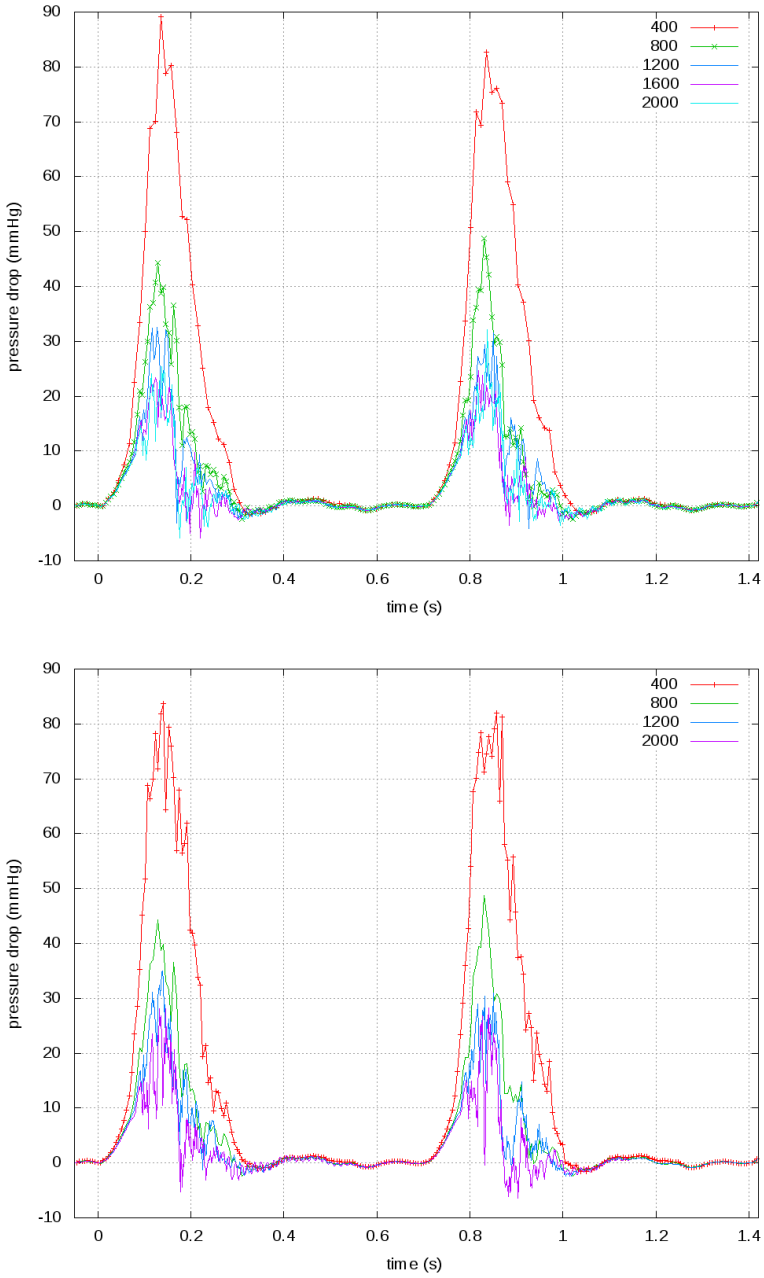


Figure 2.14: The pressure drop between the planes π_1 and π_2 around the coarctation are shown for different spatial resolutions. The upper graph shows the resulting curve for the higher time resolution. The according time-step sizes can be found in Table 2.2.

	Spatial Resolution	$235 \times 118 \times 402$		$468 \times 234 \times 802$		$702 \times 350 \times 1202$		$935 \times 466 \times 1602$	$1168 \times 582 \times 2002$	
	Time Resolution [s]	$5.64 \cdot 10^{-6}$	$2.28 \cdot 10^{-6}$	$2.28 \cdot 10^{-6}$	$1.41 \cdot 10^{-6}$	$1.88 \cdot 10^{-6}$	$9.41 \cdot 10^{-7}$	$1.41 \cdot 10^{-6}$	$1.13 \cdot 10^{-6}$	$5.64 \cdot 10^{-7}$
First cycle	Peak pressure difference [mmHg]	80.249	83.642	44.256	51.635	32.519	35.011	23.325	24.707	28.150
	Mean pressure difference [mmHg]	13.674	13.772	5.889	5.713	3.887	3.864	2.566	2.585	2.689
	Flow splits (Q_{IA}/ Q_{LCCA}/ Q_{LSA}/ Q_{DAo}) [% of Q_{AA}]	19.6/6.9/ 2.5/60.5	20.1/7.0/ 2.5/60.5	28.0/12.1/ 5.7/62.2	28.0/12.1/ 5.7/62.0	31.4/13.6/ 6.6/64.3	30.4/13.3/ 6.5/61.7	30.4/13.3/ 6.5/61.7	32.9/14.5/ 7.1/67.2	33.0/14.5/ 7.17/66.9
	Pressure in AAo (Sys/-Dia) [mmHg]	250.51/65	232.76/65	166.04/65	154.20/65	144.57/65	128.20/65	134.27/65	131.54/65	116.97/65
Second cycle	Peak pressure difference [mmHg]	82.657	83.643	48.804	50.422	29.400	30.353	24.738	32.108	28.224
	Mean pressure difference [mmHg]	14.242	14.119	5.405	5.943	4.008	3.568	2.601	2.788	—
	Flow splits (Q_{IA}/ Q_{LCCA}/ Q_{LSA}/ Q_{DAo}) [% of Q_{AAo}]	20.0/6.9/ 2.5/60.5	20.1/7.0/ 2.5/60.5	28.0/12.1/ 5.7/62.2	28.0/12.2/ 5.7/61.9	31.5/13.5/ 6.6/64.6	31.5/13.5/ 6.6/64.3	32.4/14.2/ 7.0/65.8	33.0/14.5/ 7.2/67.4	—
	Pressure in AAo (Sys/-Dia) [mmHg]	248.89/65	231.07/65	165.77/65	150.61/65	141.95/65	128.24/65	135.05/65	129.68/65	115.48/65
	Processors	64 Intel Xeon X5650 @2.67GHz			128 Intel Xeon X5355 @2.66GHz			512 AMD Opteron @2.6GHz		
	Computation time [h]	1.4	2.8	14.19	13.7	26.4	52.7	20.3	45.4	cancelled after 6 days

Table 2.2: Computed values for pressure and flow both cycles; Q_{IA}: flow through innominate artery; Q_{LCCA}: flow through left common carotid artery; Q_{LSA}: flow through left subclavian artery; Q_{DAo}: flow through descending aorta; Q_{AAo} flow through ascending aorta; The values for the systolic and diastolic pressure have been adjusted to fit the experimentally taken values of 115/65 mmHg.

3 PARTICULATE FLOWS: ONE WAY COUPLING

For simulating particle-fluid flows, the Lattice Boltzmann Method (LBM) has been shown to be an efficient solver of the Navier–Stokes Equation (NSE) model of the carrier phase, which has been introduced in the previous chapter. This chapter concentrates on the modelling and simulation of the particle phase. It is assumed that the particle concentration is small enough, such that the particles' influence on the fluid can be neglected. For that a *one way coupling* is chosen, meaning that the carrier phase acts on the particle phase but not vice versa.

The particle phase can be modelled either as discrete particles or as a continuous medium. For discrete particles usually a Lagrangian point of view is chosen. With it, the observer moves through space and time along with each individual particle. Plotting the position of a particle over time one obtains the pathline or trajectory. This point of reference can be visualised by a boat swimming in a river. In the Lagrangian system the observer is sitting in the boat, moving along with the flow of the river. The Lagrangian case is easy to understand, conservation of mass and momentum are obvious. However, assuming no particle interaction, the computational costs grow with the number of simulated particles, as each particle trajectory needs the solution of an additional equation of motion.

For continuous media usually an Eulerian point of view is chosen, where one focuses on a fixed point in space. Tracking all particles passing through said fixed point one obtains streaklines. In the picture of the river, the observer remains in a fixed position e. g. sitting on a bridge or the bank of the river, watching the water pass. The computational effort is independent of the number of particles, as all possible particle motions are treated by one Advection–Diffusion Equation (ADE) (3.21).

This chapter is split in two parts. The first part assumes the particle phase to consist of discrete spherically shaped mass points (Sections 3.1–3.4). The particulate flow is therefore modelled by an Euler–Lagrange (EL) ansatz, meaning that the fluid phase is seen in an Eulerian system of reference and the particle phase in a Lagrangian system of reference. The second part assumes the particle phase to be of continuous character, hence the particle and the carrier phase are both seen in an Eulerian perspective, called Euler–Euler (EE) ansatz (Sections 3.5 –3.6).

	EULER	LAGRANGE
point of observation	space point	material point
notation	lower case $\underline{x}$	upper case $\underline{X}$
rate of change	$\underbrace{\partial_t \underline{\phi}}_{\text{Eulerian rate of change}} + \underbrace{\underline{u} \cdot \nabla \underline{\phi}}_{\text{convective rate of change}}$	$\underbrace{D_t \underline{\phi}}_{\text{Lagrangian rate of change}}$
computational cost	independent of amount of particles	grows with amount of particles

Table 3.1: Comparison of the Eulerian and Lagrangian point of view.

The section about the EL approach starts by introducing the mathematical model of the fluid forces acting on submerged particles (Section 3.1). We then propose numerical methods to solve the equations modelling the particle trajectories and analyse their limitations. In Section 3.2 we concentrate on the implementation by explaining in detail the generic concept, that is responsible for the Lagrangian particle phase. We then propose two strategies for parallel computation of the particle trajectories, which are both based on the domain decomposition of the LBM as introduced in Section 2.4.2. Using Message Passing Interface (MPI) routines, the implementation of the potentially faster approach is explained in Section 3.2.3. The prepared methods are applied to simulations of airborne particulates in the human respiratory systems in Section 3.6. Firstly, simulations of particulate flows in a model of the trachea and a bronchial bifurcation are executed in order to validate the implementation and obtain speedup results. Then the validated method is used to simulate particle deposition in a patient specific geometry of a human nasal cavity in Section 3.4.

In the second part the EE approach is introduced. In this case both, the carrier phase and the dilute particle phase are assumed to be continuous. The fluid phase is modelled by a NSE, the particle phase by an ADE. Both equations are solved by an LBM. In contrast to the EL method, this allows to use the parallelisation intrinsic to LBM for both phases. However, for the particle phase new boundary conditions and an adaption of the Smagorinsky model (Section 2.3.2) are needed and introduced in Sections 3.5.4 and 3.5.5. Finally the simulations of Section 3.6 are repeated using the newly developed methods and their results are compared to the previous ones.

3.1 EULER–LAGRANGE

The aim of this section is to propose a method to simulate dilute one-way coupled particulate flows. The carrier phase of this two component flow is modelled by the NSE and solved by a LBM as introduced in Section 2. A particle is assumed to be solid, of spherical shape, with mass $M > 0$ and radius $R > 0$, much smaller than the characteristic length $L > 0$ of the flow. At time $t \in I = [0, T]$, it resides at position $\underline{X} : I \to \Omega \subset \mathbb{R}^d$ and moves with velocity $\underline{U} : I \to \mathbb{R}^d$ as reaction to a force $\underline{F} : I \to \mathbb{R}^d$ acting on it. The domain Ω and the time interval I are generally the same that are used for the fluid simulation.

The particulate flow is modelled by the NSE

$$\partial_t \underline{u}^F + (\underline{u}^F \cdot \nabla)\underline{u}^F + \nu \Delta \underline{u}^F = -\frac{1}{\rho^F}\nabla p \qquad \text{in } \Omega \times I\,, \tag{3.1a}$$

$$\nabla \cdot \underline{u}^F = 0 \qquad \text{in } \Omega \times I \tag{3.1b}$$

and the equations of motion for a number $N \in \mathbb{N}$ of particles

$$d_t \underline{U}_k(t) = \frac{1}{M}\underline{F}_k(t) \qquad \text{for } k \in \{1, \dots N\}\,, \tag{3.2a}$$

$$d_t \underline{X}_k(t) = \underline{U}_k(t) \qquad \text{for } k \in \{1, \dots N\} \tag{3.2b}$$

plus suitable initial and boundary conditions. The above systems of equations are one-way coupled by the hydrodynamic forces acting on the particles, which generally depends on the fluid velocity $\underline{F}_k(t) = \underline{F}_k(\underline{u}^F, t)$. Parti-

cle rotation is neglected. Two important dimensionless quantities are used to characterise the particle behaviour. The first is the *particle Reynolds number*

$$\mathrm{Re}^{\mathrm{P}} := \frac{2R\left\|\underline{U}-\underline{u}^{\mathrm{F}}\right\|_2}{\nu} ,$$

which is defined similar to the fluid Reynolds number Re, but is computed using the particle diameter as characteristic length and the difference of the particle and the fluid velocity as characteristic velocity. For a small particle Reynolds number the surrounding flow can be assumed to be laminar, while for higher Reynolds numbers the flow becomes turbulent and vortices form behind the particle.

Reynolds number: $\mathrm{Re} = \frac{LU}{\nu}$

The second quantity is the *Stokes number*

$$\mathrm{St} = \frac{\tau^{\mathrm{P}}}{\tau^{\mathrm{F}}} ,$$

which is the ratio of the characteristic response time of the particle $\tau^{\mathrm{P}} := (2\rho^{\mathrm{P}}R^2)/(9\nu)$ and the fluid $\tau^{\mathrm{F}} := L/U$. Using the definitions of the characteristic times we can reformulate the Stokes number as

$$\mathrm{St} = \frac{2}{9}\frac{\rho^{\mathrm{P}}R^2}{\nu}\frac{U}{L} , \tag{3.3}$$

where $U > 0$ is the characteristic velocity. For $\mathrm{St} \ll 1$ the particle's response time is much shorter than the characteristic time of the flow field and the particle has sufficient time to adjust to the fluid flow. On the other hand, for $\mathrm{St} \gg 1$ the particle has almost no time to adjust and will be hardly affected by the fluid phase.

3.1.1 *Fluid-Particle Forces*

Basset [12], Boussinesq [18] and Oseen [141] started to describe the motion of a sedimenting sphere under gravity in a fluid that was otherwise in rest. Tchen [175] picked up on their research and first proposed an equation of motion for an inertial particle in non-uniform flow by rewriting the Basset–Boussinesq–Oseen equation in a reference system moving with a small fluid parcel. His work has again been discussed and revised in several articles [22, 36, 126]. Stated below is today's best accepted version by Maxey and Riley [127], including a correction in the virtual mass term by Auton, Hunt and Prud'Homme [8]. The force on a particle submerged in a Newtonian fluid can be modelled as

Alfred Barnard Basset *(1854–1930) English mathematician*
Joseph Valentin Boussinesq *(1842–1929) French mathematician*
Carl Wilhelm Oseen *(1879–1944) Swedish theoretical physicist*

$$\begin{aligned}
M\mathrm{d}_t\underline{U}(t) &= (M-M^{\mathrm{F}})\underline{g} && (3.4a)\\
&\quad + M^{\mathrm{F}}\mathrm{D}_t\underline{u}^{\mathrm{F}}(\underline{X}(t),t) && (3.4b)\\
&\quad - \frac{1}{2}M^{\mathrm{F}}\left(\mathrm{D}_t\underline{U}(t) - (\mathrm{d}_t + \frac{1}{10}R^2\Delta)\underline{u}^{\mathrm{F}}(\underline{X}(t),t)\right) && (3.4c)\\
&\quad - 6\pi R\mu\left(\underline{U}(t) - (1+\frac{1}{6}R^2\Delta)\underline{u}^{\mathrm{F}}(\underline{X}(t),t)\right) && (3.4d)\\
&\quad - 6\pi R^2\mu\int_0^t \frac{\mathrm{d}_\tau\left(\underline{U}(\tau) - (1+\frac{1}{6}R^2\Delta)\underline{u}^{\mathrm{F}}(\underline{X}(\tau),\tau)\right)}{\sqrt{\pi\nu(t-\tau)}}\,\mathrm{d}\tau , && (3.4e)
\end{aligned}$$

where $M^{\mathrm{F}} = 4/3\,\pi R^3\rho^{\mathrm{F}}$ denotes the fluid mass displaced by the particle, $\mathrm{d}_t\underline{u}^{\mathrm{F}} = \partial_t\underline{u}^{\mathrm{F}} + \underline{u}^{\mathrm{F}}\nabla\cdot\underline{u}^{\mathrm{F}}$ denotes the material derivative with respect to a fluid parcel and $\mathrm{D}_t\underline{u}^{\mathrm{F}} = \partial_t\underline{u}^{\mathrm{F}} + \underline{U}\nabla\cdot\underline{u}^{\mathrm{F}}$ denotes the material derivative

with respect to the particle velocity. This model has been considered as the definitive study of the equation of motion of a solid sphere under creeping flow conditions [129]. Its derivation is based on the assumptions that a non-rotating, spherical and rigid sphere moves through an initially undisturbed, infinite fluid domain at small particle Reynolds numbers i.e. $Re^P \ll 1$. The Laplacian terms in (3.4c)–(3.4e) are accounted to Faxén, a student of Oseen's, and treat non-uniform effects in the flow field. They scale as $R^2/L^2 \ll 1$ and can be neglected in many practical applications [129].

The Maxey–Riley Equation (3.4) with $\underline{U}(t) = d_t\underline{X}(t)$ and for given $\underline{u}^F$ is an implicit integro-differential equation of second order; the basic properties of its solution have been explored by Farazmand and Haller [53]. In their work they prove existence and uniqueness of mild solutions and show that the results extend to strong solutions for special initial conditions.

We now give an overview of the fluid forces appearing in the above equation. These are gravitational and buoyancy force $\underline{F}^g$ (3.4a), force due to pressure gradient $\underline{F}^p$ (3.4b) , virtual mass effect $\underline{F}^{vm}$ (3.4c), Stokes drag force $\underline{F}^{St}$ (3.4d) and Basset Force $\underline{F}^B$ (3.4e). For a more detailed description of the forces than the following, we refer to Böttner [17], Lantermann [115] and Crowe, Sommerfeld and Tsuji [38] or the original articles.

GRAVITATIONAL AND BUOYANCY FORCE

The gravitational force accelerates the particle in direction of gravity. The buoyancy force opposes the gravity and results from the displacement of fluid mass by the particle (Archimedes' principle). We merge both forces to

$$\underline{F}^g = V^P \left(\rho^P - \rho^F\right) \underline{g} \,, \tag{3.5}$$

where $\underline{g} \in \mathbb{R}^d$ is acceleration due to gravity and $V^P = 4/3\pi R^3$ the volume of the particle. One can see that for a small density fraction $\rho^F/\rho^P \ll 1$ the buoyancy part is small to negligible. The buoyancy force is a special case of the following pressure-gradient force.

PRESSURE-GRADIENT FORCE

If a particle is moving in a non-isobar current, a non-homogeneous pressure distribution is imposed on the particle surface. Assuming a constant pressure gradient ∇p a force $\underline{F}^p$ in opposite direction of the pressure gradient occurs

$$\underline{F}^p = -\frac{M}{\rho^P}\nabla p \,,$$

which is proportional to the ratio of the particle mass M^P and density ρ^P. Solving the NSE for the pressure gradient one obtains

$$-\frac{1}{\rho^F}\nabla p = d_t\underline{u}^F - \mu\Delta\underline{u}^F \,.$$

Assuming constant shear on the particle surface the viscous term vanishes and one obtains

$$\underline{F}^p = M d_t\underline{u}^F \,.$$

VIRTUAL MASS EFFECT

Alongside the acceleration of a particle the encompassing fluid is accelerated, which results in an additional inertial mass $M_v > 0$. Crowe, Sommer-

feld and Tsuji [38] compute the total kinetic energy of the fluid surrounding the sphere as

$$E^{vm} = \frac{1}{2}\rho^F \int_V (\underline{u}^F(\underline{x}, t))^2 \, d\underline{x} \, ,$$

where the integral is taken over "all the fluid"[38]. Assuming the fluid velocity can be expressed as the derivative of a potential function $\phi : \Omega \times I \to \mathbb{R}$

$$\underline{u}^F(\underline{x}, t) = \nabla\phi(\underline{x}, t) \, ,$$

the kinetic energy can be reformulated

$$E^{vm} = \frac{1}{2}\rho^F \int_V (\nabla\phi(\underline{x}, t))^2 \, d\underline{x} \, .$$

Using the continuity Equation (3.1b) and Gauss theorem the integral is transformed into an integral over the particle boundary

$$E^{vm} = \frac{1}{2}\rho^F \int_S \phi(\nabla\phi \cdot \underline{n}) d\underline{s} \, .$$

Substituting ϕ by the potential function for a sphere moving through a fluid, the total kinetic energy can be found to

$$E^{vm} = \frac{\pi\rho^F R^3 (\underline{u}^F)^2}{3} \, .$$

The virtual mass force can than be obtained by differentiation with respect to time

$$\underline{\hat{F}}^{vm} = \frac{M^F}{2} d_t \underline{u}^F \, ,$$

where $M^F = 4/3\,\pi R^3$ is the fluid mass displaced by the particle. This is the force of the particle on the fluid. However the relative acceleration between particle and fluid has to be considered, and Auton, Hunt and Prud'Homme [8] found that the fluid acceleration should be derived with respect to the fluid

$$\underline{F}^{vm} = \frac{M^F}{2} \left(d_t \underline{u}^F - D_t \underline{U} \right) \, .$$

The term *virtual mass* becomes apparent by looking at the particle's equation of motion

$$M D_t \underline{U} = \frac{M^F}{2} \left(d_t \underline{u}^F - D_t \underline{U} \right)$$

and adding the rightmost term on both sides

$$(M + \frac{M^F}{2}) D_t \underline{U} = \frac{M^F}{2} d_t \underline{u}^F \, .$$

Therefore the accelerated body behaves as having an additional *virtual mass* of half the displaced fluid mass.

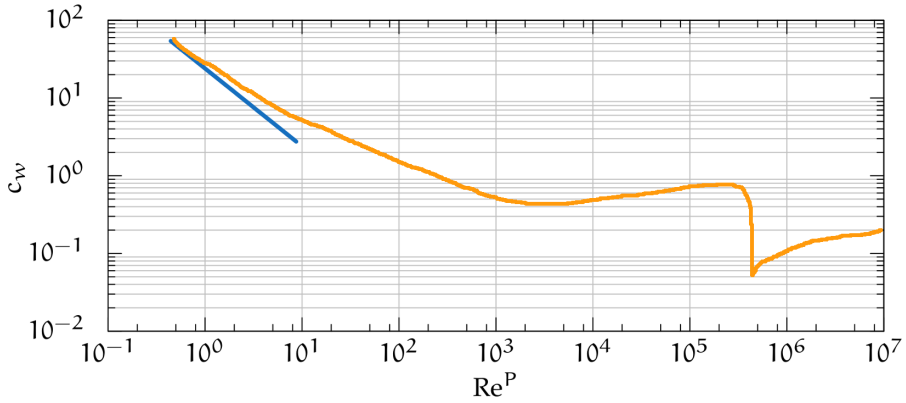


Figure 3.1: Drag coefficient c_W versus the Reynolds number Re, — standard drag curve, — Stokes drag.

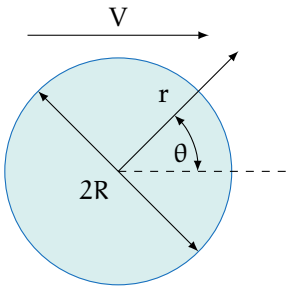


Figure 3.2: The coordinate system used by Stokes.

DRAG FORCE

The steady-state drag is the force acting if there is vanishing relative acceleration between the particle and the fluid. It is given by the equation

$$\underline{F}^d = \frac{1}{2}\rho^F c_w A \|\underline{u}^F - \underline{U}\|_2 (\underline{u}^F - \underline{U}) \,, \tag{3.6}$$

with the projected area of the particle $A > 0$ and the dimensionless drag coefficient $c_w > 0$. For non-rotating spherical objects in incompressible flow the drag coefficient is a function in the particle Reynolds number, $c_w = c_w(Re^P)$. It is the result of numerous experimental observations, and its graph is shown in Figure 3.1.

For small particle Reynolds numbers, $Re^P < 1$, we can assume a Stokesian flow regime, where Stokes Equations (2.9) hold. Stokes [170] first applied this equations to the flow over a sphere. By introducing a polar coordinate system he obtained the solution [38]

$$\underline{u}(\theta, r) = V\left(-\sin(\theta)\left(1 - \frac{3}{4}\frac{R}{r} - \frac{1}{4}\left(\frac{R}{r}\right)^3\right), \cos(\theta)\left(1 - \frac{3}{2}\frac{R}{r} - \frac{1}{2}\left(\frac{R}{r}\right)^3\right)\right) .$$

The polar coordinate system used by Stokes is illustrated in Figure 3.2, with the free stream velocity $V > 0$. It is easy to see that for the particle surface $r = R$ the velocity is zero, whereas for $r \to \infty$ the velocity equals the uniform flow velocity $\underline{u}^F$. With it, the hydrodynamic force on the particles surface can be calculated

$$\underline{F}^{St} = 6\pi\mu R(\underline{u}^F - \underline{U}) \,. \tag{3.7}$$

This particular force is named *Stokes drag* after its discoverer and can be obtained from the general drag equation (3.6) for

$$c_w = \frac{24}{Re^P} = \frac{24\mu}{2R\left\|\underline{U} - \underline{u}^F\right\|_2} .$$

For Reynolds numbers up to 5, Oseen [142] extended Stokes' results by including first-order inertial effects [38] and obtained

$$c_w = \frac{24}{Re^P}(1 + \frac{3}{16}Re^P) .$$

For particles with diameter of the same scale as the molecular free path of the surrounding gas, Cunningham [41] used a gas kinetic approach to calculate the drag carried out onto a particle. His result takes into account the molecular free path $\lambda_l \in \mathbb{R}^+$ of the gas molecules

$$\underline{F}^{Cun} = \underline{F}^{St}(1 + 1.63\frac{\lambda_l}{R}) .$$

From a (semi-)empirical perspective many studies have been conducted to approximate the *standard drag curve*, Clift, Grace and Weber [35] alone lists twelve. Their results can be applied in a wide range of the particle Reynolds number.

BASSET FORCE

The Basset force [12] or history force

$$F_B = 6R^2\sqrt{\pi\rho^F\mu^F}\int_0^t \frac{1}{\sqrt{t-t'}}\frac{d}{dt}\left(\underline{u}^F - \underline{U}^P\right)\, dt'$$

accounts for viscous effects. It addresses the temporal delay in acceleration of the surrounding fluid and depends on the acceleration history up to present time.

SUMMARY

We introduced the Maxey–Riley Equation (3.4) as best accepted summary of hydrodynamic forces acting on submerged particles. Of its components we will apply gravitational force (3.5) and Stokes drag (3.7) in the applications below. All other components are small in comparison and can be neglected.

3.1.2 *Integration of Particle Trajectories*

After introducing the mathematical model of the dilute Lagrangian particle phase of a two component flow, we now propose numerical methods to solve the Newtonian equations of motion (3.2). First the backward Euler Method (bEM) is introduced and its limitations are analysed and demonstrated using an example. The limitations can be overcome by the subsequently introduced forward Euler Method (fEM). Additionally two Verlet algorithms are mentioned, which provide good numerical stability, reduce the error in comparison to the Euler methods and are used in Section 4.3.1. Finally and to close the section trilinear interpolation is explained. As the fluid velocity is computed on the discrete lattice only, and the particle trajectories can be continuous in the domain interpolation becomes necessary.

EULER METHODS

In the previous section a mathematical model of particle trajectories and fluid-particle forces was introduced. We now propose numerical methods to solve the particle equations of motion (3.2) beginning with Euler methods. Let $\underline{U} : I \to \mathbb{R}^d$ be a solution of the initial value problem

$$d_t \underline{U} = \underline{g}(t, \underline{U}) \qquad \text{for } t \in I\,, \tag{3.8}$$

$$\underline{U}(0) = \underline{U}_0\,, \tag{3.9}$$

with $\underline{g} : I \times \mathbb{R}^d \to \mathbb{R}^d$. For the discrete time interval $I_h = \{t_i = i\Delta_t \in I \mid i \in \mathbb{N}, \Delta_t \in \mathbb{R}^+\}$ with time step Δ_t, the above Ordinary Differential Equation (ODE) (3.8) can be solved by a forward Euler Method (fEM)

$$\underline{U}_{i+1} = \underline{U}_i + \Delta_t \underline{g}(t_i, \underline{U}_i) \qquad \text{for } t_i \in I_h\,, \tag{3.10}$$

which has the following global error:

THEOREM 4 (Global error of the forward Euler Method (fEM)). *Let* $I = [0, T)$ *and* $g : I \times \mathbb{R}^d \to \mathbb{R}^d$ *continuously differentiable and globally Lipschitz continuous with Lipschitz constant* L

$$\left\|\underline{g}(t, \underline{U}) - \underline{g}(t, \underline{Z})\right\|_2 \leqslant L \left\|\underline{U} - \underline{Z}\right\|_2 \qquad \textit{for all } t \in I \textit{ and } \underline{U}, \underline{Z} \in \mathbb{R}^d\,.$$

If $\underline{U}$ *is the unique solution of* (3.8) *and if* $\underline{U}_i, i = 1, \ldots, N$ *are approximations of the fEM at time* $t_i \in I$*, then*

$$\begin{aligned}\|\underline{U}(t_i) - \underline{U}_i\|_2 &\leqslant \frac{(1 + L\Delta_t)^i - 1}{2L} \|d_t^2 \underline{U}\|_{[0,T]} \Delta_t \\ &\leqslant \frac{e^{(LT)} - 1}{2L} \|d_t^2 \underline{U}\|_{[0,T]} \Delta_t\end{aligned}$$

holds for $i = 0, \ldots, N$. *Here* $\|d_t^2 \underline{U}\|_{[0,T]} = \max_{0 \leqslant t \leqslant T} \|d_t^2 \underline{U}\|_2$.

Proof. See Hanke-Bourgeois [75, Satz 74.1]. □

In many situations only Stokes drag force in the form of (3.7) is acting on a particle, and (3.8) becomes

$$\begin{aligned} d_t \underline{U}(t) &= \underline{g}(t, \underline{U}(t)) \\ &= \frac{1}{M} \underline{F}^{St}(t) \\ &= \frac{6\pi\mu R}{4/3\,\pi R^3 \rho^P} (\underline{u}^F(t) - \underline{U}(t)) \\ &= \frac{9}{2} \mu \frac{1}{R^2 \rho^P} (\underline{u}^F(t) - \underline{U}(t))\,. \end{aligned} \tag{3.11}$$

It is obvious, that the upper limit of the approximation error of the fEM

$$\|\underline{U}(t_i) - \underline{U}_i\|_2 \leqslant \frac{e^{(LT)} - 1}{2L} \frac{9}{2} \mu \frac{1}{R^2 \rho^P} \|d_t(\underline{u}^F - \underline{U})\|_{[0,T]} \Delta_t$$

not only depends on the time step Δ_t, but also on the particle radius and density, as well as the fluid viscosity.

Besides error estimation, stability is a key issue of numerical methods. Without loss of generality let $\underline{u}^F = 0$. Then (3.11) can be written in the form

$$d_t \underline{U}(t) = \lambda \underline{U}(t)\,, \tag{3.12}$$

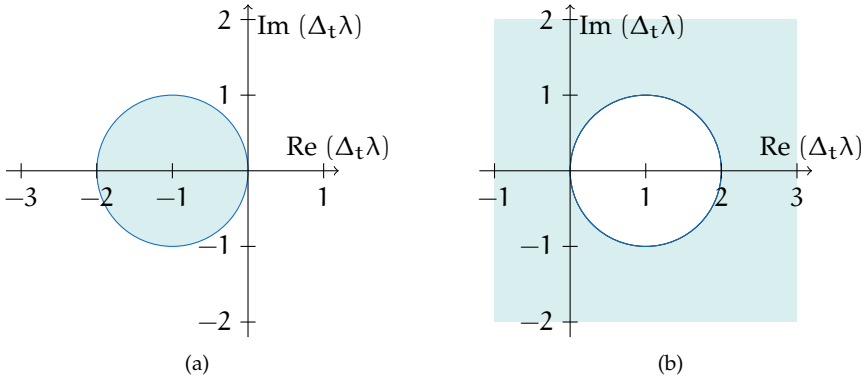


Figure 3.3: Stability region of the backward (3.3a) and forward (3.3b) Euler method, for the solution of Equation (3.12) and complex λ.

where $\lambda = -9\mu/(2R^2\rho^P)$. With the initial condition $\underline{U}(0) = \underline{U}_0$ the exact solution is

$$\underline{U}(t) = e^{\lambda t}\underline{U}_0 \,,$$

which is bounded for all $t > 0$, if $\lambda \leqslant 0$ and $\underline{U}(t) \to 0$ for $t \to \infty$, if $\lambda < 0$. The fEM applied to (3.12) returns

$$\underline{U}_{i+1} = \underline{U}_i + \Delta_t\lambda\underline{U}_i = (1 + \Delta_t\lambda)^{n+1}\underline{U}_0 \,.$$

The series $(\underline{U}_i)_i$ is bounded for $t > 0$, if $|1 + \Delta_t\lambda| \leqslant 1$ and $\underline{U}_i \to 0$ as $n \to \infty$ if $|1 + \Delta_t\lambda| < 1$ holds.

DEFINITION 3 (Stability region [85, Definition 10.3]). The stability region of a numerical method is denoted by

$$S = \{z \in \mathbb{C} \mid z = \Delta_t\,\lambda;\ \text{the method returns bounded solutions } (\underline{U}_n)_{n\geqslant 0},\ \text{if applied to } d_t\underline{U} = \lambda\underline{U},\ \underline{U}(0) = \underline{U}_0 \text{ with step size } \Delta_t\}\,.$$

The stability region $S_{bE} = \{|1 + \Delta_t\lambda| \leqslant 1\}$ of the fEM is illustrated in Figure 3.3a for $\lambda \in \mathbb{C}$.

In Section 3.6 particle deposition in a bronchial bifurcation is discussed. In this setup fine particles are transported by airflow. The dynamic viscosity of air is $\mu = 1.84 \cdot 10^{-05}\,\mathrm{kg/(m\,s)}$ and the particle density $\rho^P = 998.2\,\mathrm{kg/m^3}$. To obtain a stable LBM spatial discretisation and time discretisation are set to $h = 2.60 \cdot 10^{-2}\,\mathrm{m}$ and $\Delta_t = c \cdot h^2 = 3.48 \cdot 10^{-4}\,\mathrm{s}$. This simulation is included as an example in `OpenLB` since v1.0. Using these quantities, we now compute the smallest possible particle radius to still remain in the stability region of the fEM.

$$\begin{aligned}
&\iff && 1 \geqslant |1 + \Delta_t\lambda| && \\
&\iff && 1 \geqslant -1 - \Delta_t\lambda && \text{for } \Delta_t\lambda \leqslant -1 \\
&\iff && 2R^2 \geqslant \frac{9}{2}\mu\frac{1}{\rho^P}\Delta_t && \\
&\iff && 2R^2 \geqslant 2.87 \cdot 10^{-11} && \\
&\iff && R \geqslant 3.79 \cdot 10^{-6}\,[\mathrm{m}]\,. &&
\end{aligned}$$

Typical inhalable particles have diameter between 2.5 µm and 10 µm and can therefore not be simulated by these parameters. One possible solution

would be to use a smaller timestep, which comes with the costs of a higher resolution of the lattice, as time and space resolutions are coupled by $\Delta_t \propto h^2$. Another solution is to use a different integration method.

Besides the fEM, the backward Euler Method (bEM) can be used to solve Equations (3.8). While the fEM uses the current timestep t_i in the rightmost addend in its definition (3.10), the bEM uses implicitly determined values of $f(t_{i+1})$ at the upcoming timestep

$$\underline{U}_{i+1} = \underline{U}_i + \Delta_t \underline{g}(t_{i+1}, \underline{U}_{i+1}) \qquad \text{for } t_i \in I_h \,. \tag{3.13}$$

This implicit definition generally leads to a non-linear system of equations for each timestep $i = 0, 1, 2, \dots, N$.

THEOREM 5 (Global error of the backward Euler Method (bEM)). *Let* $I = [0, T]$ *and* $f : I \times \mathbb{R}^d \to \mathbb{R}^d$ *continuously differentiable and satisfying the one-sided Lipschitz condition*

$$\left(\underline{g}(t, \underline{U}) - \underline{g}(t, \underline{Z})\right)(\underline{U} - \underline{Z}) \leqslant \|\underline{U} - \underline{Z}\|_2^2$$

for all $(t, \underline{U}), (t, \underline{Z}) \in I \times \Omega$ *and one* $l \in \mathbb{R}$. *Additionally, let the timestep* $\Delta_t > 0$ *satisfies the condition* $\Delta_t l < 1$. *Then for the bEM the error estimate*

$$\|\underline{U}(t_i) - \underline{U}_i\|_2 \leqslant \frac{1}{2l}\left(\left(\frac{1}{1 - l\Delta_h}\right)^i - 1\right)\|d_t^2 \underline{U}\|_{[0,T]}\Delta_t$$

holds for all $t_i = i\Delta_t \in I$.

Proof. See Hanke-Bourgeois [75, Satz 75.2]. □

Applying (3.13) to (3.12) ends in

$$\underline{U}_{i+1} = \underline{U} + \Delta_t \lambda \underline{U}_{i+1} \,,$$

which has the numerical solution

$$\underline{U}_i = \left(\frac{1}{1 - \Delta_t \lambda}\right)^i \underline{U}_0 \,,$$

which is bounded for all $i \geqslant 0$, if

$$\frac{1}{|1 - \Delta_t \lambda|} \geqslant 1 \tag{3.14}$$

holds. The stability region of the bEM therefore is $S_{fE} = \{1/|1 - \Delta_t \lambda| \leqslant 1\}$ and is illustrated in Figure 3.3b. Substituting $\lambda = -9\mu/(2R^2\rho^P) < 0$ in (3.14) and using the same quantities as before

$$\begin{aligned}
&\iff && 1 \geqslant \frac{1}{|1 - \Delta_t \lambda|} \\
&\iff && 0 \leqslant -\Delta_t \lambda && \text{for } 1 \geqslant \Delta_t \lambda \\
&\iff && 0 \leqslant \Delta_t \frac{9}{2}\mu \frac{1}{R^2 \rho^P} \\
&\iff && 0 \leqslant \Delta_t \frac{1}{R^2} 3.79 \cdot 10^{-6}\,\mathrm{m} \,,
\end{aligned}$$

we find that the bEM is stable for all possible radii R. Given the case that only Stokes drag is employed and the fluid velocity is known for all $\underline{X}(t) \in \Omega$, the bEM turns out to be analytically solvable for $\underline{U}_{i+1}$

$$\begin{aligned}
\underline{U}_{i+1} &= \underline{U}_i + \frac{1}{M}\Delta_t \underline{F}^{St}(t_{i+1}) \\
&= \underline{U}_i + \frac{6\pi R\mu}{M}\Delta_t \left(\underline{u}^F(t_{i+1}) - \underline{U}_{i+1}\right) \\
&= \underline{U}_i + \Delta_t\lambda \left(\underline{u}^F(t_{i+1}) - \underline{U}_{i+1}\right) \\
\iff \qquad \underline{U}_{i+1}\,(1+\Delta_t\lambda) &= \underline{U}_i + \Delta_t\lambda \underline{u}^F(t_{i+1}) \\
\iff \qquad \underline{U}_{i+1} &= \frac{\underline{U}_i + \Delta_t\lambda \underline{u}^F(t_{i+1})}{1+\Delta_t\lambda} \,.
\end{aligned}$$

For both Euler methods the particle position equation (3.2b) is updated by a bEM.

The local approximation error of the particle position can be estimated by expanding $\underline{X}(t_i + \Delta_t)$ in its Taylor series

$$\underline{X}(t_i + \Delta_t) = \underline{X}(t_i) + \Delta_t d_t \underline{X}(t_i) + O(\Delta_t^2) \tag{3.15}$$

$$= \underline{X}(t_i) + \Delta_t \underline{U}(t_i) + O(\Delta_t^2)\,, \tag{3.16}$$

and is found to be of order $O(\Delta_t^2)$.

VERLET METHODS

Loup Verlet *(1931) French physicist, pioneer in MD*

Besides the fEM, Verlet integration [181, 182] is another common method used in the Discrete Element Method (DEM). It can be obtained by expanding the particle position at timesteps $t + \Delta_t$ and $t - \Delta_t$ in its Taylor series

$$\underline{X}(t+\Delta_t) = \underline{X}(t) + \Delta_t d_t \underline{X}(t) + \frac{1}{2}\Delta_t^2 d_t^2 \underline{X}(t) + \frac{1}{6}\Delta_t^3 d_t^3 \underline{X}(t) + O(\Delta_t^4) \tag{3.17}$$

$$\underline{X}(t-\Delta_t) = \underline{X}(t) - \Delta_t d_t \underline{X}(t) + \frac{1}{2}\Delta_t^2 d_t^2 \underline{X}(t) - \frac{1}{6}\Delta_t^3 d_t^3 \underline{X}(t) + O(\Delta_t^4) \tag{3.18}$$

and adding both equations

$$\underline{X}(t+\Delta_t) = 2\underline{X}(t) - \underline{X}(t-\Delta_t) + \Delta_t^2 d_t^2 \underline{X}(t) + O(\Delta_t^4)\,.$$

It can be seen, that the jerk (third derivative of the position with respect to time) terms cancel each other and the truncation error is of order $O(\Delta_t^4)$. The particle velocity does not have to be explicitly computed in this method. However, it is often necessary to evaluate the particle forces or to compute the energy of the system. The velocities can be computed from the positions by

$$\underline{U}(t) = \frac{\underline{X}(t+\Delta_t) - \underline{X}(t-\Delta_t)}{2\Delta_t}\,.$$

This has several drawbacks. Firstly, the error associated to this expression is of order $O(\Delta_t^2)$ rather than $O(\Delta_t^4)$. Secondly, it is not self-starting, as for computing $\underline{X}(\Delta_t)$, besides $\underline{X}(0)$ also $\underline{X}(-\Delta_t)$ has to be known, which is generally not the case. Finally, the velocity at timestep t can only be computed at the subsequent timestep $t + \Delta_t$.

An improved implementation of the same algorithm computes the velocity first. It is therefore known as *velocity Verlet*. Substituting t by $t + \Delta_t$ in (3.18) and adding the obtained equation to (3.17) yields

$$d_t \underline{X}(t) = d_t \underline{X}(t+\Delta_t) - \frac{1}{2}\Delta_t^2 \left(d_t^2 \underline{X}(t) + d_t^2 \underline{X}(t+\Delta_t)\right)$$

or

$$\underline{U}(t) = \underline{U}(t+\Delta_t) - \frac{1}{2M}\Delta_t^2\left(\underline{F}(t) + \underline{F}(t+\Delta_t)\right) \ .$$

The particle positions are obtained by

$$\underline{X}(t+\Delta_t) = \underline{X}(t) + \Delta_t\underline{U}(t) + \frac{1}{2M}\Delta_t^2\underline{F}(t).$$

Carl Runge ☾ *(1856–1927) German mathematician*
Wilhelm Kutta *(1867–1944) German mathematician*
John Couch Adams ☾ *(1819–1892) English mathematician*
Francis Bashforth *(1819–1912) British mathematician*

The velocity Verlet algorithm removes all the disadvantages of the previous position Verlet algorithm. The advantage of the smaller local error in comparison to the Euler methods has to be paid for by storing the force on each particle of the last timestep, i.e. the memory requirement increases.

Alternative algorithms, such as Predictor-Corrector, Runge-Kutta [111, 156] or Adams-Bashforth methods exist which allow to solve the particle trajectories using larger timesteps at the cost of higher memory usage. See e. g. Matuttis and Chen [125] for a review. However, due to the explicit character of the LBM and the integration of the particle tracking algorithm in an already existing Lattice Boltzmann (LB) code, small timesteps are available; using larger timesteps than the LBM would dismiss available fluid information.

3.1.3 *Interpolation of Fluid Velocity*

As the particle position $\underline{X} : I \to \Omega$ moves in the continuous domain Ω and information on the fluid velocity can only be computed on lattice nodes $\underline{x}_i \in \Omega_h$ interpolation of the fluid velocity is necessary every time fluid–particle forces are computed. Let $\underline{u}_i^F = \underline{u}^F(\underline{x}_i)$ be the computed solution of the NSE at lattice nodes $\underline{x}_i$. Let $p \in P_n$ be the interpolating polynomial of order n with $p(\underline{x}_i) = \underline{u}_i^F$ and $\overline{(\underline{x}_0, \dots \underline{x}_n)}$ the smallest interval containing all points in the brackets. Furthermore, let $C^n[\underline{a}, \underline{b}]$ be the vector space of continuous functions that have continuous first n derivatives in $[\underline{a}, \underline{b}]$. Then the interpolation error of the polynomial interpolation is stated by the following theorem.

THEOREM 6 (Interpolation error). *Let* $\underline{u} \in C^{n+1}[\underline{a}, \underline{b}]$, $\underline{a}, \underline{b} \in \Omega$. *Then for every* $\underline{x} \in [\underline{a}, \underline{b}]$ *there exists one* $\hat{\underline{x}} \in \overline{(\underline{x}_0, \dots \underline{x}_n, \underline{x})}$, *such that*

$$\underline{u}^F(\underline{x}) - p_n(\underline{x}) = \frac{d_{\underline{x}}^{n+1}\underline{u}^F(\hat{\underline{x}})}{(n+1)!}\prod_{j=0}^{n}(\underline{x} - \underline{x}_j) \tag{3.19}$$

holds.

Proof. See Rannacher [150, Satz 2.3] or Deuflhard, Bornemann and Hohmann [45, Satz 7.16]. □

Using linear ($n = 1$) interpolation for the fluid velocity between two neighbouring lattice nodes $\underline{a} = \underline{x}_0 \in \Omega_h, \underline{b} = \underline{x}_1 \in \Omega_h, \|\underline{x}_1 - \underline{x}_0\|_2 = h$ clearly the following holds

$$\begin{aligned} f(\underline{x}) - p_1(\underline{x}) &= \frac{1}{2}d_{\underline{x}}^2\underline{u}^F(\hat{\underline{x}})(\underline{x} - \underline{x}_0)(\underline{x} - \underline{x}_1) \\ &\leqslant \frac{1}{2}d_{\underline{x}}^2\underline{u}^F(\hat{\underline{x}})h^2 \end{aligned}$$

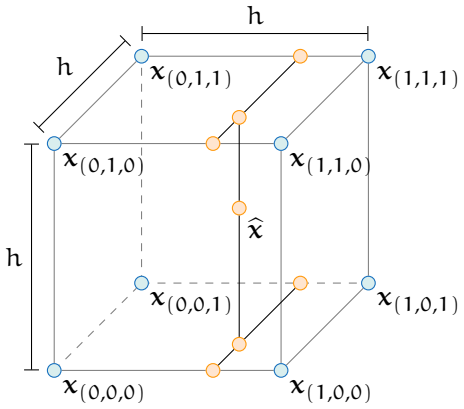


Figure 3.4: Trilinear interpolation.

and the approximation error of the linear interpolation is of order $\mathcal{O}(h^2)$. In the following we give reason why this order of interpolation is sufficient.

Let's assume there exists an ideal error law of the form

$$\|\underline{u}^F - \underline{u}^{F^*}\|_{L^2(\Omega_h)} = ch^\alpha ,$$

for the discrete solution $\underline{u}^F_h$ obtained by an LBM with lattice spacing h and the analytic solution $\underline{u}^{F^*}$. Then $\alpha \in \mathbb{R}^+$ is the to be determined order of convergence. We further define the relative error

$$\mathrm{Err}_h = \frac{\|\underline{u}^F_h - \underline{u}^{F^*}\|_{L^2(\Omega_h)}}{\|\underline{u}^{F^*}\|_{L^2(\Omega_h)}} .$$

The ratio of the error laws of two distinct lattice spacings h_i and h_j, forms the Experimental Order of Convergence (EOC) as

$$\mathrm{EOC}_{i,j} = \frac{\ln(\mathrm{Err}_{h_i}/\mathrm{Err}_{h_j})}{\ln(h_i/h_j)} . \tag{3.20}$$

With this Krause [110, Chapter 2.3] determines an EOC of approximately $\alpha \approx 2$ for the discrete solution towards the analytic solution of a stationary flow in the unit cube governed by the incompressible NSE. Therefore the order of converge of the fluid velocity obtained by an LBM can be assumed to be $\mathcal{O}(h^2)$. This conclusion is backed up by the theoretical results obtained by [24]. This leads to the assumption that an interpolation scheme of higher order than 2 would not be exhausted as the error of the incoming data is too large.

The interpolation is implemented as a trilinear interpolation using the eight nodes surrounding the particle. Let the point of interpolation $\hat{\underline{x}} \in [x_{(0,0,0)}, x_{(1,1,1)}]$ be in the cube spanned by the lattice nodes $\underline{x}_{(0,0,0)}$ and $\underline{x}_{(1,1,1)}$, see Figure 3.4 for an illustration. We will denote by

$$\underline{d} = (d_0, d_1, d_2)^T = \hat{\underline{x}} - \underline{x}_{(0,0,0)}$$

the distance of the particle to the next smaller lattice node. The fluid velocities at the eight corners are named accordingly $\underline{u}_{(i,j,k)}$, $i,j,k \in \{0,1\}$. The

Listing 3.1: LBM algorithm including Lagrangian particles.

```
0  for t ∈ I_h {
     for x ∈ Ω_h {
       for i = 0,...,q−1 {
         f̃_i(x,t) = f_i(x,t) − (1/τ)(f_i(x,t) − M(x,t))
         f_i(x + h²ξ_i, t + h²) = f̃_i(x,t)
       }
     }
     communicate overlap

     foreach particle {
10     compute F(t) = Σ_i F_i(t)
       U(t + h²) = U(t) + h²F(t)/M
       X(t + h²) = X(t) + h²U(t + h²)
       compute boundaries
     }
15   communicate particles
   }
```

trilinear interpolation is executed by three consecutive linear interpolations in the three different space directions. First we interpolate along the x-axis

$$
\begin{aligned}
u_{(d,0,0)} &= u_{(0,0,0)}(h - d_0) + u_{(1,0,0)} d_0 \\
u_{(d,1,0)} &= u_{(0,1,0)}(h - d_0) + u_{(1,1,0)} d_0 \\
u_{(d,0,1)} &= u_{(0,0,1)}(h - d_0) + u_{(1,0,1)} d_0 \\
u_{(d,1,1)} &= u_{(0,1,1)}(h - d_0) + u_{(1,1,1)} d_0
\end{aligned}
$$

followed by interpolation along the y-axis

$$
\begin{aligned}
u_{(d,d,0)} &= u_{(d,0,0)}(h - d_1) + u_{(d,1,0)} d_1 \\
u_{(d,d,1)} &= u_{(d,0,1)}(h - d_1) + u_{(d,1,1)} d_1
\end{aligned}
$$

and finally in direction of the z-axis

$$
\underline{u}(\hat{\underline{x}}) = u_{(d,d,d)} = u_{(d,d,0)}(h - d_2) + u_{(d,d,1)} d_2 .
$$

3.2 IMPLEMENTATIONAL ASPECTS

The particle solver proposed in the previous section has been integrated in the LBM algorithm and implemented in the present version 1.0 of `OpenLB`. Listing 3.1 shows the advancements compared to Listing 2.1, where lines 9–16 have been added, which are necessary to integrate the Lagrangian particle phase. The dilute particles are assumed to be independent and can therefore be computed independently in one `for` loop. First, all forces acting on a particle have to be computed and summed up. With the force known, it is possible to advance the particle velocity and position by one timestep. After all particles have been updated, their affiliation to the current cuboid has to be checked, and the particles are redistributed if necessary.

Similar to the previous chapter on LBM, we will pick out aspects concerning the implementation of the dilute particle phase. We first present a generic implementation concept, which handles all particle dynamics. Then we introduce two new parallelisation schemes for the communication of particles between cuboids and analyse and compare their computational costs.

3.2.1 *The Class* `SuperParticleSystem3D`

The implementation of the particle phase follows an hierarchical ansatz, similar to the `Cell` → `BlockLattice3D` → `SuperLattice3D` ansatz used for the implementation of the LBM. The equivalent class in the context of Lagrangian particles are `Particle3D` → `ParticleSystem3D` → `SuperParticleSystem3D`. The class `Particle3D` allocates memory for the variables of one single particle, such as its position, velocity, mass, radius and the force acting on it. It also provides the function `bool getActive()`, which returns the activity state of the particle. Active particles' positions are updated during the simulation, in contrast to non-active particles, which are only used for particle-particle interaction. The class `Particle3D` is intended to be inherited from, in order to provide additional properties, such as electric or magnetic charge. The particles in the domain of a specific `BlockLattice3D` are combined in the class `ParticleSystem3D`. Finally the class `SuperParticleSystem3D` combines all `ParticleSystem3D`s and handles the transfer of particles between them.

`Cell`
container for data of one lattice node
`BlockLattice3D`
stores the `Cell`s in one $\widetilde{\Omega}_h^l$
`SuperLattice3D`
organises several $\widetilde{\Omega}_h^l$ *across PU borders*

The concept of the class `SuperParticleSystem3D` is to provide an easily adaptable framework for simulation of a large number of particles arranged in and interacting with a fluid. In this context *easily adaptable* means that simulated forces and boundary conditions are implemented in a modular manner, such that they are easily exchangeable. Development of new forces and boundary conditions can be readily done by inheritance of provided base classes. Particle–particle interaction can be activated if necessary and deactivated to decrease simulation time. The contact detection algorithm is interchangeable. This section introduces the `SuperParticleSystem3D` and the mentioned properties in more detail.

The class `SuperParticleSystem3D` is initialised by a call to the constructor

```
SuperParticleSystem3D(CuboidGeometry3D<T>& cuboidGeometry,
          LoadBalancer<T>& loadBalancer, SuperGeometry3D<T>& sGeometry,
          LBconverter<T>& conv);
```

simultaneously on all PUs. During the construction each PU instantiates one `ParticleSystem3D` for each local cuboid. Subsequently for each `ParticleSystem3D` a list of the ranks of PUs holding neighbouring cuboids is created.

Particles can be added to the `SuperParticleSystem3D` by a call to one of the `addParticle()` functions.

```
/// Add a Particle to SuperParticleSystem
void addParticle(PARTICLETYPE<T> &p);
/// Add a number of identical Particles equally distributed in a given
    IndicatorF3D
void addParticle(IndicatorF3D<T>& ind, T mas, T rad, int no=1, std::
    vector<T> vel={0.,0.,0.});
/// Add a number of identical Particles equally distributed in a given
    Material Number
void addParticle(std::set<int>  material, T mas, T rad, int no=1, std::
    vector<T> vel={0.,0.,0.});
/// Add Particles form a File. Save using saveToFile(std::string name)
void addParticlesFromFile(std::string name, T mass, T radius);
```

Currently there are four implementations of this class. The first adds single predefined particles, the second and third add a given number of equally distributed particles of the same mass and radius in an area that can be defined by either a set of material numbers or an indicator function. The initial particle velocity can be set optionally. Additionally particles can be

added from an external file containing their positions. In all cases the assignment to the correct ParticleSystem3D is carried out internally. Particle forces and boundaries are implemented by the base classes Force3D and Boundary3D.

```
template<typename T, template<typename U> class PARTICLETYPE>
class Force3D {
public:
  Force3D();
  virtual void applyForce(typename std::deque<PARTICLETYPE<T> >::
      iterator p, int pInt, ParticleSystem3D<T, PARTICLETYPE>& psSys)
      =0;
}
```

```
template<typename T, template<typename U> class PARTICLETYPE>
class Boundary3D {
public:
  Boundary3D();
  virtual void applyBoundary(typename std::deque<PARTICLETYPE<T> >::
      iterator p, int pInt, ParticleSystem3D<T, PARTICLETYPE>& psSys)
      =0;
}
```

Both classes are intended to be derived from in order to implement force and boundary specialisations, e. g. the introduced forces in Subsection 3.1.1. The key function in both classes are `applyForce()` and `applyBoundary()`, which are called in each timestep of the main LBM loop. Force3D and Boundary3D specialisations are added to the SuperParticleSystem3D, by passing a pointer to a class instantiation via a call to the respective function.

```
/// Add a force to system
void addForce(std::shared_ptr<Force3D<T, PARTICLETYPE> > f);
/// Add a boundary to system
void addBoundary(std::shared_ptr<Boundary3D<T, PARTICLETYPE> > b);
```

Both functions add the passed pointer to a list of forces and boundaries, which will be looped over during the simulation step. A contact detection algorithm can be added.

```
/// Set contact detection algorithm for particle-particle contact.
void setContactDetection(ContactDetection<T, PARTICLETYPE>&
    contactDetection);
```

See Section 4.1.1 for more details.

Finally one timestep is computed by a call to the function `simulate()`.

```
template<typename T, template<typename U> class PARTICLETYPE>
void SuperParticleSystem3D<T, PARTICLETYPE>::simulate(T dT)
{
  for (auto pS : _pSystems) {
    pS->_contactDetection->sort();
    pS->simulate(dT);
    pS->computeBoundary();
  }
  updateParticleDistribution();
}
```

This function contains a loop over the local ParticleSystem3Ds calling the local sorting algorithm and the functions ParticleSystem3D`::simulate()` and ParticleSystem3D`::computeBoundary()`. The sorting algorithm determines potential contact between particles according to the set ContactDetection.

The inline function `ParticleSystem3D::simulate()` first calls the local function `ParticleSystem3D::computeForce()`.

```
inline void simulate(T dT) {
  _pSys->computeForce();
  _pSys->explicitEuler(dT);
}
```

The function `ParticleSystem3D::computeForce()` consists of a loop over all particles stored by the calling `ParticleSystem3D`.

```
template<typename T, template<typename U> class PARTICLETYPE>
void ParticleSystem3D<T, PARTICLETYPE>::computeForce()
{
  typename std::deque<PARTICLETYPE<T> >::iterator p;
  int pInt = 0;
  for (p = _particles.begin(); p != _particles.end(); ++p, ++pInt) {
    if (p->getActive()) {
      p->resetForce();
      for (auto f : _forces) {
        f->applyForce(p, pInt, *this);
      }
    }
  }
}
```

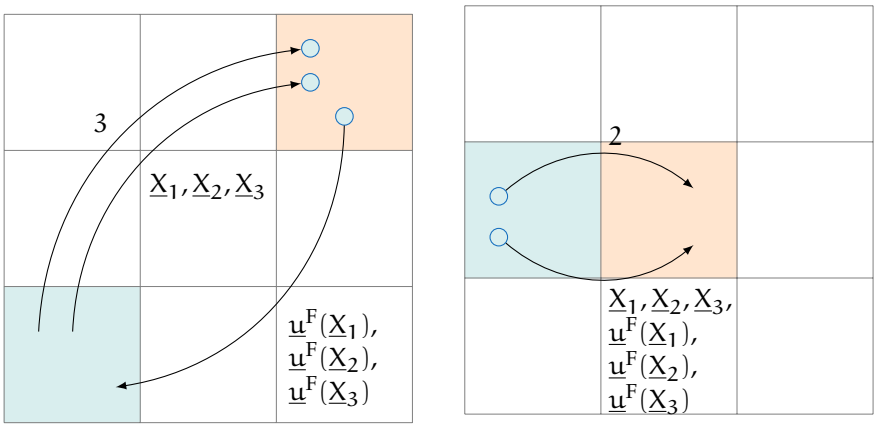
If the particle state is active, its force variable is reset to zero. Then the value computed by each previously added particle force is added to the particle's force variable. Finally, the particle velocity and position is updated by one step of an integration method – here the backward Euler Method (bEM).

Returning to `SuperParticleSystem3D::simulate(T dT)`, a call to the function `ParticleSystem3D::computeBoundary()` is the next command in the loop, which has the same structure as the `ParticleSystem3D::computeForce()`. After executing the loop, the function `updateParticleDistribution()` is called, which redistributes the particles over the `ParticleSystem3D`s according to their updated position. A detailed description of this function is provided at the end of the next section.

3.2.2 *Parallelisation of the Particle Phase*

This section is devoted to the parallelisation of the Lagrangian particle phase based on the domain decomposition of the LBM. If after an update of the particle position the Processing Unit (PU) computing the particle trajectory and the PU computing the surrounding fluid do not match, information has to be transferred between the PUs. The following two paragraphs introduce two parallelisation strategies, which have originally been published in Henn et al. [82].

Parallelisation of the particle phase is trivial as long as particle-particle interaction is omitted and particle forces are independent of the fluid. However, introducing fluid-particle interaction leads to a dilemma, because the fluid velocity at the position of a particle has to be communicated by the PU at each time step. Distributing particles equally on all ranks leads to short computation times but increased communication, while distribution according to their position in the fluid leads to low communication and increased computing times if particles are distributed non-homogeneously, i.e. if they are concentrated in a subdomain assigned to one PU. Two different strategies for parallel computation of particle trajectories are presented in the follow-



(a) *Load optimal strategy*. Particles are evenly distributed over all PUs, leading to particles computed by a PU (red) not holding the respective subdomain $\widetilde{\Omega}_h^l$ (blue). Position $\underline{X}_i$ and velocity $\underline{u}^F(\underline{X}_i)$ have to be communicated in a three step procedure.

(b) *Communication optimal strategy*. Particles are stored and computed on the PU holding the corresponding subdomain $\widetilde{\Omega}_h^l$. Particles leaving a subdomain are communicated in two steps.

Figure 3.5: Parallelisation strategies for dilute particle flow simulations.

ing. Both of them are based on domain decomposition of the position space Ω_h, which has been introduced in Section 2.4.2. The *load optimal strategy* is driven by the idea to reduce computing time, while the *communication ideal strategy* intends to reduce the communication effort. We will now discuss both strategies in detail for respective worst case scenarios.

The *load optimal strategy* (Figure 3.5a) aims at achieving a balance of computational load. Therefore $N \in \mathbb{N}$ particles are spread over $P \in \mathbb{N}$ PUs, where they remain for the entire simulation. Hence every unit is responsible for the storage and calculation of N/P particles, which leads to a computational effort of $\mathcal{O}(N/P)$ for every iteration of the lattice Boltzmann loop. During the update of the particle velocity, information about the fluid velocity $\underline{u}^F(\underline{X}(t), t)$ becomes necessary and has to be communicated. We assume that processor A (red), which is in need of the velocity data, knows which processor holds $\underline{u}^F(\underline{X}(t), t)$. The latter one is called processor B (blue). The data exchange is conducted in three steps. During the first step A sends the number of particles in its responsibility, which stay in the fluid domain of processor B, to processor B. In the second step, A sends the positions $\underline{X}$ of the respective particles, whereupon B returns the fluid velocities $\underline{u}^F(\underline{X}(t), t)$ in step three. In a worst case scenario none of the particles are assigned to the processor holding the respective fluid domain. Looking at the effort in communication, we see that in every step P PUs send a message to all other $P-1$ PUs, which sums up to $3P(P-1)$ communications per iteration. In the first step one datum is transmitted by every node ($\mathcal{O}(P)$). Hence N/P positions have to be submitted by the P PUs, resulting in $(N/P)P$ communicated informations in the second step. In the last step the $(N/P)P$ velocities are returned. Altogether the cost of communication aggregates to

$$\mathcal{O}(P(P-1)) + \mathcal{O}(P(P-1)N) + \mathcal{O}(P(P-1)N) = \mathcal{O}(P^2N).$$

The *communication optimal strategy* (cf. Figure 3.5b) targets the reduction of communication costs. Therefore particles are always stored and computed on the PU associated with the connected fluid subdomain $\widetilde{\Omega}_h^l$. In a worst

COSTS		LOAD IDEAL	COMMUNICATION IDEAL
communication	1^{st} step	$\mathcal{O}(P^2)$	$\mathcal{O}(P)$
	2^{nd} step	$\mathcal{O}(P^2N)$	$\mathcal{O}(P+N)$
	3^{rd} step	$\mathcal{O}(P^2N)$	
computation		$\mathcal{O}(N/P)$	$\mathcal{O}(N)$

Table 3.2: Summary of the computation and communication costs of the two parallel strategies

case scenario, all particles are on one PU, leading to computational costs of $\mathcal{O}(N)$ for this particular node, while the remaining PUs run idle. If a particle leaves its node it is transferred to the processor holding the next sub-domain. Hence communication is only necessary between neighbouring domains, which reduces the number of sends to a constant (number of neighbours $NB = 8$ in 2 dimensions, $NB = 26$ in 3 dimensions). The particle transfer itself is done by a two step approach. In the first step the sending PU tells its neighbours how many particles will arrive in their sub-domain. This step is necessary for the receiving PU to allocate memory and adds up to $P \cdot NB$ sends. During the second step particles are sent to the receiving processors. In the worst case scenario all N particles on one node are transferred at the same time. In an optimal state in which all particles are evenly distributed, but are all changing domains, transfer of $(N/P) \cdot P$ particles is required. Hence communication costs sum up to

$$NB \cdot \mathcal{O}(P) + \mathcal{O}(N) = \mathcal{O}(P+N).$$

Table 3.2 summarises the costs for computation and communication. Both strategies depend heavily on the underlying domain decomposition and the distribution of the particles in the position space Ω. In a worst case scenario the *load ideal strategy* is superior to the *communication ideal strategy* with respect to the computational costs, whereas the *communication ideal strategy* outruns the *load ideal strategy* with respect to communication costs. Assuming that particles are evenly distributed over the PUs the computational cost of strategy two becomes $\mathcal{O}(N/P)$ and is hence equal to those of strategy one resulting in the *communication ideal strategy* to be more efficient.

3.2.3 *Implementation of the* Communication Optimal Strategy

The *communication optimal strategy* is implemented in the function `SuperParticleSystem3D::updateParticleDistribution()` already mentioned above. The function has to be called after each update of the particle positions, in order to check if all particles remained in their current cuboid, as otherwise segmentation faults may occur during the computation of particle forces. The transfer is implemented using non-blocking operations of the MPI library.

```
template<typename T, template<typename U> class PARTICLETYPE>
void SuperParticleSystem3D<T, PARTICLETYPE>::updateParticleDistribution
    ()
{
  /* Find particles on wrong cuboid, store in relocate and delete */
  //maps particles to their new rank
```

```
_relocate.clear();
for (unsigned int pS = 0; pS < _pSystems.size(); ++pS) {
  auto par = _pSystems[pS]->_particles.begin();
  while (par != _pSystems[pS]->_particles.end()) {
    //Check if particle is still in his cuboid
    if (checkCuboid(*par, 0)) {
      par++
    }
    //If not --> find new cuboid
    else {
      findCuboid(*par, 0);
      _relocate.insert(
        std::make_pair(this->_loadBalancer.rank(par->getCuboid()), (*
            par)));
      par = _pSystems[pS]->_particles.erase(par);
    }
  }
}
```

The function begins with two nested loops. The outer loop is over all local `ParticleSystem3D`s, the inner loop over the `Particle3D`s of the current `ParticleSystem3D`. Each particle is checked if it remained in its cuboid during the last update by the function `checkCuboid(*par, 0)`. The first parameter of `checkCuboid(*par, 0)` is the particle to be tested and the second parameter is an optional spatial extension of the cuboid. If the function returns `true` the counter is incremented and the next particle is tested. If the function returns `false` both the particle and the rank of its new cuboid are copied to the `std::multimap<int, PARTICLETYPE<T> > _relocate` for future treatment and removed from the `std::deque<PARTICLETYPE<T> > _particles` of particles.

```
/* Communicate number of Particles per cuboid*/
singleton::MpiNonBlockingHelper mpiNbHelper;

/* Serialise particles */
_send_buffer.clear();
T buffer[PARTICLETYPE<T>::serialPartSize];
for (auto rN : _relocate) {
  rN.second.serialize(buffer);
  _send_buffer[rN.first].insert(_send_buffer[rN.first].end(), buffer,
       buffer+PARTICLETYPE<T>::serialPartSize);
}
```

The function continues by instantiating the class `singleton::MpiNonBlockingHelper`, which handles memory for `MPI_Request` and `MPI_Status` messages. Then the particles buffered in `_relocate` are serialised, meaning their data is written consecutively in memory and stored in a buffer `std::map<int, std::vector<double> > _send_buffer` in preparation for the transfer.

```
/*Send Particles */
int noSends = 0;
for (auto rN : _rankNeighbours) {
  if (_send_buffer[rN].size() > 0) {
    ++noSends;
  }
}
mpiNbHelper.allocate(noSends);
for (auto rN : _rankNeighbours) {
  if (_send_buffer[rN].size() > 0) {
    singleton::mpi().iSend<double>(&_send_buffer[rN][0], _relocate.
        count(rN)*PARTICLETYPE<T>::serialPartSize, rN, &mpiNbHelper.
        get_mpiRequest()[k++], 1);
  }
}
singleton::mpi().barrier();
```

To find the number of send operations a loop over the ranks of neighbouring cuboids is carried out, increasing the variable `count` each time data for a specific rank is available. Then the appropriate number of `MPI_Requests` is allocated. Finally the data is sent to the respective PUs via a non-blocking `MPI_Isend()` and all PUs wait until the send process is finished on each PU.

```
/*Receive and add particles*/
int flag = 0;
MPI_Iprobe(MPI_ANY_SOURCE, 1, MPI_COMM_WORLD, &flag,
    MPI_STATUS_IGNORE);
if (flag) {
  for (auto rN : _rankNeighbours) {
    MPI_Status status;
    int flag = 0;
    MPI_Iprobe(rN, 1, MPI_COMM_WORLD, &flag, &status);
    if (flag) {
      int amount = 0;
      MPI_Get_count(&status, MPI_DOUBLE, &number_amount);
      T recv_buffer[amount];
      singleton::mpi().receive(recv_buffer, amount, rN, 1);
      for (int iPar=0; iPar<amount; iPar+=PARTICLETYPE<T>::
          serialPartSize) {
        PARTICLETYPE<T> p;
        p.unserialize(&recv_buffer[iPar]);
        if (singleton::mpi().getRank() == this->_loadBalancer.rank(p.
            getCuboid())) {
          _pSystems[this->_loadBalancer.loc(p.getCuboid())]->
              addParticle(p);
        }
      }
    }
  }
}
if (noSends > 0) {
  singleton::mpi().waitAll(mpiNbHelper);
}
}
```

On the receiving side the non-blocking routine `MPI_Iprobe()` checks whether an incoming transmissions is available. The constant `MPI_ANY_SOURCE` indicates that messages from all ranks are accepted. If a message is awaiting reception the flag `flag` is set to a non-zero value and the following switch

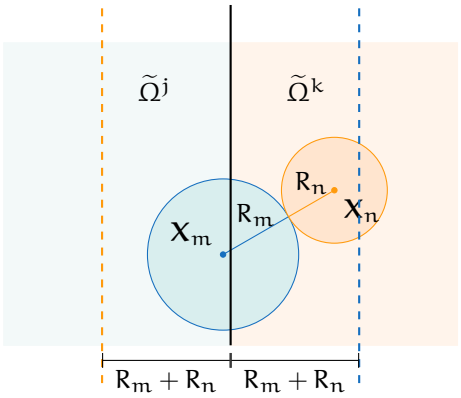


Figure 3.6: Overlap of the particle domains. Particles within a distance to of the sum of the two largest radii to a neighbour cuboid have to be transferred to this specific neighbour cuboid.

will be true. This query is not necessary, but the following loop can be entirely skipped if no particles are transferred, which is expected to be the case most of the time.

The subsequent loop tests for each single neighbouring rank if a message awaits reception. If `true`, the number of send `MPI_Doubles` is read from the `status` variable via an `MPI_Get_count()`. The appropriate memory is allocated and the message is received by wrapped call to `MPI_Recv()`, and written consecutively. Then new `Particle3D`s are instantiated, initialised with the received data and assigned to the respective `ParticleSystem3D` on the updated PU. Finally, a call to `MPI_Waitall()` makes sure that all `MPI_Isend()`s have been processed by the recipients.

If particle collisions are considered, it may happen that particles P_m with centre $\underline{X}_m \in \widetilde{\Omega}^j$ collide with particle P_n with centre $\underline{X}_n \in \widetilde{\Omega}^k$ in a different cuboid, as illustrated in Figure 3.6. Therefore P_n has to be known on $\underline{X}_m \in \widetilde{\Omega}^j$ and so-called shadow particles are introduced. Shadow particles are static particles, whose positions and velocities are not explicitly computed during the update step. Particle collision across cuboid boundaries can only occur if the distance $d = \|\underline{X}_n - \underline{X}_m\|_2$ between the participating particles is less then the sum of the two largest radii of all particles in the system. Hence the width of the particle overlap has to be at least the sum of the two largest particle radii and all particles within this overlap have to be transferred to the neighbour cuboid after each update of the particle position by an additional communication step similar to the one introduced above.

3.3 APPLICATION: LUNG BIFURCATION (EULER–LAGRANGE)

We now apply the introduced particle model to the simulation of airborne particles in the human respiratory system. This section has been published previously in [82] and has been altered to fit the notation in this thesis.

Airborne particulates pose a serious health risk. They may cause respiratory diseases such as lung cancer or asthma. On the other hand, purposefully used e.g. in nasal or asthma sprays they can help to treat such diseases. Investigating the deposition of particulates in the human respiratory system is therefore of great interest, however, *in vivo* examinations are risky, expensive and sometimes not feasible. Instead, simulations are feasible and

provide detailed information on deposition. Yet, simulations of dilute particulate flows remain a challenging task in the research field of Computational Fluid Dynamics (CFD). Especially transient flows through complex geometries such as the nasal cavity demand for computational power that can only be satisfied by massive parallel systems. The EL ansatz allows tracking of individual particles and is widely used to simulate such particulate flows. The LBM is an explicit algorithm to compute fluid flows, that allows to embed a numerical integration scheme for particle tracking. One of LBM's greatest advantages compared to classical CFD methods is an efficient parallelisation by domain decomposition [84, 83, 58]. We have theoretically expanded the domain decomposition of the carrier phase to the particle in Section 3.2.2 by introducing two parallelisation strategies of the particle tracking algorithm. The strategy promising higher performance has been implemented and explained in detail in Section 3.2.3. In the following we will investigate its parallel performance by simulating time-dependent particulate flows in the human respiratory system.

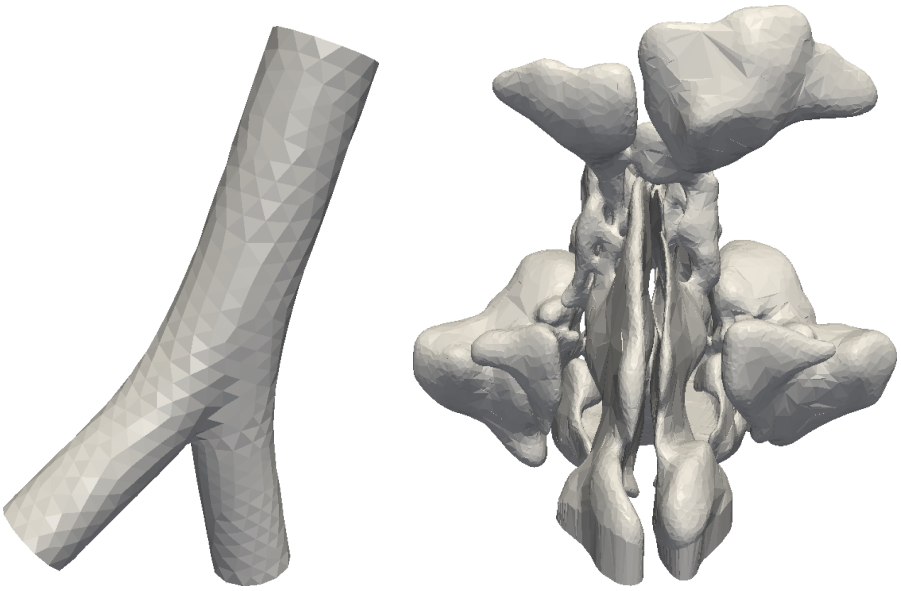
Flow characteristics in the human airways have been studied before, both experimentally and numerically. In a general manner, Kleinstreuer and Zhang [105] provide a detailed overview on publications considering state-of-the-art models, experimental observations and computer simulations for all parts of the respiratory system. Vasconcelos et al. [179] simulated impact distributions of particles in a model of the tracheobronchial tree and stated a simple connection between the Stokes number and the escape rate.

Studies concerning particle deposition in the human nasal cavity have been performed in the past, experimentally on nasal airway replicas [34, 103] as well as *in vivo* measurement on humans [33]. Finck, Hänel and Wlokas [59] simulated particulate flows through the model of a nasal cavity using LBM and Lagrangian particle tracking. Usually in numerical studies particles are injected only once at a certain time, e. g. to model particle drug delivery with a nasal spray [31, 95, 96]. In most of the studies, a constant inhalation flow rate with constant inlet velocity at the nostrils is used, while simulations of unsteady inhalation are rare in literature. To our knowledge only Se, Inthavong and Tu [159] simulated particle deposition using an unsteady flow. When simulating flow in nasal cavities most authors omit the paranasal sinuses. As far as known to the authors, only Tu, Inthavong and Ahmadi [178] investigated deposition of nanoparticles for laminar flow in a geometry considering sinuses with focus on the diffusion process.

The objective of this section is to use the provided parallel algorithm to simulate unsteady particulate flows in the human respiratory system. It is organised as follows: In Section 3.3.1 the mathematical model and numerical method for particulate flows are introduced. Sections 3.3.2–3.3.5 present numeric results of simulations in an idealised bifurcation of the tracheobronchial system. The simulation is validated and speedup tests are performed. The implementation is then used to simulate unsteady particle laden flows in the complex geometry of a patient specific nasal cavity including paranasal sinuses.

3.3.1 *Methods*

In this section of the thesis we employ the EL approach. Hence we choose an Eulerian representation for the fluid phase and each trajectory of the discrete particle phase is tracked individually in a Lagrangian way. The simulations



(a) (b)

Figure 3.7: *Left*: Schematic geometry of one bifurcation of the human tracheobronchial system. *Right*: Geometry of nasal cavity used in Section 3.4.

will be repeated using an EE approach in Section 3.5 and compared to the results obtained here.

In finding particle deposition patterns in the human respiratory system the continuous phase is air. We choose to approximate it as an incompressible Newtonian fluid, which is the general approach for flows with Mach number less than 0.3. Therefore the incompressible NSE as introduced in Equation (3.1) is applied and solved by an by a D3Q19 Bhatnagar–Gross–Krook (BGK)-LBM as introduced in Section 2.3..

Each particle of the disperse phase is tracked according to Newton's second law of motion as introduced in Equations (3.2a) and (3.2b). We assume that the particles are only affected by Stokes drag (3.7). Other forces, such as Basset force or virtual mass, are extremely small in the considered setup and can be neglected [129]. The particle equations of motion are solved by an bEM as introduced in Section 3.1.2.

For the fluid phase, pressure and velocity boundary conditions are realised as proposed in Skordos [164]. A no-slip bounceback condition is set for boundaries representing walls. For the particle phase boundaries are approximated by a triangular mesh. If a particle crosses the boundary, i.e. its position is outside of the domain Ω, its velocity is set to zero and the particle is neglected in further computations.

3.3.2 *Convergence*

Figure 3.7a illustrates the simplified model of the first bifurcation of the tracheobronchial tree used in the simulations. At the upper opening a pressure boundary is imposed, while at the lower openings a velocity boundary condition with a Poiseuille profile is used. The Reynolds number is defined as $Re = UD/\nu$, where $U > 0$ is the mean entrance velocity and $D > 0$ the diameter of the trachea. A number of 10^4 equally distributed particles of each considered radius is injected. More information on the inflow Q, Reynolds

	SYMBOL	VALUE
diameter of trachea	D	0.02 m
mean entrance flow	Q	12, 25, 62, 114, 296 ml s^{-1}
Reynolds number	Re	50, 100, 250, 500, 1300
kinematic viscosity	ν	$1.5111 \cdot 10^{-5}$ m^2 s^{-1}
fluid density	ρ^F	1.205 kg m^{-3}
particle density	ρ^P	998.2 kg m^{-3}
particle radius	R	$3 \cdot 10^{-6} \ldots 10^{-3}$ m

Table 3.3: Simulation parameters for flow through a tracheobronchial bifurcation

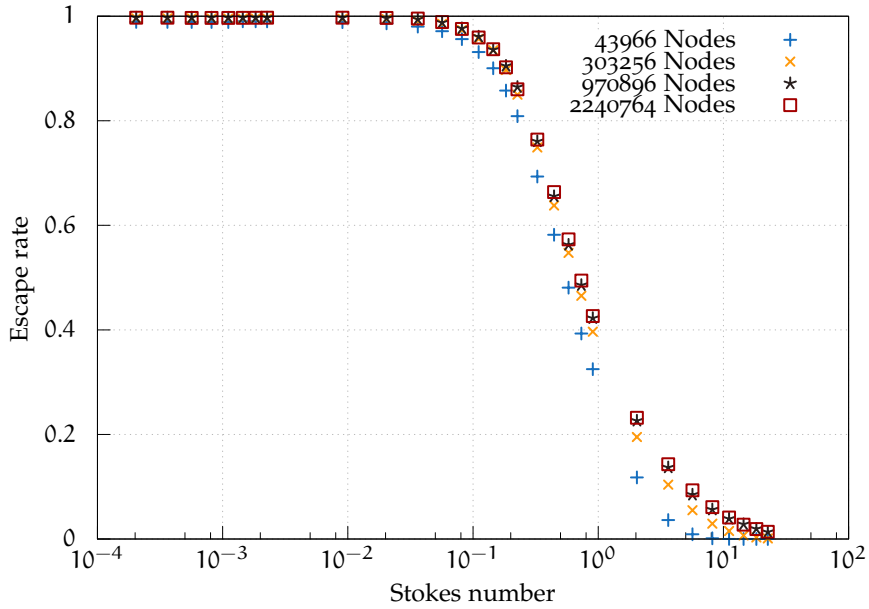


Figure 3.8: Convergence analysis. Results for $Re = 50$.

number Re and the particle radii is listed in Table 3.3. Simulations are terminated after all particles either escaped the geometry or deposited at the wall. The escape rate

$$E = \frac{N^{esc}}{N^{inj}}$$

is defined as the ratio of the number N^{esc} of escaped and number N^{inj} of injected particles.

Figure 3.8 shows escape rates for increasing lattice resolutions and $Re = 50$. The lattices consist of $0.04 \cdot 10^6$, $0.3 \cdot 10^6$, $1 \cdot 10^6$ and $2.2 \cdot 10^6$ fluid cells, respectively, which corresponds to a spacing of $h_1 = 1.04 \cdot 10^{-3}$m, $h_2 = 5.2 \cdot 10^{-4}$m, $h_3 = 3.5 \cdot 10^{-4}$m and $h_4 = 2.6 \cdot 10^{-4}$m. Simulations have been carried out for $30 \cdot 10^5$ particles of radius $3 \cdot 10^{-6}, \ldots, 10^{-4}$m. The relative error Err_i, $i = 1, \ldots, 4$, is defined as

$$Err_i := \frac{\sqrt{\sum_{j=0}^{J} \left(E_{j,i} - E_{j,4}\right)^2}}{\sqrt{\sum_{j=0}^{J} E_{j,4}^2}},$$

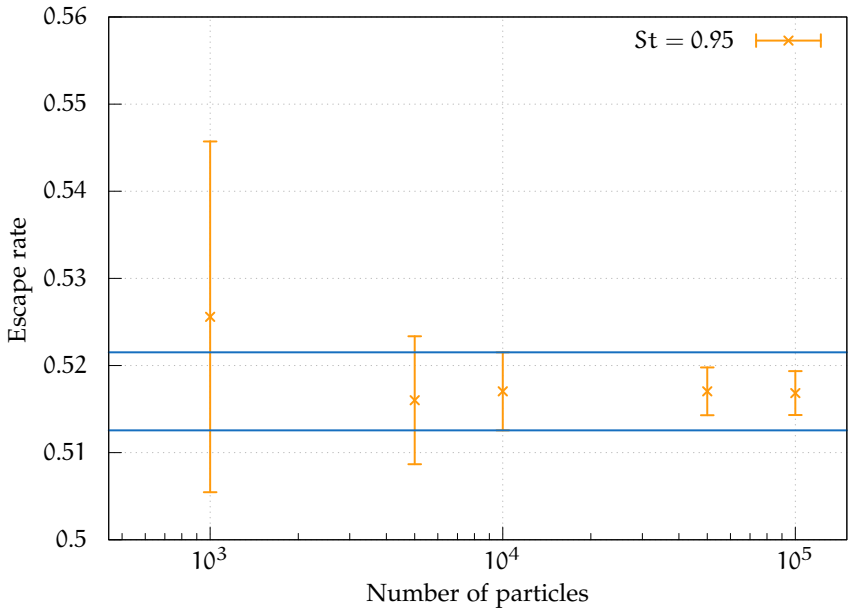


Figure 3.9: Escape rate plotted against the number of particles with $St = 0.95$ and $Re = 50$

where $E_{j,i}$ is the escape rate for particles with radius R_j on a lattice with spacing h_i. Further, the EOC is defined as in Equation (3.20). For the computed escape rates we obtain $EOC_{1,2} = 1.54$ and $EOC_{2,3} = 3.76$.

3.3.3 *Determining Number of Particles*

We now investigate the sensitivity of the escape rate on the number of injected particles. Particles with radius $R = 21\mu m$ are injected, leading to $St = 0.95$, which is in the area of the most sensitive escape rates. Ten simulations have been executed for each $5 \cdot 10^2$, 10^3, $5 \cdot 10^4$, 10^5, $5 \cdot 10^5$ and 10^6 particles. The mean and standard deviation of the resulting escape rates are plotted in Figure 3.9. One can see that for 10^4 particles the standard deviation of the escape rate is less than 0.5%, hence simulations of only a few ten thousand particles will lead to adequate results. This again is an argument for the *communication ideal strategy* as this number of particles can still be computed by a single core.

3.3.4 *Validation*

The parallel implementation of the particle phase is validated by comparing our results to those obtained by Vasconcelos et al. [179] for the same geometry. With the simulation set-up described above, escape rates for particles with radius ranging from $3 \cdot 10^{-6}$ m to 10^{-4} m and Reynolds numbers of $Re = 50, 100, 250, 500, 1300$ are computed. For $Re \geqslant 500$ extensions are added to reduce potential disturbances by the chosen boundary conditions. Figure 3.10 shows the fully developed flow and velocity contours for $Re = 1300$.

In Figure 3.11 the escape rate is plotted against the Stokes number St

$$St = \frac{\tau^P}{\tau^F} = \frac{2\rho^P R^2}{9\nu} \frac{U}{D},$$

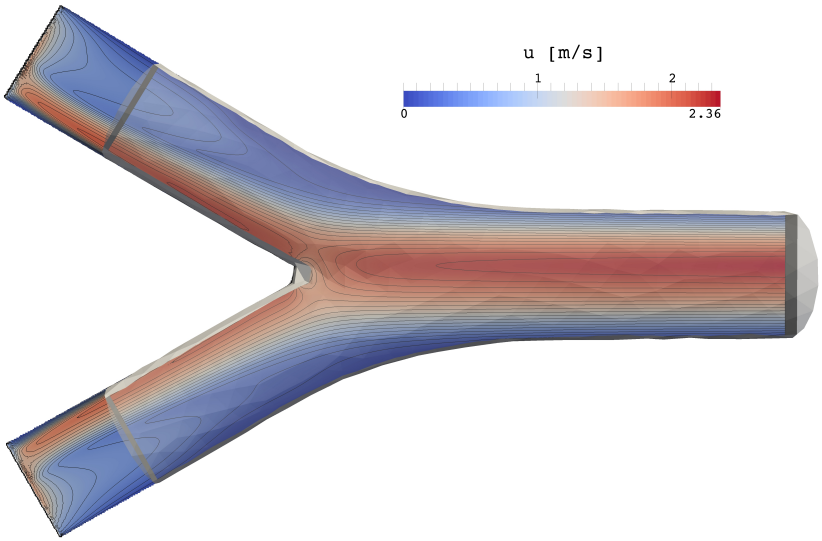


Figure 3.10: Cutting plane through the centre of the geometry, showing the fully developed flow and velocity contours for $Re = 1300$.

which compares *particle momentum response time* τ^P and *characteristic fluid time* τ^F. It indicates that for $St \ll 1$ the particle velocity adapts quickly to the fluid velocity, while for $St \gg 1$ the values of $\underline{U}$ are hardly affected by $\underline{u}^F$. One can see that the curves for low Re almost collapse, but there is a tendency towards lower escape rates with increasing Reynolds number. This tendency is also obtained by Vasconcelos et al. [179], as they write that "a slight departure from the universal collapse can be observed for Reynolds number larger than a few hundreds" [179], although their deviations start at larger Stokes numbers ($St \geqslant 0.8$). This difference might be caused by different boundary conditions, as Vasconcelos et al. [179] used a velocity boundary at the proximal end and pressure boundaries at the lateral end.

3.3.5 *Parallel Performance*

We will now investigate the parallel performance for the *communication optimal parallel strategy*. The parallel speedup is defined by

$$S_P = \frac{T_1}{T_P},$$

where T_P is the execution time on a system with P PUs. For the speedup computation, 71 of initially 128 subdomains $\widetilde{\Omega}_h^l$ are neglected. In order to obtain a result independent of the fluid computations the time necessary for computing $150 \cdot 10^3$ particle trajectories and approximately $2.24 \cdot 10^6$ fluid cells is measured separately. The performance runs are executed exclusively on a cluster system consisting of 480 nodes, each equipped with two Intel® Xeon® CPUs E5-2670 v2 @ 2.60GHz, deactivated Hyper-Threading and 64 GB memory. The nodes are connected by an InfiniBand QDR network. The code is compiled using openmpi v1.8.4 and gcc v4.9.2. This combination results in super-linear speedup for the particle phase for $P < 16$, which is probably owed to cache effects. Table 3.4 summarises the result, showing the wall time per time step averaged over 56 223 time steps and the obtained speedup. As already mentioned the parallel algorithm is highly dependent on the domain decomposition as well as the particle distribution.

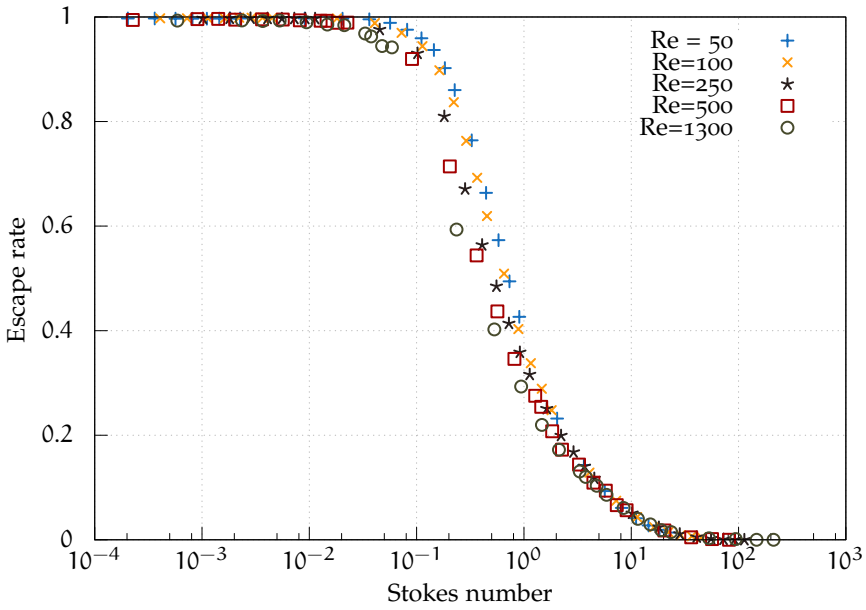


Figure 3.11: Escape rates for increasing Re.

Therefore the presented speedup results can only be seen as an indicator for the capabilities of the method. Ideal scaling can be achieved by an uniform distribution of particles in the computational domain.

3.4 APPLICATION: NASAL CAVITY

This section has been published previously in [82] and has been altered to fit the notation in this thesis.

The underlying geometry of the inner nose of a Central European male with a peripheral obstructive ventilation disorder was obtained from Computer Tomography (CT) scans. The male patient is 46 years old, 1.98 m tall and weighs 105 kg. The CT data is segmented to a surface grid and voxelised to a uniform lattice. Details for this procedure can be found in [110].

P	PARTICLES		FLUID		COMBINED	
	T [s]	S	T [s]	S	T [s]	S
1	1.288	1.00	0.559	1.00	1.847	1.00
2	0.488	2.64	0.279	2.00	0.767	2.41
4	0.219	5.88	0.147	3.80	0.366	5.05
8	0.132	9.78	0.083	6.75	0.214	8.61
16	0.082	15.64	0.057	9.87	0.139	13.29
32	0.040	31.88	0.030	18.93	0.070	26.42

Table 3.4: Averaged wall time per time step in seconds and speedup results for simulations of particle flow in one bifurcation, separated for fluid and particle computations. The speedup result for the particle phase shows a superlinear behaviour.

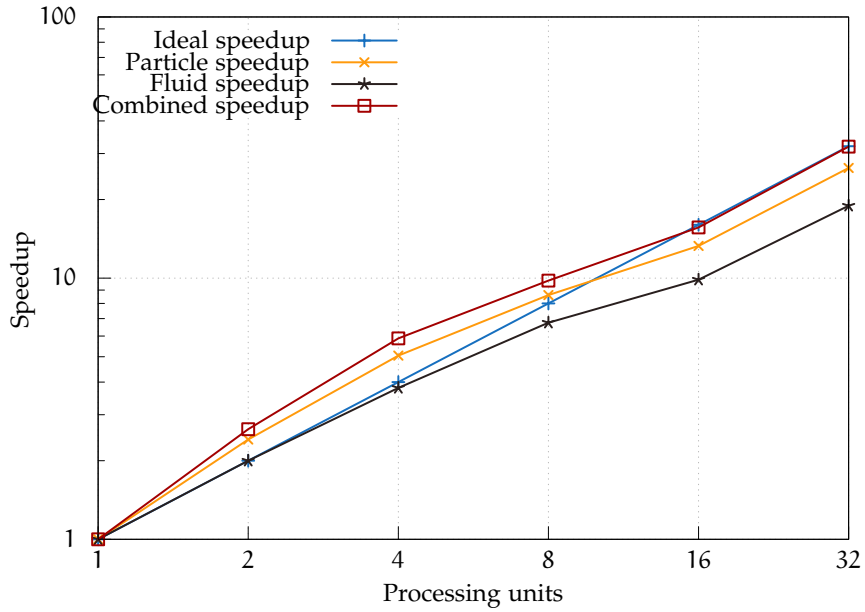


Figure 3.12: The speedup results for the particle phase shows super-linear behaviour, which may be caused by cache effects.

The lattice spacing is set to $h = 0.33\,\text{mm}$, leading to 1.1 million fluid and 0.3 million boundary voxels.

A no-slip condition, realised by a bounceback condition, is applied at the walls. A pressure boundary condition is applied at the nostrils. An inflow velocity is imposed to the nasopharynx according to Poiseuille's law for elliptic pipes. The respiratory cycle is approximated by a sinusoidal function with a period of 5 s. The maximum inflow is set to $104\,\text{ml/s}$, leading to $Re = 1925$. Particles of 10 different sizes between $1\,\mu\text{m}$ and $35\,\mu\text{m}$ are injected once at the beginning of the inhalation as well as continuously during an entire inspiratory cycle, which has — to the knowledge of the authors — not been simulated before. For single injection 120.800 particles have been used, while for continuous injection the total amount of injected particles summed up to 604.000. For non-steady inhalation continuous injection reflects everyday breathing more realistically. It also leads to a more uniform spatial distribution of the particles which results in a better load balancing for the chosen parallelisation technique. The underlying fluid flow simulation has been validated in Krause et al. [109] by comparison of the simulated pressure drop to the results of a rhinomanometric examination of the patient and literature data. Figure 3.13 shows the simulated flow field from different points of view. The fluid velocity is visualised as coloured spheres. The upper right picture shows a closeup view of the left inferior meatus with an apparent increase in velocity caused by a narrow.

The simulation results reveal that attributed to a backflow a significant amount of particles escape through the left nostril (up to 14% for $16\,\mu\text{m}$ particles and single injection, illustrated in Figure 3.15). Therefore depostion rates are computed in two ways, once by neglecting the number of particles that escaped through the nostrils N^{nostril} according to

$$D^* = \frac{N^{\text{dep}}}{N^{\text{dep}} + N^{\text{trachea}}},$$

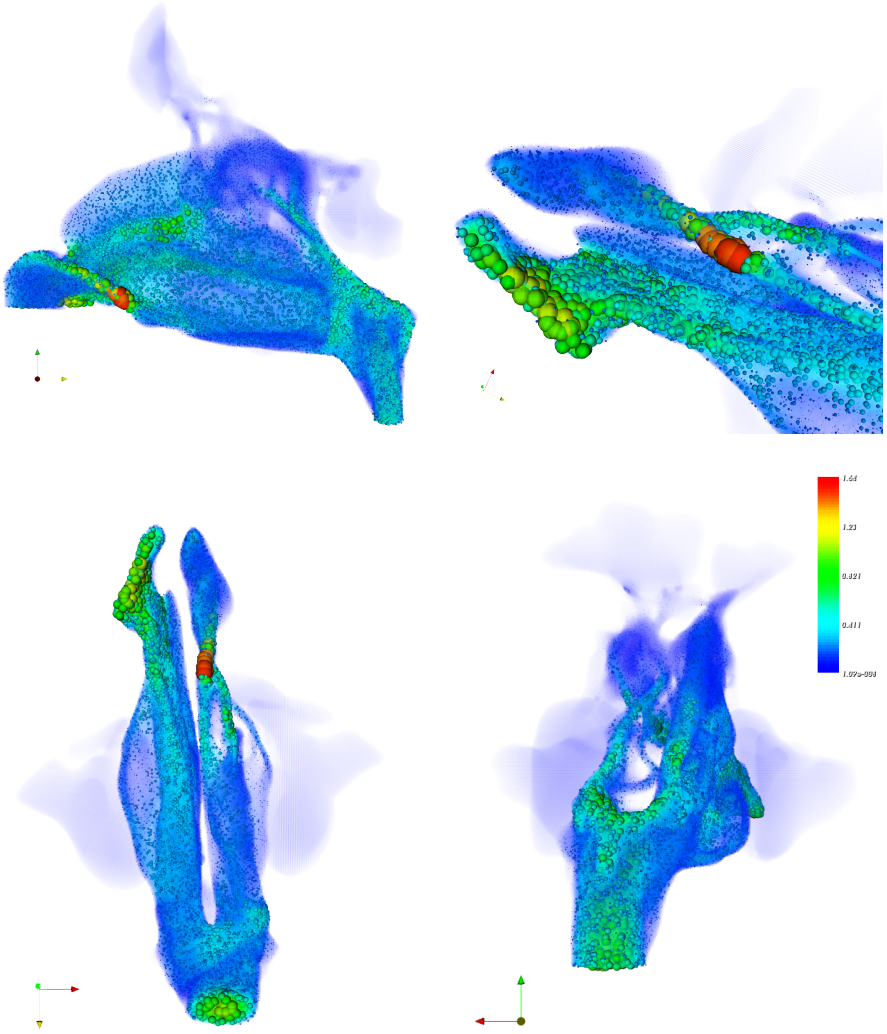


Figure 3.13: The simulated flow field for an expiration flow rate of Q = 100 ml/s from different point of views, whereby the right upper picture displays a closer look to a significant narrow at the left inferior meatus. The magnitude of the velocity is visualised as coloured spheres. Images from [110].

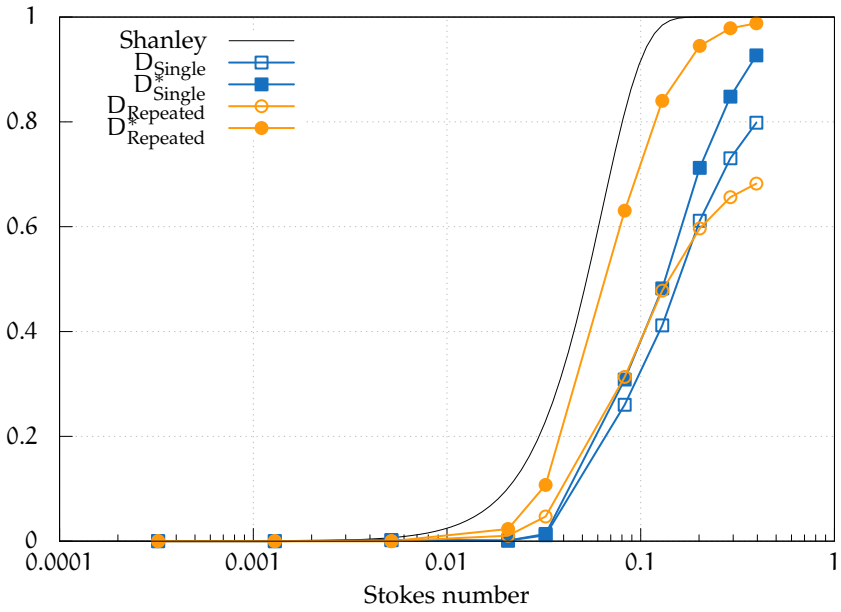


Figure 3.14: Deposition rates D plotted against the Stokes number for single and repeated injection of particles in comparison to the data obtained by Shanley et al. [161]. The adjusted curves D* neglect particle loss through the nostrils.

and once by including it

$$D = \frac{N^{dep}}{N^{dep} + N^{trachea} + N^{nostril}}.$$

The results are plotted against the Stokes number in Figure 3.14. To allow a better comparison to steady inhalation, St is computed from the spatially and temporally averaged inflow velocity. The deposition values are taken after one complete respiratory cycle (inspiration and expiration). A good agreement for D^* is found by comparison to the function

$$E_d(St) = 1 - \exp(-250\,(St)^2)$$

fitted to the simulation data obtained by Shanley et al. [161] for steady inhalation. Differences may be attributed to variability of nasal cavity models, the narrow in the left nostril and the time-dependent inflow. It can be seen that the deposition rate is again highly dependent on the Stokes number. Particles with Stokes numbers less than 0.01 can escape the nose unhindered, while more than 90% of the particles with $St = 1$ deposit. For the adjusted data, deposition for repeated injection is greater than for single injection for the entire range of the Stokes number. For the original data, deposition D is comparable for single and repeated injection and $St < 0.2$. For greater St depostion for single injection increases faster. After a second respiratory cycle without re-injection of particles, the rates vary by less than 2%.

The change of the escape rates over time for particles of radius 10 µm is illustrated in Figure 3.16 for single injection and Figure 3.17 for continuous injection. The blue curve indicates the fluid velocity applied at the boundary, where negative values mean inhalation. At time $t = 0$ s particles are injected, 0.5 s later particle escape through the trachea starts. The escape rate remains constant after 2.4 s at 72 %. Particle loss through the left nostril starts at 0.6 s.

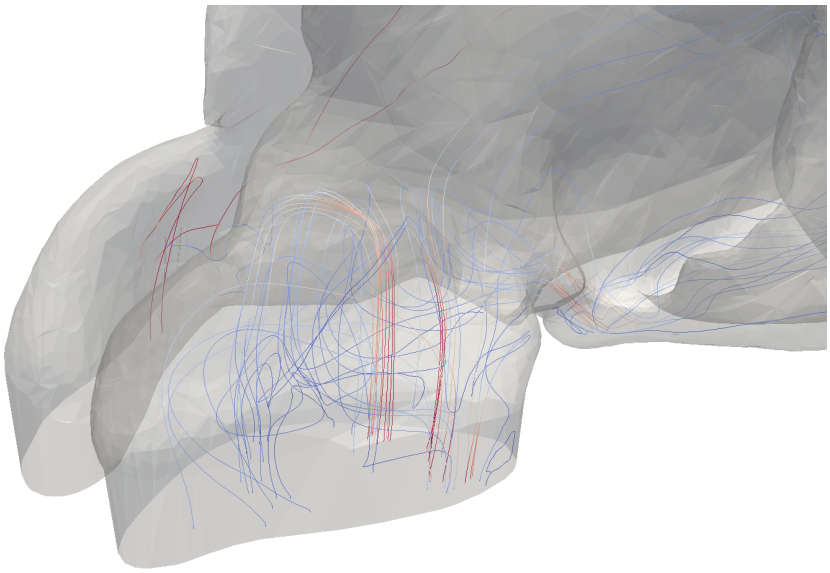


Figure 3.15: Closeup of the left nostril. Streamlines indicate a backflow, forcing particles to turn and escape through the left nostril.

For single injection after 1s no more particles escape through the nostrils. With beginning exhalation, there is no increase in the escape rates of the nostrils. For repeated injection, particles released shortly before the exhalation change direction with beginning exhalation at 2.5 s and exit the cavity through both nostrils. For single injection, 72 % escape through the trachea which is considerably more than for continuous injection (36 %).

In Figure 3.18 and Figure 3.19 deposition patterns for 10 µm particles for single and continuous injection are illustrated. Red colour indicates deposited particles, while blue coloured particles are still active. The upper two images show the distribution after inhalation and the lower images after an entire respiratory cycle. In both cases most particles deposit in the anterior part of the nasal cavity, minor deposition occurs at the transition to the nasopharynx. This is reflected in the deposition patterns, as significantly more particles deposit on the right side. Active particles are located in both the frontal and the maxillary sinuses and remain there even after the completed cycle.

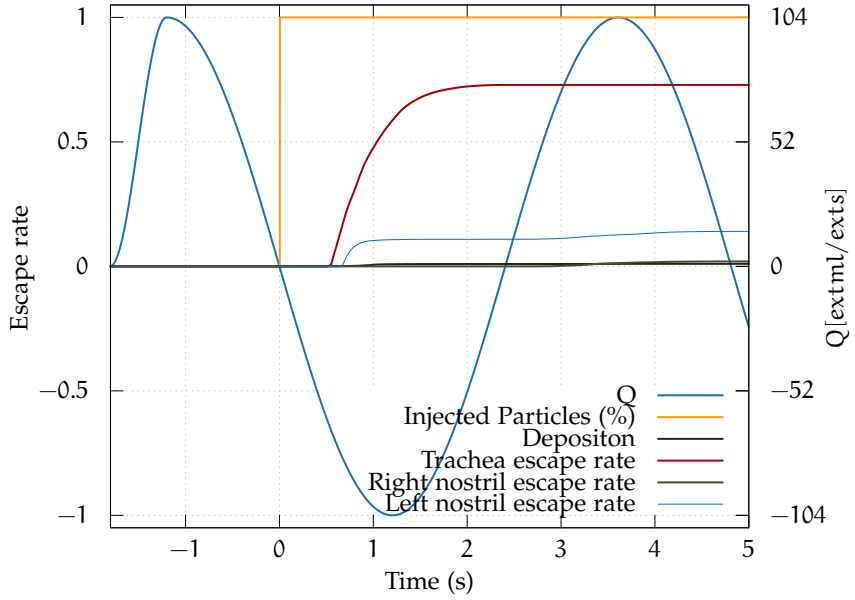


Figure 3.16: Simulation of particulate flow through a nasal cavity. Particles of radius 10 µm are injected once at 0 s. The red line indicates the time-dependent flow of maximal 104 ml/s at the nasopharynx. Positive values denote an outflow through the nasopharynx.

3.4.1 *Summary*

When simulating time-dependent particulate flows, one faces the problem, that the domain decomposition used for fluid simulation is not optimal for parallel computation of particle trajectories. Therefore, we compared the complexity of a *load optimal* and a *communication optimal* strategy for parallelisation of particle trajectory computation. Both strategies are based on the given fixed domain decomposition from the underlying fluid computation with LBM. Under the assumption of homogeneous distribution of particles the *communication optimal* strategy is found to be more efficient and is therefore implemented and validated. The EOC of the implemented method is computed for varying lattice resolutions and particle numbers. Speedup tests are conducted and lead to super-linear scaling, that may be caused by caching effects. The method is used to investigate particle deposition in the human nasal cavity for non-steady airflow for one respiratory cycle. Particles of radii between $R = 1\,\mu m$ and $R = 35\,\mu m$ are injected both, once at the beginning of inspiration as well as continuously during inspiration. For $St < 0.01$ most particles follow the air stream and exit the cavity through the trachea, while for $St > 1$ most particles deposit. If the particle loss through the nostrils is neglected, repeated injection shows higher deposition for all simulated Stokes numbers. Particles of radius $R = 10\,\mu m$ are investigated in more detail. A significant amount of particles gets lost through the nostrils in consequence of a backflow. It is found that for continuous injection the particle escape rates through the nostrils increase during expiration. In both cases most particles deposit in the anterior region and remain in the sinuses after one completed respiratory cycle. Therefore, transient simulations are necessary, when simulating particle flow in nasal cavities.

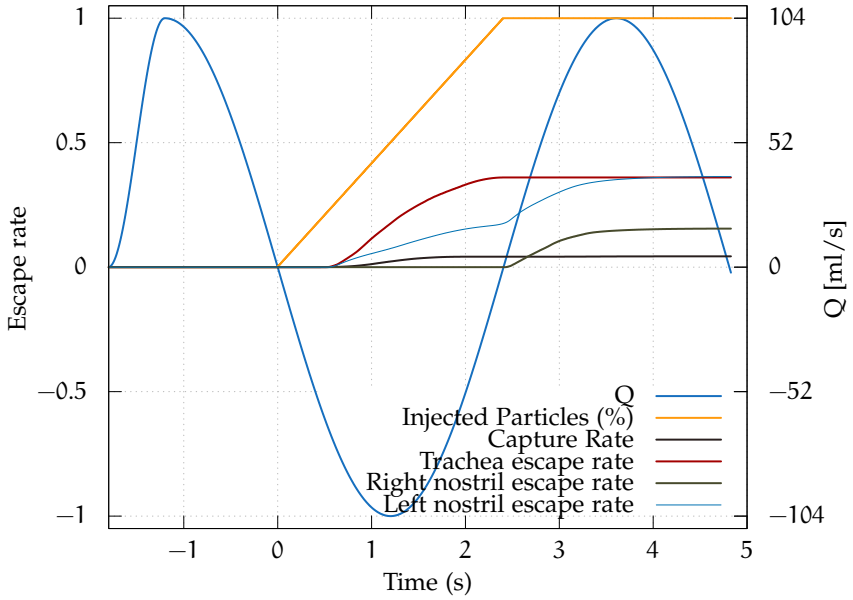


Figure 3.17: Simulation of particulate flow through a nasal cavity. Particles of radius 10 µm are injected continuously during the inspiratory cycle. The red line indicates the time-dependent flow of maximal 104 ml/s at the nasopharynx. Positive values denote an outflow through the nasopharynx.

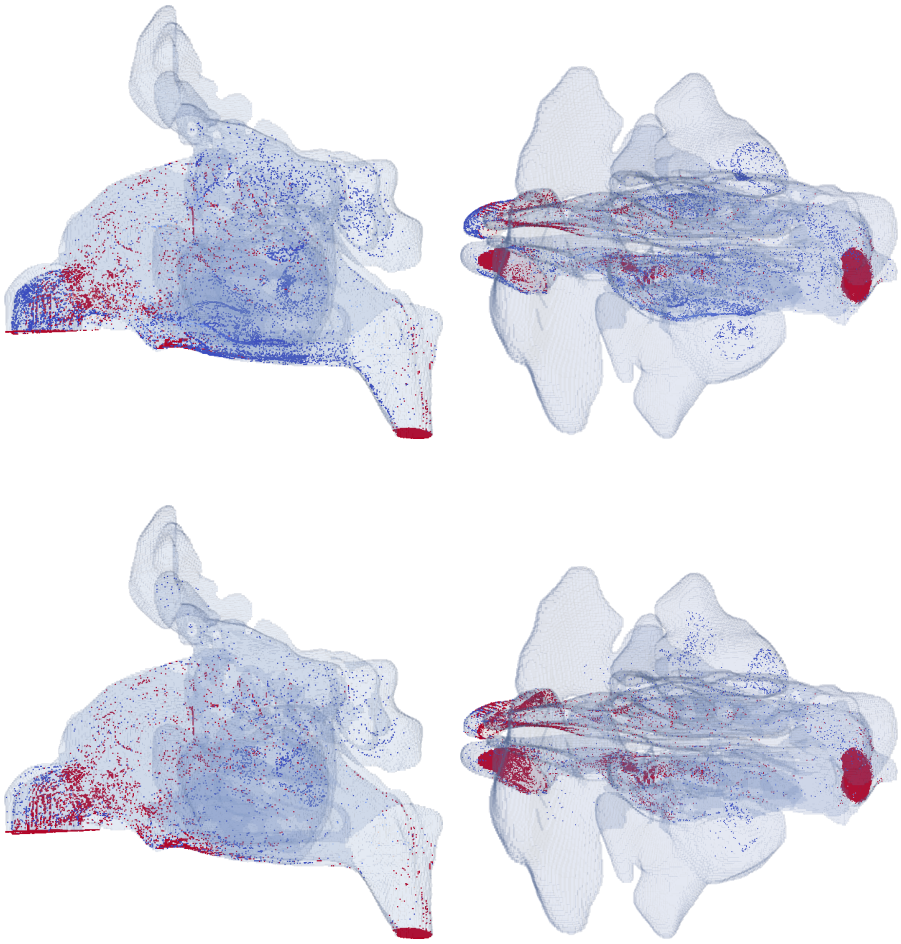


Figure 3.18: Particle distribution for non-recurring injection of 10 µm particles in a human nasal cavity (nostrils on the left side). The upper pictures show the particle deposition after the inhalation, the lower ones after the completed breathing cycle. Red particles are deposited, while blue particles are still active.

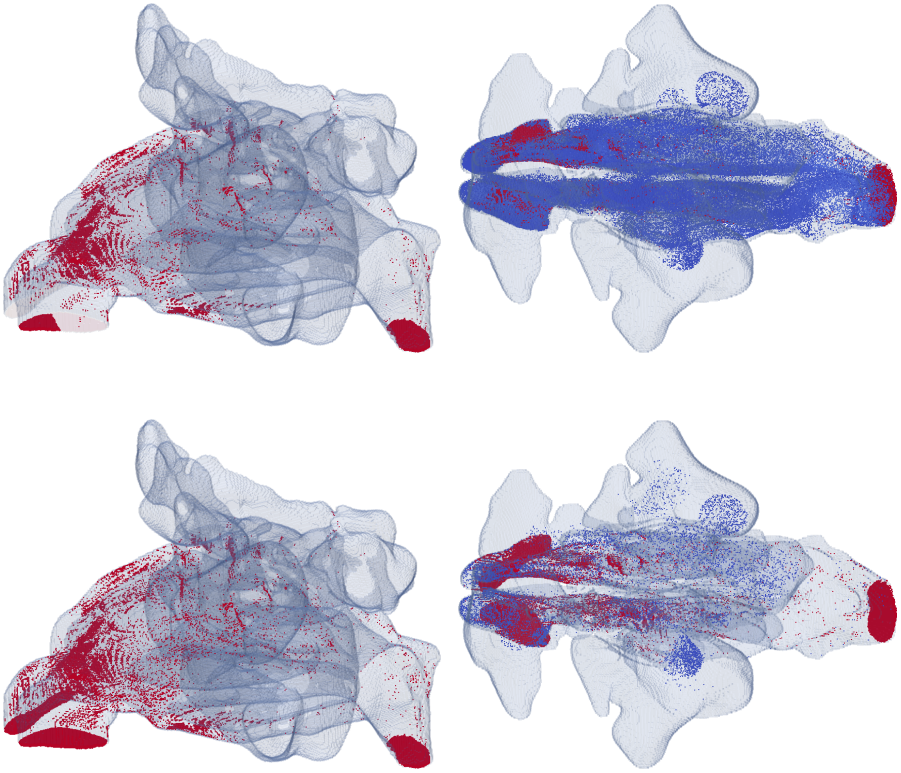


Figure 3.19: Particle distribution for continuously injected 10 µm particles in a human nasal cavity (nostrils on the left side). The upper pictures show the particle deposition after the inhalation, the lower ones after the completed breathing cycle. Red particles are deposited, while blue particles are still active.

3.5 EULER–EULER

After focusing on the EL approach in the first half of this chapter we will now concentrate on the EE method. In the EE approach both components, the particles and the fluid, are assumed to be continuous and are modelled in the Eulerian form. The remainder of this chapter has been published originally in [176] and was slightly revised.

For the design of new filtering systems or to enable predictions on the impact of the exposure of the human respiratory system to air pollution, as well as in many different engineering disciplines like chemical or process engineering the simulation of particles in a fluid flow is of great importance [120, 152]. Here, spherical particles in a fluid with radii between 1 µm and 800 µm are considered. This is of interest in many applications, e. g. in filtration, drug delivery by spray medication with a nebuliser or the simulation of particle-laden flows in a bifurcation of the upper human bronchial tree, i.e. the trachea. The deposition is studied regarding the Stokes number, defined as the ratio of particle relaxation time to a characteristic time of the flow. Examined are particles with values between $St = 0.01$ and $St = 15$ for the considered setup. This has also been investigated by Henn et al. [82] and Vasconcelos et al. [179]. In the described regime the particle component is characterised as dilute and inertial. The low concentration results in a negligible diffusion coefficient, that is also inversely proportional to the product of particle radius and solvent viscosity. In this range of parameters however, the drag force has to be taken into account, since also Stokes numbers greater than 0.1 are considered. This implies that inertia has to be included in the model.

For the simulation of systems with this parameter setup usually an Euler–Lagrange (EL) approach is chosen [82, 179]. In this approach the particles are seen in a Lagrangian frame of reference, with the observer moving along. However, for a higher concentration of considered objects, which is required if a precise deposition pattern is desired, a continuous model is more efficient, since the parallelisation of a Lagrangian particle component requires more effort, as shown in Section 3.2.2. The computational costs of an EE scheme scale solely with the resolution of the computational domain and not with the amount of particles. For this approach the point of view is fixed in space and the particle concentration is observed as a continuous quantity. This frame of reference has been used by Lee and Liu [120] and Rao and Faghri [152] for the simulation of particle-laden flows in filter geometries. It is usually applied to flows of nano-particles that require the consideration of diffusion, which is included in this model. The Stokes number on the other hand is small and inertial effects can be neglected. To investigate the deposition for a broad range of particle radii in the upper bronchial tree and the pharynx, Zhang et al. [203] and Kleinstreuer and Zhang [106, 105] used the Eulerian approach in addition to the Lagrangian one to also consider nano-particles. However, in this framework the formulation of boundary conditions is less flexible to model particle deposition. The reason is that in an Eulerian approach with continuous concentration the formulation of an outflow condition that prevents a flow back into the computational domain is not as straight forward as for Lagrange particles.

The main aim of this section is to extend the EE approach to be applicable to micro-particles and to take advantage of the reduced computational cost. The equations for the fluid as well as the particle component are solved using the LBM [2, 30] and the proposed scheme is implemented in `OpenLB`.

Within this framework domain decomposition methods can be applied to both components, which leads to massive improvements in computational performance [58, 84]. Since LBM also allows a formulation of boundary conditions on a mesoscopic scale, the deposition of particle densities is modelled by a newly proposed capture boundary. It is designed to only allow an unidirectional flow of particles. On the macroscopic scale, this requires a complicated implementation. This gives the possibility to model sticky walls that capture particles during the simulation and also keep track of the amount of already deposited particles.

Extensions to the Eulerian approach that account for inertia and hence are applicable to the regime of micro-particles have also been proposed by Sokolichin et al. [167] and Kaufmann et al. [102]. As found by the latter, numerical instabilities can occur in the considered simulation setup, since the problem is convection-dominated. The application of a larger diffusion coefficient or numerical stabilisation schemes that add artificial diffusion, as discussed by John and Schmeyer [98] and Augustin et al. [7], lead to deviations in the solution compared to the one obtained with a Lagrangian approach. In this article a Smagorinsky turbulence model is used for the fluid flow and a similar model is proposed and newly applied to the particle component. Compared to the physical model, additional diffusion is applied by this scheme. To reduce the influence of this effect to a negligible scale on the relevant quantity, i.e. the escape rate, a correction method is introduced. Therefore the proposed scheme also yields viable results in the convection-dominated regime. To the knowledge of the authors this is the first EE approach to the simulation of deposition of inert particles using LBMs. The proposed extension consists of a one way coupling of a NSE on an ADE by calculating the particle velocity from the flow field. This is done using Newton's second law of motion and Stokes drag force.

This chapter is organised as follow: The mathematical model is described in Section 3.5.1. Then the novel numerical framework for the computation as well as appropriate and also novel boundary conditions are presented in Section 3.5.3 combined with the applied stabilisation techniques and an algorithm for the whole scheme. Finally, in Section 3.6 the introduced Smagorinsky model for the ADE is evaluated and the proposed method is applied to a schematic bifurcation of the trachea into the main bronchi. The results are compared to the ones obtained in Section 3.3.4 and the work of Vasconcelos et al. [179] for validation. The results are found to be in excellent agreement, validating this extension of the EE scheme to inertial particles.

3.5.1 *Mathematical Modelling*

The model consists of three parts, namely the fluid and particle component as well as their coupling. To describe the fluid flow the incompressible NSE (3.1) is used. As initial condition

$$\underline{u}^F(\underline{x}, 0) = \underline{0} \qquad \text{for } \underline{x} \in \Omega$$

is chosen. The boundary conditions are discussed in Section 3.5.4.

The distribution of the particle concentration $c : \Omega \times I \to \mathbb{R}^+$ is computed by solving an ADE

$$\partial_t c + \underline{u}^P \cdot \nabla c - D\Delta c = S \qquad \text{in } \Omega \times I\,, \tag{3.21}$$

where the particle velocity is denoted by $\underline{u}^P : \Omega \times I \to \mathbb{R}^d$ and S represents source and sink terms within the considered domain. $D \in \mathbb{R}^+$ is a constant

diffusion coefficient. Cussler [42] lists experimentally obtained values for D. They reach from $\approx 10^{-5}\,m^2/s$ for diffusion within gases at one athmosphere to $\approx 10^{-50}\,m^2/s$ for diffusion within solids. Initially, the particles are assumed to be uniformly distributed,

$$c(\underline{x}, 0) = 1 \qquad \text{for } \underline{x} \in \Omega\,.$$

Boundary conditions for this equation are also discussed in Section 3.5.4.

Since the developed scheme is used to describe particles with a diameter in the micrometer scale, drag force has to be taken into account. This is done by applying Newton's second law of motion

$$M d_t \underline{u}^P = \underline{F}^{St} \qquad \text{in } \Omega \times I\,, \tag{3.22}$$

with particle radius $R > 0$ and mass $M > 0$ in order to compute the particle velocity $\underline{u}^P$ from the fluid field $\underline{u}^F$. The force used here is the Stokes drag force, given by

$$\underline{F}^{St} = 6\pi\rho^F \nu R(\underline{u}^F - \underline{u}^P)\,. \tag{3.23}$$

At this point other forces of the Maxey–Riley Equation (3.4) may be considered, however they are of minor importance in the chosen range of parameters and will be neglected. Since all quantities are viewed in the Eulerian frame of reference, the material derivative of the velocity has to be considered. The resulting velocity field $\underline{u}^P$ is applied in the ADE, which represents the coupling of the fluid to the particles. Since the particle fraction is assumed to be dilute, a back-coupling can be neglected.

3.5.2 *Fluid Component*

The carrier phase modelled by the NSE (3.1) is solved on the discrete lattice Ω_h and discrete time interval I_h by a D3Q19 BGK-LBM as introduced in Section 2.3.

3.5.3 *Particle Component*

For the simulation of the particle component two steps are required. The velocity has to be modified to account for inertia of the considered objects and is applied in an ADE that is solved afterwards.

PARTICLE DISTRIBUTION

In the LBM framework it is also possible to solve the ADE (3.21) by considering the system

$$g_i(\underline{x}_h + \underline{\xi}_i h^2, t + h^2) = \widetilde{g}_i(\underline{x}_h, t)\,, \tag{3.24}$$

$$\widetilde{g}_i(\underline{x}_h, t) = g_i(\underline{x}_h, t) + \frac{1}{\tau^P}\left(g_i(\underline{x}_h, t) - g_i^{eq}(\underline{x}_h, t)\right)\,. \tag{3.25}$$

In contrast to the LBM solving the NSE, the density distribution functions $g_i : \Omega_h \times I_h \to \mathbb{R}^+$ relax towards the equilibrium distribution

$$g_i^{eq}(\underline{x}_h, t) = \omega_i \rho_h^P \left(1 + 3\underline{\xi}_i \underline{u}_h^P(\underline{x}_h, t)\right)\,,$$

that is obtaind by truncating the Taylor expansion of the continuous Equilibrium Distribution Function (EDF) (2.26) after the second addend. The concentration of the particles is then denoted by $\rho_h^P : \Omega_h \times I_h \to \mathbb{R}^+$, cf. Mohamad

[132] and the discrete particle velocity by $\underline{u}_h^P : \Omega_h \times I_h \to \mathbb{R}^d$. Both can be computed as the zeroth and first discrete moments of g_i. The numerical particle concentration $\rho_h^P \propto c$ is proportional to the physical particle concentration c and should not be mistaken for the particle mass density ρ^P. The relaxation parameter τ^P is linked to the diffusivity D by $\tau^P = 3D + 1/2$, instead of the viscosity. To recover the ADE from the lattice Boltzmann formulation, a smaller set of discrete velocities is sufficient, compared to the setup for the NSE. This has been shown e. g. by Huang, Lu and Sukop [92]. However, for this smaller set, e. g. a D2Q5 lattice with weights $\omega_0 = 1/3$ and $\omega_1 = \omega_2 = \omega_3 = \omega_4 = 1/6$, the momentum conservation is no longer guaranteed, whereas the conservation of mass still holds [92, 166].

PARTICLE VELOCITY

By inserting the formulation of Stokes drag (3.23) and taking the material derivative, (3.22) yields

$$\begin{aligned} M\big(\partial_t \underline{u}_h^P(\underline{x}_h, t) + \underline{u}_h^P(\underline{x}_h, t) \cdot \nabla \underline{u}_h^P(\underline{x}_h, t)\big) = \\ = 6\pi R \rho^F \nu \big(\underline{u}_h^F(\underline{x}_h, t) - \underline{u}_h^P(\underline{x}_h, t)\big) \, . \end{aligned} \tag{3.26}$$

The mass of one particle M can be computed from ρ^P and R, since all particles are assumed to be spherical. Considering the Stokes number St, as defined in Equation 3.3, the right hand side of (3.26) can be rearranged. Applying a fEM for temporal discretisation, the particle velocity of a new time step reads

$$\underline{u}^P(\underline{x}_h, t + h^2) = \left(\underline{u}^P + h^2 \left(\frac{1}{St} \frac{U}{L} (\underline{u}^F - \underline{u}^P) - \underline{u}^P \cdot \nabla \underline{u}^P\right)\right)(\underline{x}_h, t) \, . \tag{3.27}$$

The fluid velocity $\underline{u}^F$ is obtained from the fluid component and $\nabla \underline{u}^P$ is computed by a first order upwind scheme.

3.5.4 *Boundary Conditions*

While for the fluid component standard boundary conditions can be applied, the formulation of a new capture boundary is required for the particle component. For the fluid phase, a velocity boundary as proposed by Skordos [164] is used as inflow condition, while the bounceback rule [171] can be applied at the no-slip walls. For a grid node $\underline{x}_h$ on the outflow boundary, the condition

$$f_i(\underline{x}_h, t) = \frac{1}{2}\big(f_i(\underline{x}_h + h^2 \underline{n}, t) + f_i(\underline{x}_h + 2h^2 \underline{n}, t)\big), \quad i \in \{0, \ldots, q-1\}$$

is applied, with $\underline{n}$ being the unit normal vector on the boundary pointing to the interior of the considered domain. By this interpolation, the conditions

$$\partial_{\underline{n}}(\rho_h^F \underline{u}_h^F) = \underline{0} \quad \text{and} \quad \partial_{\underline{n}} \rho_h^F = 0$$

hold approximately, where $\partial_{\underline{n}} = \partial / \partial \underline{n}$ denotes the derivative in direction of the normal vector. At this point the Neumann boundary condition $\partial_{\underline{n}} \underline{u}_h^F = 0$ can be inferred by

$$0 = \partial_{\underline{n}}(\rho_h^F \underline{u}_h^F) = \underline{u}_h^F \partial_{\underline{n}} \rho_h^F + \rho_h^F \partial_{\underline{n}} \underline{u}_h^F = \rho_h^F \partial_{\underline{n}} \underline{u}_h^F \, .$$

This outflow condition is found to have only a limited influence on the interior of the computational domain for laminar flow patterns. A more

detailed discussion on LBM outflow boundary conditions can be found in Junk and Yang [100].

The particles at the inflow can be inserted using a Dirichlet boundary condition, whereas for the outflow the same condition as for the fluid is applied to the g_i of the boundary. To allow for deposition, a capture boundary condition is formulated. For reasons of simplicity, the particles are not assumed to be able to get back into the fluid once they are deposited. In the framework of LBM it is easy to construct such a boundary at a grid node $\underline{x}_h$. It simply requires setting

$$g_i(\underline{x}_h, t) = 0, \quad i \in \{j : \underline{x}_h + \underline{\xi}_j \in \Omega_h\} \tag{3.28}$$

for every i whose respective $\underline{\xi}_i$ is pointing at an interior node of the computational domain. As deposited particles are transported by a layer of mucus in the tracheobronchial system, possible particle cumulation on the boundary is small and can be neglected. The captured amount of particles is computed from density distributions pointing at such a boundary node.

3.5.5 *Stabilisation*

To be able to apply the model to flows with various Reynolds numbers, stabilisation schemes [7, 98] need to be applied. This aims at eliminating numerical errors that occur when leaving the laminar regime, with as little impact on the desired solution as possible. Again standard schemes can be applied for the fluid component, while an appropriate procedure has to be found for the ADE in the convection-dominated case. For the fluid a Smagorinsky type turbulence model is chosen as explained in Section 2.3.2.

The computation of the particle velocity is stabilised by applying a classical first order upwind scheme, described by Courant, Isaacson and Rees [37] for the computation of the velocity gradient in Equation (3.27). When high velocities are considered, the Péclet number $Pe = LU/D$ becomes large. This indicates a convection-dominated problem, which results in large deviations of the distribution functions f_i from the equilibrium state in the framework of LBM. Instabilities in the results of the particle component, arising from this condition, can be smoothened by applying a Smagorinsky model similar to the one for the fluid introduced in Section 2.3.2. Equations (2.49) and (2.50) remain unchanged. However, $\Pi_{\alpha,\beta}$ and $S_{\alpha,\beta}$ can be seen as indicators of critical areas, however they have no physical equivalent in this context. While in Section 2.3.2 the Smagorinsky model adapts τ, which is connected to the fluid viscosity ν, in this case τ^P is changed, which is connected to the diffusivity D. In this sense, on the macroscopic level, artificial diffusion is introduced in regions of large deviation from the equilibrium state. This leads to more interaction between the particles and the walls. Regarding the chosen boundary conditions this leads to a higher deposition rate due to numerical diffusion.

The focus in the following simulations will be on the escape rate as a function of the Stokes number given as

$$E(St) = \frac{N^{esc}(St)}{N^{inj}},$$

given as the ratio of escaped to injected particles. By comparison, the results of the proposed Eulerian scheme show deviations to the Lagrangian approach by Kaufmann et al. [102]. This is caused by the increased diffusion of the Smagorinsky model resulting in particles being trapped at the walls.

Listing 3.2: LBM algorithm including Eulerian particles.

```
for t ∈ I_h {
  for x_h ∈ Ω_h {
    for i = 0, ..., q − 1 {
      f̃_i(x_h, t) = f_i(x_h, t) + 1/τ (f_i(x_h, t) − M_{h,i}(x_h, t))
      f_i(x_h + ξ_i h², t + h²) = f̃_i(x_h, t)
    }
  }
  compute fluid boundary conditions
  compute u^P(x_h, t + h²) from u^P(x_h, t), u^F(x_h, t), St, U, L
  for i = 0, ..., q − 1 {
    g̃_i(x_h, t) = g_i(x_h, t) + 1/τ (g_i(x_h, t) − g_i^eq(x_h, t))
    g_i(x_h + ξ_i h², t + h²) = g̃_i(x_h, t)
  }
  compute particle boundary conditions
}
```

To obtain viable results, a corrected value of the escape rate is introduced by

$$E_{corr}(St) = \frac{E(St)}{E_0} \,. \tag{3.29}$$

The reference value E_0 is computed by directly applying the fluid velocity to the particles, therefore omitting the drag force, so that no deposition should occur since the physical diffusion is assumed to be negligible. However, due to the additional numerical diffusion, it is found that not all particles, but only a number $E_0 < 1$ exit the geometry through the bronchial openings. The percentage of particles leaving through the bronchial openings is taken as reference value E_0. This value has to be determined specifically for each parameter setup.

3.5.6 *Algorithm*

An overview of the required steps of the developed EE model for particle-laden fluid flows with LBM is given in Listing 3.2. For the computation of the particle velocity $\underline{u}^P(\underline{x}, t)$ Equation (3.27) is applied.

3.6 APPLICATION: LUNG BIFURCATION (EULER–EULER)

For the simulations an idealised bifurcation, representing the trachea splitting into the main bronchi, is chosen as computational domain (Figure 3.20). It is designed after Weibel's model "A" [188] with main tube diameter $d_M = 2\,\text{cm}$ and daughter tube diameter $d_D = 2^{-1/3} d_M$. The angle between the branches is $\alpha = 60°$. The inflow fluid velocity is defined by a Poiseuille flow profile and the particles are assumed to be uniformly distributed while entering the geometry through the main tube. For the walls the capture boundary described in Section 3.5.4 is applied. The amount of particles leaving the geometry through the daughter tubes is measured and used for the computation of the escape rate. Additional simulation parameters are given in Table 3.5. As characteristic values the diameter of the main tube L and

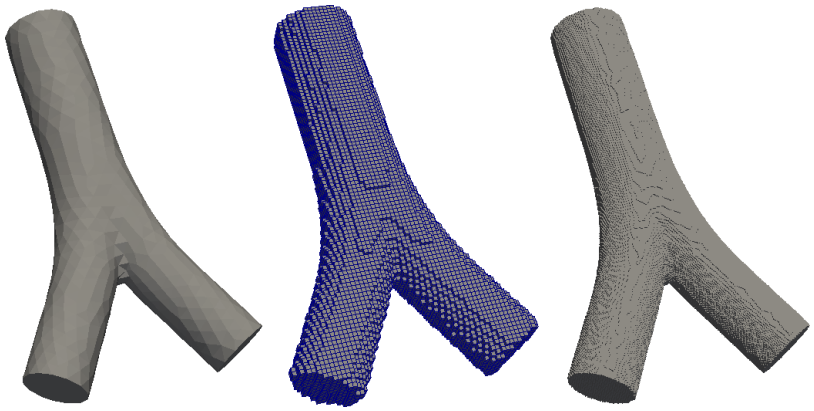


Figure 3.20: Considered bifurcation: original geometry (left), discretisation into a grid with 43,966 nodes (mid) and 2,240,764 nodes (right).

QUANTITY	SYMBOL	VALUE
characteristic length	L	$0.02\,\mathrm{m}$
characteristic velocity	U	$0.0075, 0.0375, 0.075, 0.375\,\mathrm{m}^2/\mathrm{s}$
kinematic viscosity	ν	$0.15 \cdot 10^{-4}\,\mathrm{m}^2/\mathrm{s}$
Reynolds number	Re	$10, 50, 100, 500$
fluid density	ρ^F	$1.205\,\mathrm{kg}/\mathrm{m}^3$
particle density	ρ^P	$998.2\,\mathrm{kg}/\mathrm{m}^3$
particle radius	R	$18.9, ..., 735.5\,\mu\mathrm{m}$
Stokes number	St	$0.01, ..., 15$
Diffusivity	D	$10^{-6}\mathrm{m}^2/\mathrm{s}$

Table 3.5: Model parameter for the simulation of dilute particle-laden flows in a bifurcation.

the magnitude of the mean entrance velocity U have been chosen, while all other values represent the simulation setup of water droplets in air.

The computations are performed for various Reynolds numbers, which is achieved by applying different entrance velocities. Despite that the flow pattern remains mainly laminar. The described stabilisations have to be applied for $U = 0.075\,\mathrm{m/s}$ and $U = 0.375\,\mathrm{m/s}$, since the Péclet number becomes large, although the chosen diffusivity already exceeds the realistic value by several orders of magnitude. For both components the Smagorinsky model is applied with the constant $C_S = 0.01$ for the computations.

DISCUSSION

The results of the particle flow field are shown in Figure 3.21 for the absence of inertia and particles with Stokes numbers 0.63 and 10. The indicated streamlines imply a deposition rate increasing with St. The inflow of particles is constant over time, hence the computation yields a steady state solution, shown in Figure 3.22. The escape rate is given by

$$E(St) = \frac{N^{esc}(St)}{N^{inj}}\,, \tag{3.30}$$

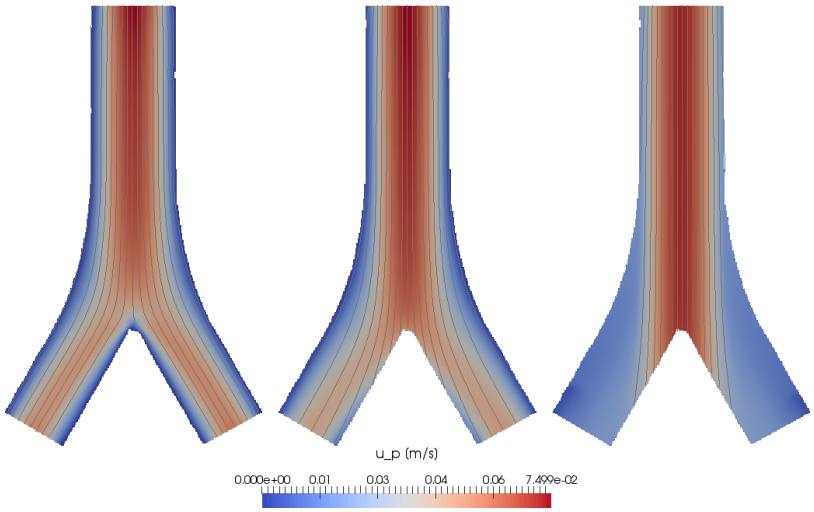


Figure 3.21: Slice through the bifurcation showing the flow field for air (left) and particles with $St = 0.63$ (mid) and $St = 10$ (right) with streamlines.

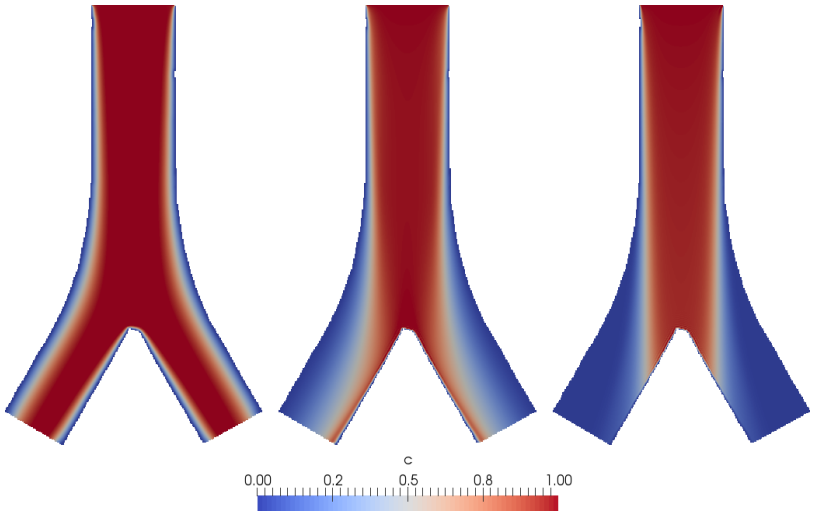


Figure 3.22: Slice through the bifurcation showing the distribution of particles not affected by inertia (left), with $St = 0.63$ (mid) and with $St = 10$ (right).

with $N^{esc}(St)$ and N^{inj} being computed by integration of the populations entering or leaving the computational domain through the respective boundaries.

The results in Figure 3.23 show the escape rate before correction and therefore the impact of a non-physical diffusion coefficient. Even for small St a considerable amount of particles deposits, although they are expected to directly follow the streamlines of the fluid and therefore have no wall interaction. This demonstrates the necessity of post-processing the computed data. The corrected escape rate is shown in Figure 3.24. Although the loss of particles due to diffusion has been reduced, deviations for different D can still be observed for Stokes numbers between 0.1 and 1, where less diffusivity yields higher escape rates.

To show grid independence of the solution, the computations have been performed on different resolutions (see Figure 3.25) with $Re = 50$ and $D = 10^{-6} m^2/s$ for the lattice spacings $h_1 = 1.04 \cdot 10^{-3} m$, $h_2 = 0.52 \cdot 10^{-3} m$,

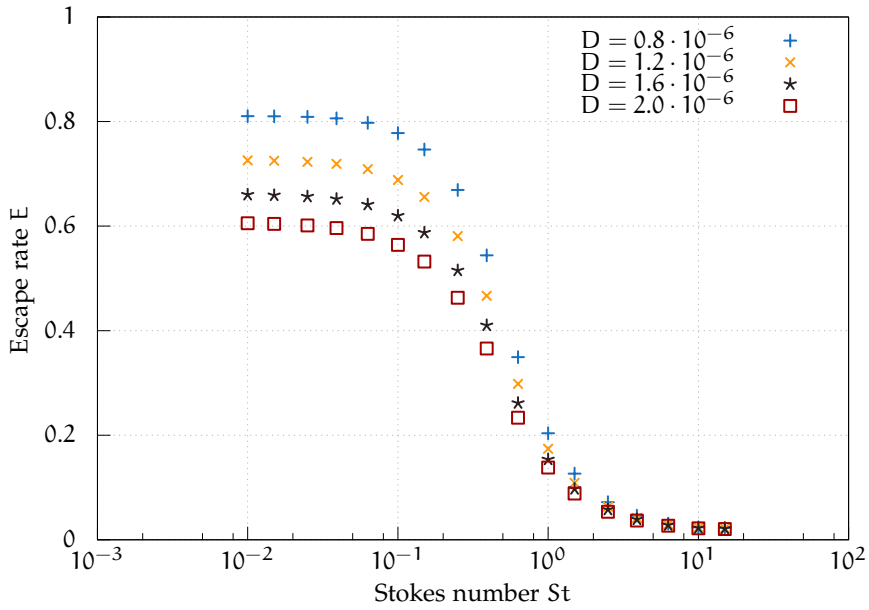


Figure 3.23: Escape rate plotted against the Stokes number for various diffusion coefficients, with Re = 50 on a grid with 970896 nodes.

$h_3 = 0.35 \cdot 10^{-3}$m and $h_4 = 0.26 \cdot 10^{-3}$m in a diffusive scaling ($\Delta_t \sim h^2$). Labelling the corresponding sets of escape rates with E_i, $i = 1, \ldots, 4$, the EOC becomes

$$EOC_{j,k} = \frac{\ln \|E_j - E_4\|_2 - \ln \|E_k - E_4\|_2}{\ln h_j - \ln h_k}. \tag{3.31}$$

For the presented results this yields $EOC_{1,2} \approx 1.38$ and $EOC_{2,3} \approx 2.94$. By varying resolutions the main deviation is found for Stokes numbers between 0.1 and 1 also. With respect to the diffusion coefficient a diffusive EOC EOC^D can be computed analogously by the results shown in Figure 3.24. For its computation the h_i in Equation (3.31) are replaced by $D_1 = 2 \cdot 10^{-6}\text{m}^2/\text{s}$, $D_2 = 1.6 \cdot 10^{-6}\text{m}^2/\text{s}$, $D_3 = 1.2 \cdot 10^{-6}\text{m}^2/\text{s}$ and $D_4 = 0.8 \cdot 10^{-6}\text{m}^2/\text{s}$. For Re $= 50$ and $h = 0.26 \cdot 10^{-3}$m this yields $EOC^D_{1,2} \approx 1.3$ and $EOC^D_{2,3} \approx 1.94$.

For further validation the results are compared to those obtained by Vasconcelos et al. [179] and Henn et al. [82]. In both articles an EL approach has been applied to an identical geometry for different velocities. While in the first one the computations have been performed within a finite volume framework, LBM has been used in the latter. The results in Figure 3.26 show that the escape rates coincide for small Reynolds numbers, whereas for Re $\geqslant 100$ a deviation towards higher deposition can be seen for St $\in [0.1, 1]$. This has also been observed by Henn et al. , while Vasconcelos et al. found a similar trend, but for St > 0.9. A comparison to their results is shown in Figures 3.27 and 3.28. The deviations to other results might arise from the difference between the EL and EE approach, used for the mathematical modelling. The latter has been shown to be sensitive in this range of Stokes numbers regarding the diffusion coefficient.

3.6.1 *Summary*

When applying common EL approaches, the simulation of particle deposition can lead to large computational effort. This is the case especially in a

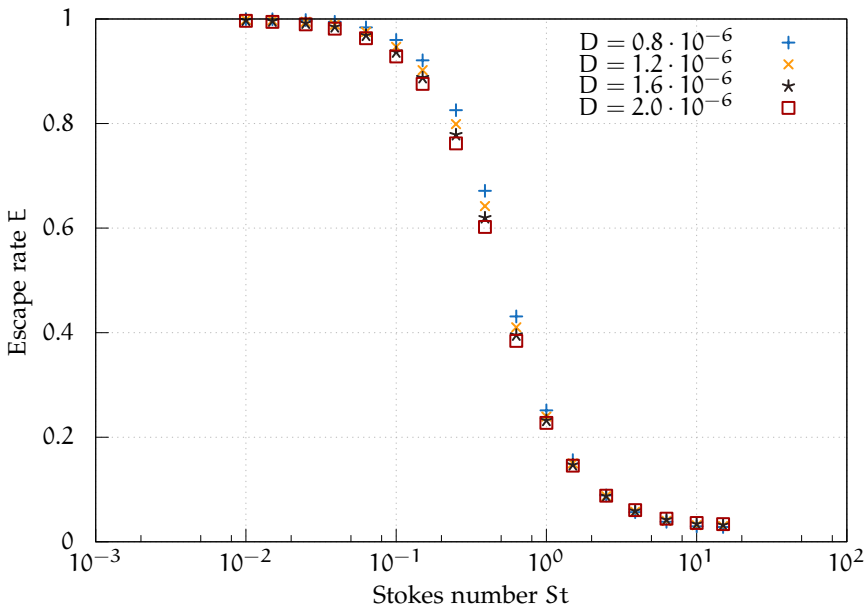


Figure 3.24: Corrected escape rate plotted against the Stokes number for various diffusion coefficients, with $Re = 50$ on a grid with 970896 nodes.

complex geometry if an adequate amount of particles has to be considered to reach a desired level of accuracy. An EE approach has been extended to the domain of dilute and inertial particles, to take advantage of its easy parallelisability in the framework of LBM. As this method allows to formulate boundary conditions on a mesoscopic scale, a new capture boundary has been introduced. It achieves more flexibility in the modelling and can be further extended. Also the problem of distorted results caused by the stabilisation approach has been solved by normalising the escape rate with a reference value.

The developed scheme has been validated in the test case of the application to a bifurcation and it has been shown that viable results are obtained. Respecting the artificial diffusion by a proposed reasonable correction of the obtained data with computed reference values, the solution is found to fit the physical model.

For Stokes numbers less than 0.1 the particles traverse the geometry unhindered, while more than 95% are deposited for $St > 10$. The transition region shows a high sensitivity with respect to parameter variation, especially for $St \in [0.1, 1]$, whereas the limits of this range are just weakly affected.

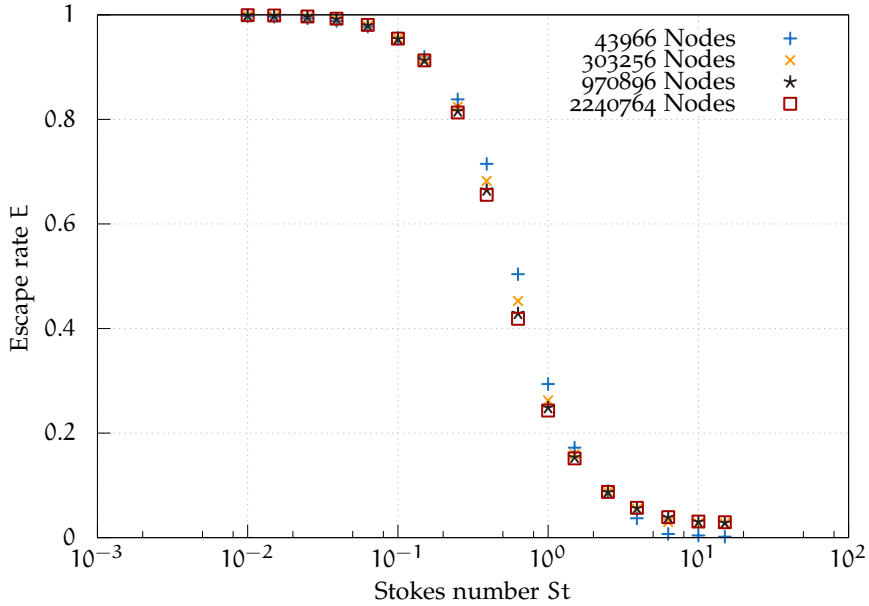


Figure 3.25: Escape rate plotted against the Stokes number for different resolutions, with Re $= 50$ and D $= 10^{-6}\mathrm{m}^2/\mathrm{s}$, showing grid independence of the results.

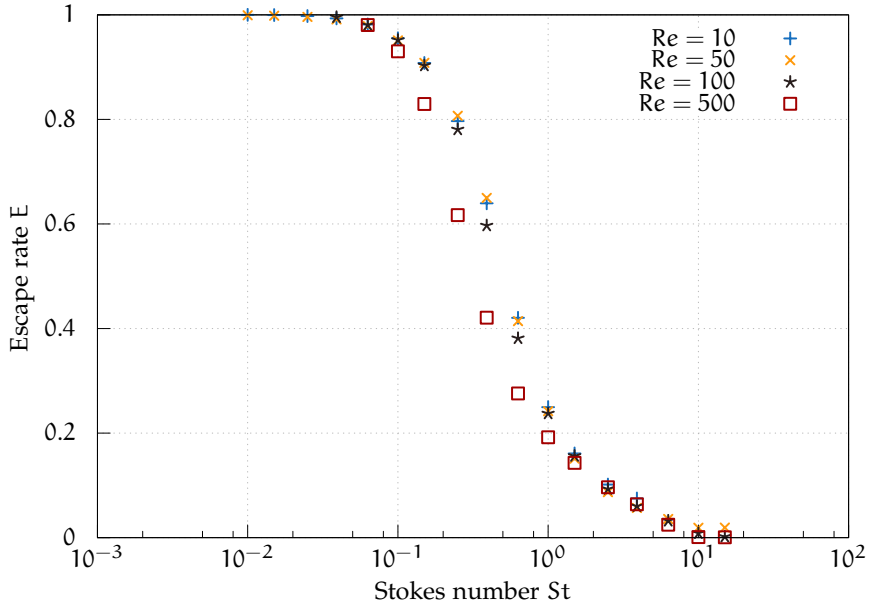


Figure 3.26: Escape rate plotted against the Stokes number for various Reynolds numbers, with Re $= 50$, D $= 10^{-6}\mathrm{m}^2/\mathrm{s}$ on a grid with 970 896 nodes.

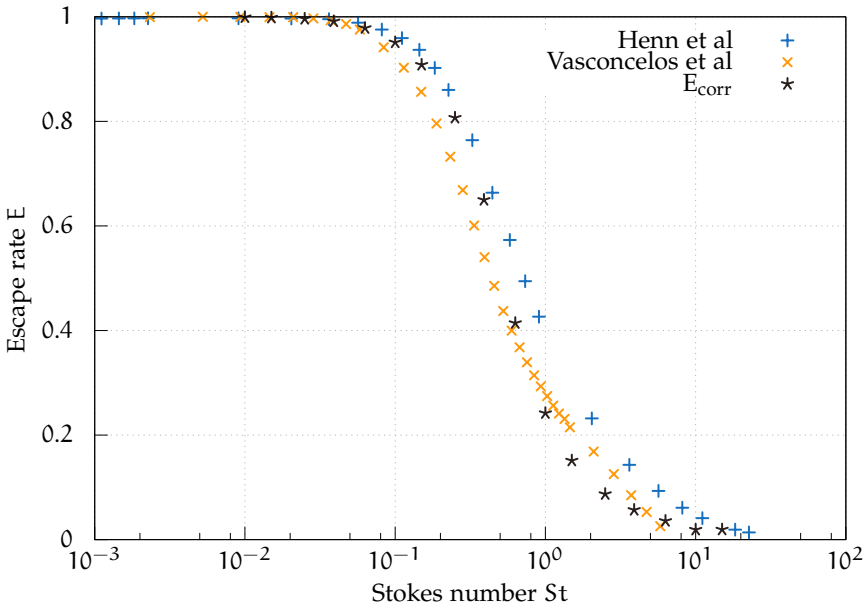


Figure 3.27: Escape rate plotted against the Stokes number for Re = 50 and D = $10^{-6} m^2/s$ in comparison to the results obtained by Henn et al. [82] and Vasconcelos et al. [179]. Denoted by $E_{corr} = E/E_0$ is the corrected escape rate.

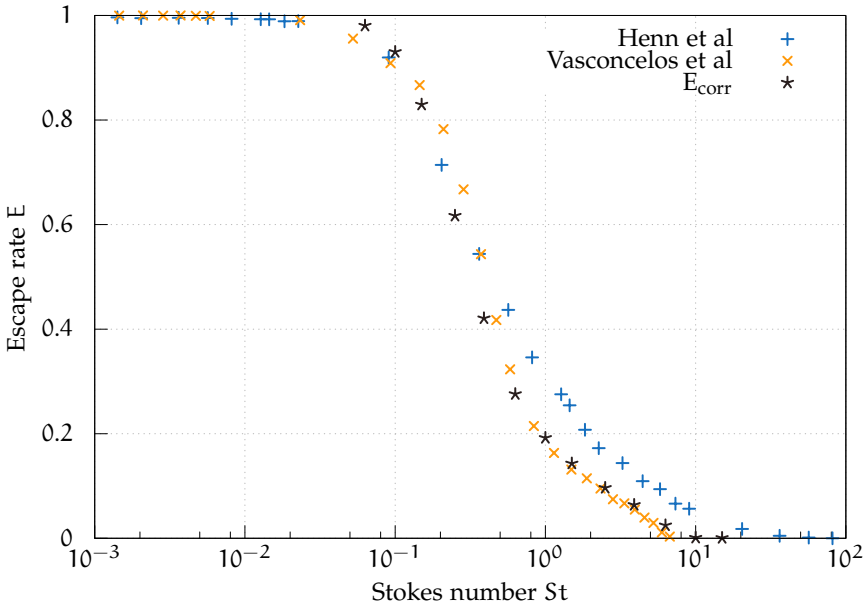


Figure 3.28: Escape rate plotted against the Stokes number for Re = 500 and D = $10^{-6} m^2/s$ in comparison to the results obtained by Henn et al. [82] and Vasconcelos et al. [179]. Denoted by $E_{corr} = E/E_0$ is the corrected escape rate.

4 PARTICULATE FLOWS: TWO WAY COUPLING

When particles move through a fluid, a certain amount of fluid, which was initially at rest is displaced and starts moving. This is a minor factor for few particles of small size, but if the particle concentration and/or particle sizes increase, the growing impact of physical processes that have been neglected in the previous chapter require to model the back coupling of the particles on the fluid. Such models are said to be *two way coupled*. Processes that require two way coupling can be e. g. the clogging of a filtration device [117] or the movement of sediment in a river. It is obvious, that also particle-particle interaction has to be respected during filtration simulations, as a filter does not clog if all filtered particles deposit at the exact same position. Besides that, it would be non-physical. Simulations respecting also inter-particle collisions are said to be *four way coupled*.

To include back coupling in our simulations, we first have to reformulate the mathematical model introduced in Section 3. We therefore divide the domain $\Omega \subset \mathbb{R}^d$ into a fluid part $\Omega^F(t) \subset \Omega$ and $N \in \mathbb{N}$ pairwise disjoint rigid particles $\Omega_k^P(t) \subset \Omega,\ k = 1, \dots, N$, such that at all times $t \in I = [0, T]$,

$$\Omega^F(t) \cup \bigcup_{k=1}^{N} \Omega_k^P(t) = \Omega \qquad \text{for } t \in I\,. \tag{4.1}$$

Using the particle density $\rho_k^P : \Omega_k^P \times I \to \mathbb{R}$ and the total mass of a rigid body $M_k = \int_{\Omega_k^P(t)} \rho_k^P(\underline{x}, t)\, d\underline{x}$, the center of mass of each subdomain $\Omega_k^P(t)$ is defined as

$$\underline{X}_k(t) := \frac{1}{M_k} \int_{\Omega_k^P(t)} \rho_k^P(\underline{x}, t)\, \underline{x}\, d\underline{x}\,.$$

Although we defined the particle density to be depending on space and time, throughout this thesis we assume it to be constant.

Again we use capital letters for Lagrangian notation and properties of Lagrangian particles and lower case letters for quantities seen from an Eulerian point of view. The fluid domain is governed by the Navier–Stokes Equation (NSE) in the following form

$$\partial_t \underline{u}^F + (\underline{u}^F \cdot \nabla)\underline{u}^F = \frac{1}{\rho^F} \nabla \cdot \underline{\underline{\sigma}}(\underline{u}^F, p) + \underline{f} \qquad \text{in } \Omega^F \times I\,, \tag{4.2a}$$

$$\nabla \cdot \underline{u}^F = 0 \qquad \text{in } \Omega^F \times I\,, \tag{4.2b}$$

$$\underline{u}^F = \underline{u}_0^F \qquad \text{on } \Gamma(t) \times I\,, \tag{4.2c}$$

plus adequate initial conditions. Here $\Gamma(t) = \partial\Omega^F \setminus \bigcup_k \partial\Omega_k^P(t)$ represents the part of the fluid boundary that does not belong to a particle. A boundary condition at the particle boundaries will be stated later. In this work, we only treat Newtonian fluids, where the stress tensor $\underline{\underline{\sigma}}(\underline{u}^F, p) = 2\nu \underline{\underline{S}}(\underline{u}^F) - p\underline{\underline{Id}}$, and the strain rate $\underline{\underline{S}}(\underline{u}^F) = 1/2(\nabla \underline{u}^F + (\nabla \underline{u}^F)^\top)$.

The rigid particles are seen in a Lagrangian frame and move according to

$$M_k d_t \underline{U}_k(t) = \underline{F}_k(t) \qquad \text{for } t \in I\,, \tag{4.3a}$$
$$d_t \underline{X}_k(t) = \underline{U}_k(t) \qquad \text{for } t \in I\,, \tag{4.3b}$$
$$d_t \left(\underline{\underline{I}}_k(t) \underline{\omega}_k(t) \right) = \underline{T}_k(t) \qquad \text{for } t \in I\,, \tag{4.3c}$$
$$d_t \underline{\theta}_k(t) = \underline{\omega}_k(t) \qquad \text{for } t \in I\,, \tag{4.3d}$$

for $k = 1, \dots, N$. Here $\underline{U}_k : I \to \mathbb{R}^d$ is the translational velocity of the centre of mass $\underline{X}_k$ of the rigid domain Ω_k^P. The angle of rotation is denoted by $\underline{\theta}_k : I \to \mathbb{R}^d$, its derivative, the rotational velocity by $\underline{\omega}_k : I \to \mathbb{R}^d$ and the torque acting on particle k by $\underline{T}_k : I \to \mathbb{R}^d$. The elements of the moment of inertia tensor $\underline{\underline{I}}_k : I \to \mathbb{R}^{d \times d}$ of particle k for rotation around an axis through the center of mass are defined by

$$I_{k,\alpha\beta} := \int_{\Omega_k^P} \rho(\underline{r}) \underline{r}^2 \delta_{\alpha\beta} - r_\alpha r_\beta \, d\underline{x} \qquad \text{for } \alpha, \beta = 1, \dots, d,$$

where $\underline{r} = \underline{x} - \underline{X}_k = (r_1, \cdots, r_d)^\top$. The local velocity $\underline{u}_k^P : \Omega_k^P \times I \to \mathbb{R}^d$ at a point in the particle domain can then be computed by

$$\underline{u}_k^P(\underline{x}, t) = \underline{U}_k^P(t) + \underline{\omega}_k(t) \times (\underline{x} - \underline{X}_k(t)) \qquad \text{in } \Omega_k^P \times I\,.$$

For calculations in two dimensions, the three dimensional angular velocity is defined as $\omega_k(t) := (0, 0, \omega_k)(t)$ and the cross product as $\omega_k \times \underline{x} := (-\omega_k x_1, \omega_k x_0, 0)$ for a two dimensional vector $\underline{x} = (x_0, x_1)$.

Equations (4.2) and (4.3) are further coupled by the hydrodynamic forces

$$\underline{F}_k = \int_{\partial\Omega_k^P} \underline{\underline{\sigma}}(\underline{u}^F, p) \underline{n} \, d\underline{S} + \sum_j \underline{F}_k^{(j)} \qquad \text{for } k = 1, \dots, N \tag{4.4}$$

and torques

$$\underline{T}_k = \int_{\partial\Omega_k^P} (\underline{x} - \underline{X}_k) \times \underline{\underline{\sigma}}(\underline{u}^F, p) \underline{n} \, d\underline{S} + \sum_j \underline{T}_k^{(j)} \quad \text{for } k = 1, \dots, N\,, \tag{4.5}$$

where $\underline{n}$ is the unit outwards pointing normal on the particle boundary $\partial\Omega_k^P$ and $\underline{F}_k^{(j)}$ and $\underline{T}_k^{(j)}$ are additional forces and torques that may apply.

Finally we enforce a *no-slip condition* determining that the fluid velocity at the particle boundaries equals the particle velocity

$$u^F(\underline{x}, t) = u_k^P(\underline{x}, t) \qquad \text{on } \partial\Omega_k^P \times I\,. \tag{4.6}$$

Two way coupled Euler–Lagrange (EL) simulation methods can be categorised in *Fixed Mesh* methods and *Body Conformal Mesh* methods, such as the Arbitrary Lagrangian–Eulerian (ALE) method [89]. Due to the algorithmic complexity as well as the high computational cost for the re-meshing, the latter methods are limited to the simulation of only a few particles. Such EL methods, where the objects are not resolved in the fluid, need a drag force model such as the one introduced in Equation (3.7) enabling a two-way coupling of the fluid with the moving object. In the following, this type of particles are called *subgrid scale particles*. With it, the simulation of millions of particles becomes feasible [177]. However, the objects' shapes are not arbitrary but limited to rather simple primitives [184]. The most prominent *Fixed Mesh* methods for fully resolved objects are the Immersed Boundary

Method (IBM) [144] and Fictitious Domain Method (FDM) [66, 68]. Both employ multiple grids, one Eulerian for the fluid phase and one Lagrangian for each particle. The Lagrangian grids move with the particle velocity that results from the applied forces. The back coupling onto the fluid can be realised by forcing schemes as introduced in Section 2.3.3. Contrary to Euler–Euler (EE), EL methods in principle allow the simulation of arbitrary shapes [135]. However, there seem to exist only simulations for single non-spherical particles [3, 32, 50, 91, 104, 122, 155, 173, 174, 194] which may be due to the lack of knowledge of collision models for arbitrarily shaped particles [3, 50], high computational demand or complex implementation [174].

The Lattice Boltzmann Method (LBM) has been shown to be able to simulate particulate flows since Ladd [113, 114] used an EL method in 1994 and most recently by an EE method [176]. Similar to Ladd, Götz et al. [69] incorporated spherical particles by a boundary condition method and demonstrated its capacity by simulating up to 150.000 particles. An IBM was introduced to LBM context by Feng and Michaelides [56, 57] and further improved by Niu et al. [139], Wu and Shu [191] and Hu et al. [90]. Shi and Phan-Thien [162] proposed an FDM for particulate flows in the LBM framework, which was extended by Nie and Lin [137, 138]. All of them applied their methods to the simulation of spherical particles. Another promising EL *Fixed Mesh* approach has been proposed by Nakayama and Yamamoto [135]. The *Smoothed Profile Method* models the boundaries of the objects by a continuous transition between the fluid and the particle velocity. Later, it was applied to LBM by Jafari, Yamamoto and Rahnama [97]. In principle, the method allows the simulation of particles of arbitrary shapes. However, until now it has not been applied to other than spherically shaped particles.

The remainder of this chapter is structured as follows: In Section 4.1.1, we introduce techniques to efficiently determine inter-particle collisions by reducing the amount of potentially interacting pairs. We introduce two representatives of different classes of contact detection algorithms and compare them in terms of Central Processing Unit (CPU) time. Subsequently we introduce a collision model commonly used for the Discrete Element Method (DEM).

Subsequently, we will approximately solve the above stated system of equations by combining an LBM and a DEM. Depending on the particle size we distinguish between particles smaller than the lattice spacing and particles larger than the lattice spacing. In Section 4.2 we cover subgrid scale particles and turn towards arbitrarily shaped resolved particles in Section 4.3. New approaches based on the Homogenised Lattice Boltzmann Method (HLBM) are proposed for particles of both scales in Section 4.2.1 and Section 4.3.1. Section 4.3 and 4.5 are based on article [81], submitted to *Particuology*.

4.1 PARTICLE-PARTICLE INTERACTION

This section treats the modelling of the interaction of particles with each other. We begin by introducing techniques to efficiently determine inter-particle collisions by reducing the amount of potential pairs for interaction. Two representatives of different classes of contact detection algorithms are introduced and compared to each other in terms of CPU time. Subsequently we introduce the spring-dashpot collision model, that models pairwise particle–particle interaction as inelastic collisions.

4.1.1 *Collision Detection*

Computation of particle–particle interaction is a key feature in DEMs and also necessary for some types of fluid–particle flows. This is very resource demanding, as in every time step all possible particle pairs have to be checked for interaction, which is an $\mathcal{O}(N^2)$ operation for the most primitive algorithm. However, the complexity can be reduced by considering geometric information and the properties of the interacting forces. One has to distinguish between long ranged forces (such as gravitational force, magnetic force, force between static electrically charged particles) and short ranged forces (particle collision). Generally speaking the longer the range the higher are the computational costs, as more interacting particle pairs exist. Early algorithms used to truncate the interaction potential, neglecting the far field interaction, which leads to poor approximations [51]. Today, high-speed computers allow more modern approaches, such as the particle mesh Ewald method, which is based on Ewald sums [52], which are able to significantly reduce computational costs. The idea behind the method is to split the interaction potential in a short ranged and a long ranged sum and use a Fourier transform on the latter, leading to faster convergence. The particle mesh Ewald method has been shown to be of $\mathcal{O}(N \log N)$ complexity for computation of a periodic system of N particles [43].

Paul Peter Ewald *(1888–1985) German physicist*

Joseph Fourier ☾ *(1768–1830) French mathematician and physicist*

In the following we will limit ourselves to particle collisions, which are modelled as a force with range in order of the particle radius $R > 0$. Indeed, the introduced methods can be applied to forces with longer range, however, this is not recommended as more efficient methods exist for such cases.

DEFINITION 4. Two particles with positions $\underline{X}_i$ and $\underline{X}_j$ and radii R_i and R_j *collide*, if their distance $d_{i,j} = \left\|\underline{X}_i - \underline{X}_j\right\|_2$ is less than the sum of their radii

$$d_{i,j} < R_i + R_j \ . \tag{4.7}$$

The easiest, but most expensive option to detect all particle collisions is therefore to check Inequality (4.7) for each possible combination of two particles. For N particles $N(N-1)/2$ possible combinations exist, what leads to a computational complexity of $\mathcal{O}(N^2)$. In our experience this is a limitation to simulations of a few thousand particles. It is therefore of great interest to reduce the number of potential collisions pairs. Several algorithms exist, which use geometric information and sort the particles according to their position. There are two main classes of algorithms for this purpose. The first class is tree based and of complexity $\mathcal{O}(N \log N)$, while the second class is grid based and of complexity $\mathcal{O}(N)$. In the following we introduce an example of each class and compare them in terms of actual CPU time.

k-d TREE

A k-dimensional tree, or k-d tree is a binary tree, used to sort points in a k-dimensional space. Starting with a cuboid containing all particles, it is recursively subdivided by an axis aligned splitting hyperplane. In order to obtain a balanced tree, the splitting plane is positioned in the median of the particle positions in direction of the normal of the splitting plane. For example, if the x-axis is chosen, all particles with x-coordinate less than the median are left of the hyperplane, right otherwise. For each step down the tree the axis of the splitting plane is periodically changed. For example starting with a plane normal to the x-axis, the two newly created cuboids are split by an hyperplane normal to the y-axis, followed by a cut normal to z-axis. The recursion can be stopped if a given number of particles remains

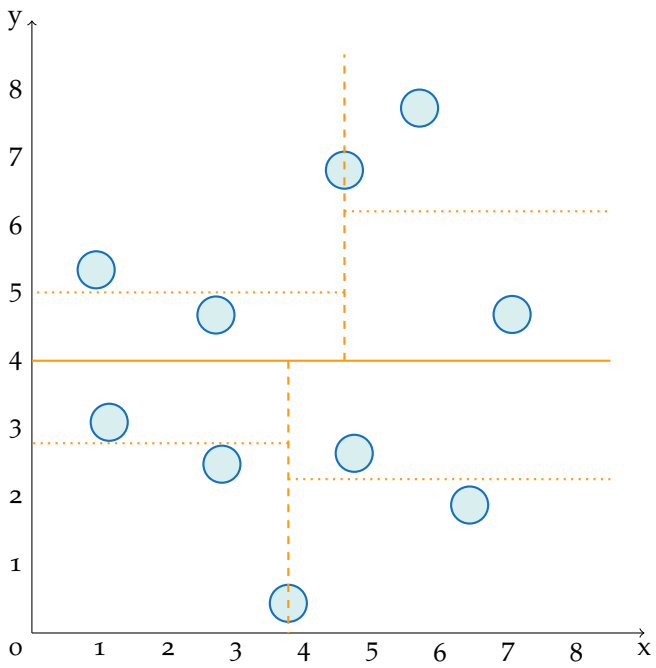


Figure 4.1: The first three steps of a k-d tree algorithm. In each step the particle set is split by the hyperplane through the median, with periodically changing normal. — step 1, - - step 2, ···· step 3

in a cuboid or performed until each cuboid contains only one single particle. The main effort during the construction is finding the median of the particles positions in each layer of the tree. Therefore the complexity for creating a k-d tree depends heavily on the chosen sorting algorithm. Using an algorithm that finds the median in $\mathcal{O}(N)$ steps one can construct a balanced k-d tree in $\mathcal{O}(N \log N)$ steps. However such algorithms are usually very demanding to implement. With algorithms that are less complicated, but still find the median in $\mathcal{O}(N \log N)$ steps, such as merge sort and heap sort the tree can be build in $\mathcal{O}(N \log^2 N)$ steps [21].

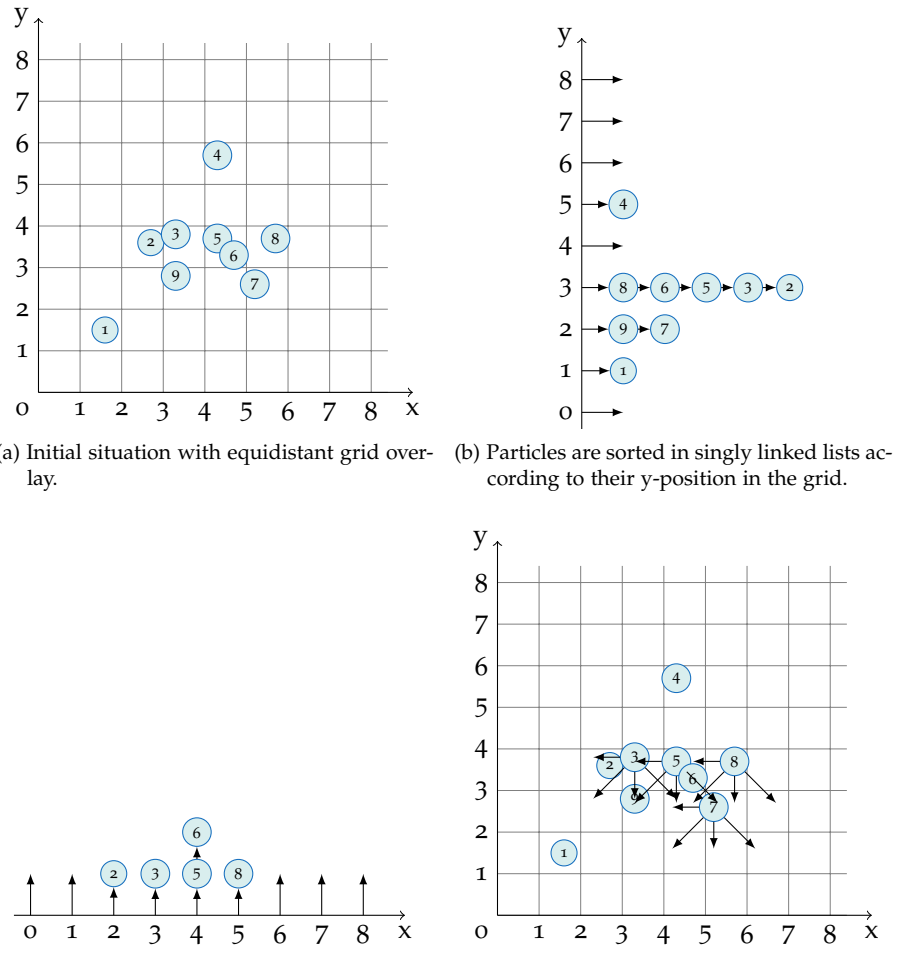
The constructed tree can then be efficiently searched by nearest neighbour and range searches. Lee and Wong [119] determined that the worst case run-time for a region search in a k-d tree with N elements is of order $\mathcal{O}(k \cdot N^{1-1/k})$.

A k-d tree is implemented in `OpenLB` by means of the nanoflann[1] library, which itself is a fork of the FLANN[2] library.

MUNJIZA NBS

Grid based collision detection methods subdivide the domain Ω into identical square (2D) or cubical (3D) cells with edge length $h > 0$, which is not necessarily the same h as in the LBM. If one chooses $h > 2R$, where R is the maximal radius of all particles, and sorts the particles in the grid, potential collision partners are located in immediate neighbouring nodes only. This can be done by using a two or three dimensional array and a single loop over the N particles, hence the algorithm needs $\mathcal{O}(N)$ steps. However, the memory demand for this simple case is immense especially for dilute suspension, when the number of cells is large compared to the number of

1 http://github.com/jlblancoc/nanoflann
2 Fast Library for Approximate Nearest Neighbours, http://www.cs.ubc.ca/research/flann/



(a) Initial situation with equidistant grid overlay.

(b) Particles are sorted in singly linked lists according to their y-position in the grid.

(c) Particles of one row (here y=3) are sorted in singly linked lists according to their x-position in the grid.

(d) Stencil indicating the grid cell that are searched for particle contact tests.

Figure 4.2: Grid based contact detection algorithm by Munjiza and Andrews [133].

particles. To tackle this challenge Munjiza and Andrews [133] developed the No Binary Search (NBS), which computes potential collision partners using singly linked lists. In the following we will reflect the basic algorithm in two dimensions. For usage in `OpenLB` the described algorithm has been expanded to three dimensions. A detailed description on how to efficiently store the linked lists can be found in Munjiza [134].

We introduce the algorithm using the 2D example illustrated in Figure 4.2. Start by dividing the domain $\Omega = [0, x_{max}) \times [0, y_{max}) \in \mathbb{R}^2$ in cells

$$c_{(i,j)} = [i\,h, (i+1)\,h) \times [j\,h, (j+1)\,h)$$

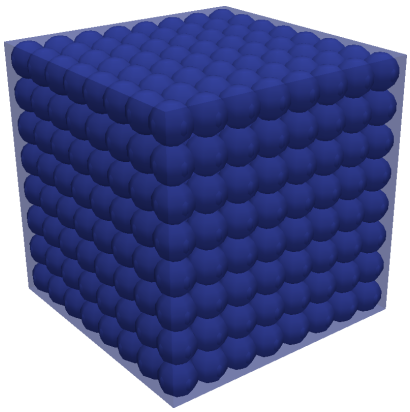
for $i = 0, \ldots, I,\ I < x_{max}/h < I+1,\ I \in \mathbb{N}$ and $j = 0, \ldots, J,\ J < y_{max}/h < J+1, J \in \mathbb{N}$. Create a singly linked list for each row $r_j = \bigcup_{i=0}^{I} c_{(i,j)}$ and sort the particles for an arbitrary but fixed numbering $p_k,\ k \in \{1, \ldots, N\}$. An exemplary situation is given in Figure 4.2a. Figure 4.2b displays the list for rows r_j. Particle p_1 in $c_{(1,1)}$ is the only element in row r_1. Then p_2 is sorted as first element in the list connected to r_3. p_3 pushes back p_2 in r_3, p_4 is sorted in r_5, p_5 pushes back p_3 in r_3 and so on. At this time all lists r_j are marked as `new`.

Now loop over all particles. If the row r_j contains particle p_k and is marked as `new` sort the particles in row r_j in singly linked list mapped to the cells $c_{(i,j)}$, $i = 0, \dots, I$, of row r_j in the same manner as before and mark list r_j as `old` and all list connected to $c_{(i,j)}$, $i = 0, \dots, I$, as `new`. Also sort the particles of r_{j-1} into list connected to the cells $c_{(i,j-1)}$, $i = 0, \dots, I$, of row r_{j-1}. Figure 4.2c shows the linked lists for the cells of row r_3.

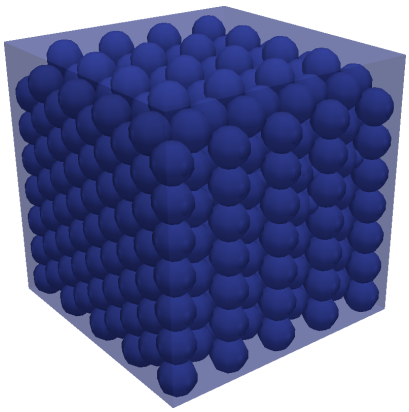
Now traverse the particles in the list mapped to r_j. If a particles belongs to a list of $c_{(i,j)}$, $i = 0, \dots, I$, marked as `new`, then mark $c_{(i,j)}$ as `old` and check for contact between particles in $c_{(i,j)}$ and particles in $c_{(i-1,j-1)}$, $c_{(i,j-1)}$, $c_{(i,j+1)}$ and $c_{(i-1,j)}$, according to the stencil shown in Figure 4.2d.

Using this technique only the lists of two rows have to be kept in the memory at the same time. Also no loops over the cells occur, which leads to a detection time in $O(N)$, according to Munjiza and Andrews [133].

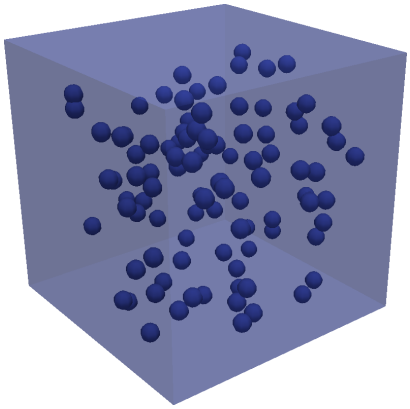
COMPARISON



Case A: Cubical distribution



Case B: Hexahedral distribution



Case C: Random distribution

Figure 4.3: Particle distribution for contact detection test cases.

We will now compare the k-d tree based algorithm and the NBS by Munjiza and Andrews [133], as introduced in the previous two sections, in terms of CPU time. Therefore three test cases are constructed. For a parameter $h \in \mathbb{R}$ particles with centre of mass X_n are distributed in the domain $\Omega = [0, 2]^3 \subset \mathbb{R}^3$ as follows

- Case A (Cubic layout):

$$X_n = \begin{pmatrix} 1/2+i \\ 1/2+j \\ 1/2+k \end{pmatrix} h\,,$$

with $i = 0,\dots,I, j = 0,\dots,J, k = 0,\dots,K$ and $I < 2/h < I+1$, $J < 2/h < J+1$, $K < 2/h < K+1$, $I,J,K \in \mathbb{N}$. This alignment leads to 6 contacts per particle for non-boundary particles.

- Case B (Hexahedral layout):

$$X_n = \begin{pmatrix} 1/2+i\sqrt{3}/2) \\ (1/2+(j+(i \bmod 2)\sqrt{3})/4 \\ (1/2+(k+(j \bmod 2+i \bmod 2)/2 \end{pmatrix},$$

with $i = 0,\dots,I, j = 0,\dots,J, k = 0,\dots,K$ and $I < 4/(\sqrt{3}h) < I+1$, $J < 4/(\sqrt{3}h) < J+1$, $K < 2/h < K+1$, $I,J,K \in \mathbb{N}$. Here the modulo operation mod returns the remainder after Euclidean division. This alignment leads to 12 contacts per particle for non-boundary particles.

- Case C: (Random layout) Uniform spatial random distribution in Ω.

See Figure 4.3 for illustrations. In cases A and B the particle radius is set to $R = 2.1\,h$, in case C, $R = 0.21\,h$. The particle contact detection algorithms are executed 20 times for each setup and number of particles. CPU times are taken for setting up the tree/grid, computing potential pairs and checking for particle contact and averaged. The test cases have been executed on one core of an Intel® Core™ i7-4930K CPU running at 3.40GHz.

Results are shown in Figs. 4.4, 4.5, 4.6 and corresponding Tables 4.1, 4.2 and 4.3. The figures show the absolute time needed for collision detection over the number of particles N. The tables state the amount of particles, the average number of collisions per particle and the measured times for collision detection in seconds. The graphs suggest an expected $N \log N$ behaviour for the k-d tree, however, the time for collision detection using the k-d tree scales almost linearly with the total number of particles in cases A and B. The linear scaling also applies to the NBS, as expected. In cases A and B the CPU times of both collision detection algorithms are comparable, neither the NBS nor the k-d tree based algorithm is clearly faster. Scaled to the number of particles, the absolute time needed in case B is about 10–30 per cent longer as in case A, which might be justified by twice the amount of collisions occurring. However in the case of random distribution, the NBS clearly outruns the k-d tree. Also in absolute numbers both algorithms perform lower than in cases A and B. The NBS is up to a factor 2, the tree based algorithm up to a factor 7 slower. In cases of NBS this might be explained by the fact, that due to the random distribution more than one particle can be placed in a cell of the grid, which leads to a higher number of potential collision pairs. In case of the tree based algorithm, we assume that for cases A and B particles are already pre-sorted due to their structured initialisation. Therefore the sorting algorithm finding the median needs less steps, and thus the tree can be constructed in less time. As case C approximates best the situation during the simulation of a realistic particulate flow the NBS should be preferred over the k-d tree.

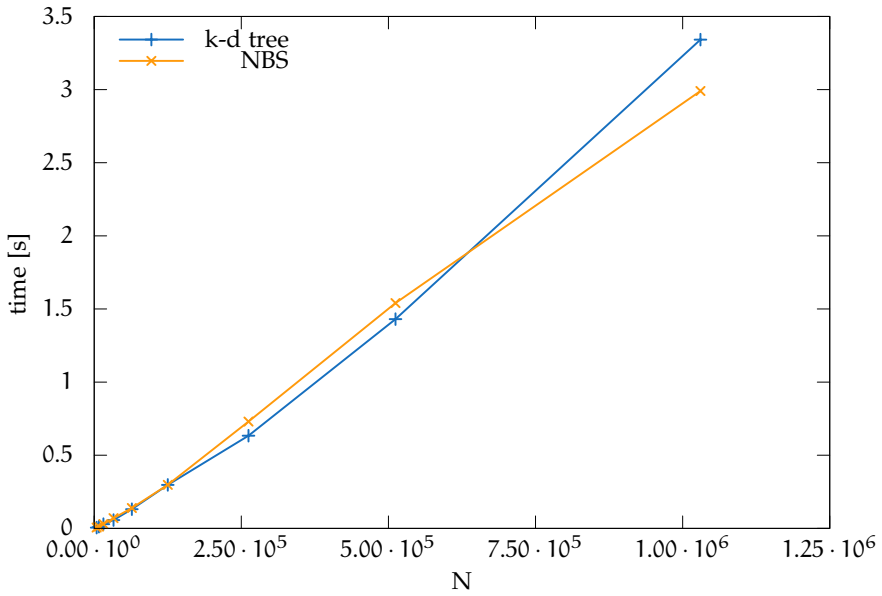


Figure 4.4: CPU times for contact detection. Case A: cubical distribution.

#	COLLISIONS	NBS [s]	k-d TREE [s]
4096	5.62	0.006	0.005
8000	5.70	0.012	0.012
15625	5.76	0.030	0.029
32768	5.81	0.070	0.057
64000	5.85	0.140	0.131
125000	5.88	0.300	0.297
262144	5.91	0.730	0.634
512000	5.93	1.540	1.430
1030301	5.94	2.990	3.342

Table 4.1: Case A: cubical distribution. The table lists the number of particles, the averaged collisions per particle, as well as the run-time for the NBS and the k-d tree algorithm.

SUMMARY

We introduced two algorithms that reduce the $\mathcal{O}(N^2)$ complexity of contact detection of N particles. In theory the k-d tree algorithm achieves $\mathcal{O}(N \log N)$, while the NBS algorithm achieves $\mathcal{O}(N)$ complexity. After implementation the algorithm run-times have been compared for three different scenarios. Particle contact was computed for two structured particle distributions and one random particle distribution. We found that the run-time of both algorithms are comparable for structured particle distribution and up to at least one million particles. In case of random particle distribution the NBS clearly outruns the k-d tree algorithm and should therefore be preferred.

4.1.2 *Collision Models*

The previous section introduced method to detect pairs of colliding particles. Knowing a potential collision a mathematical model of the interaction

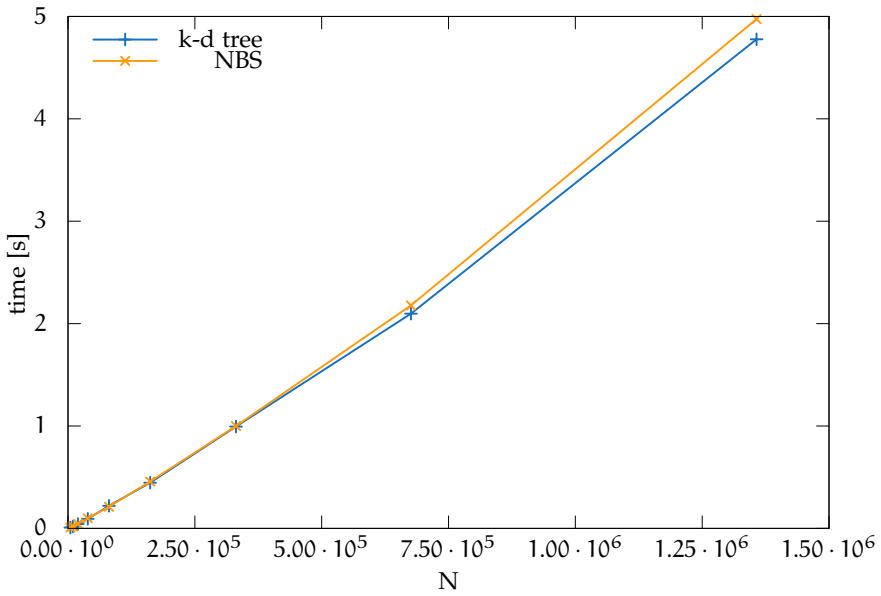


Figure 4.5: CPU times for contact detection. Case B: hexahedral distribution.

#	COLLISIONS	NBS [s]	k-d TREE [s]
4590	10.88	0.008	0.008
9614	11.12	0.019	0.019
19488	11.30	0.044	0.042
39060	11.44	0.098	0.093
80730	11.56	0.211	0.221
161994	11.65	0.457	0.446
331128	11.73	1.000	0.994
675924	11.78	2.178	2.097
1357200	11.83	4.974	4.777

Table 4.2: Case B: hexahedral distribution. The table lists the number of particles, the averaged collisions per particle, as well as the run-time for the NBS and the k-d tree algorithm.

John Lennard-Jones *(1894–1954) British theoretical physicist*

potential has to be applied. Depending on the size and properties of the simulated particles, different forces have to be respected. For atomistic particles (diameter of about 1 Å) the Lennard-Jones [99] potential which models the interaction between a neutral pair of atoms or the DLVO theory [23, 183], which combines van der Waals attraction, electrostatic repulsion and born repulsion may apply. For macroscopic particles (diameter greater then 1 µm) molecular potentials become less important and can be neglected. Therefore macroscopic properties such as recoil forces due to deformation and friction forces become crucial. This is covered by the widely used spring-dashpot model by [40], which is introduced in the following. Additionally agglomeration processes for solid particles have been investigated in the thesis by Kolbe [108].

SPRING-DASHPOT

The spring-dashpot model was initially proposed by Cundall and Strack [40] to describe the mechanical behaviour of assemblies of discs and spheres.

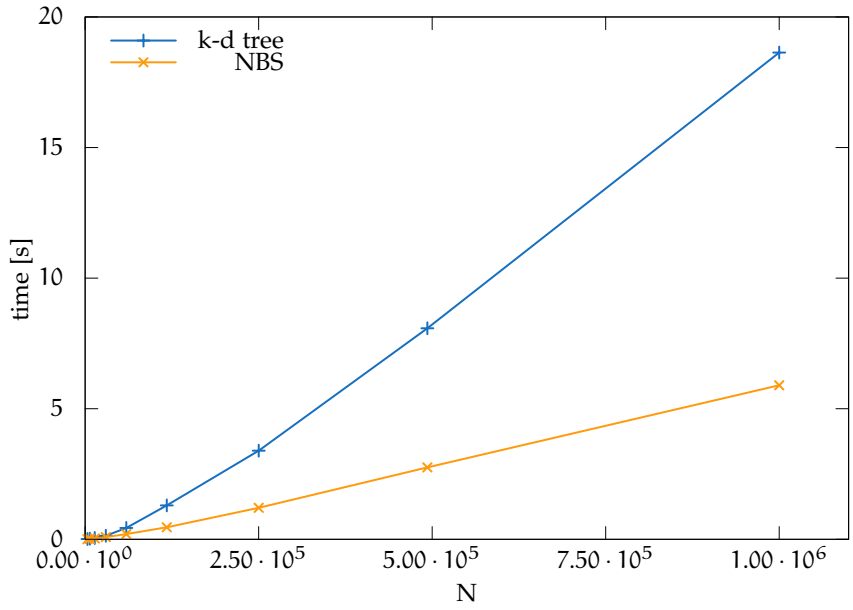


Figure 4.6: CPU times for contact detection. Case C: random distribution.

#	COLLISIONS	NBS [s]	k-d TREE [s]
3375	0.0107	0.007	0.010
6859	0.0095	0.015	0.024
13824	0.0067	0.034	0.052
29791	0.0070	0.085	0.135
59319	0.0066	0.200	0.434
117649	0.0060	0.459	1.297
250047	0.0061	1.205	3.393
493039	0.0057	2.754	8.080
1000000	0.0057	5.897	18.638

Table 4.3: Case C: random distribution. The table lists the number of particles, the averaged collisions per particle, as well as the run-time for the NBS and the k-d tree algorithm.

The idea of the spring-dashpot model is to allow a small overlap of rigid bodies. Depending on the size of the overlap a resetting force is applied. We introduce a version proposed by Yokoi [197].

For two spherical particles j, k, moving with velocity $\underline{U}_j$ and $\underline{U}_k$, rotating with angular velocity $\underline{\omega}_j$ and $\underline{\omega}_k$ and having radii $R_j, R_k > 0$, mass $M_j, M_k > 0$ and centres of mass $\underline{X}_j, \underline{X}_k \in \Omega$ the overlap Δ_d is computed by

$$\Delta_d = R_j + R_k - \|\underline{r}_{j,k}\|_2 \ ,$$

where $\underline{r}_{j,k} = \underline{X}_j - \underline{X}_k$ is the vector connecting the centres of mass. The force acting between the particles due to contact can be separated, a part normal to the particle surface, indexed by n, and a part tangential to the particle surface, in the plane with normal $\underline{r}_{j,k}$, indexed by t,

$$\underline{F} = \underline{F}^n + \underline{F}^t \ .$$

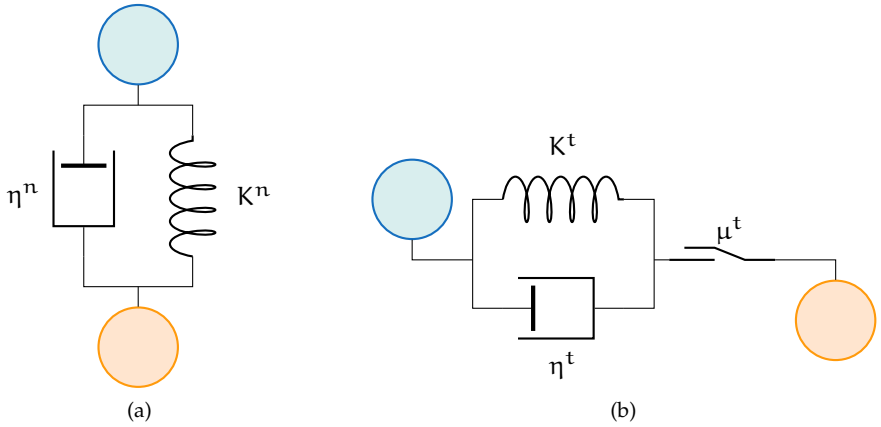


Figure 4.7: Illustration of the spring-dashpot model. K, η, and μ represent the spring constant, the damping constant and the slider constant. The index n indicates normal variables and t tangential variables.

The normal force consists of a spring force and a damping force, computed by

$$\underline{F}^n = 2M(K^n \Delta_d - \eta^n \underline{U}^n_{j,k}) \,,$$

where $M = M_j M_k/(M_j + M_k)$ denotes the reduced mass and $K^n > 0$ and $\eta^n > 0$ the normal spring constant and the normal damping constant respectively. The normal component of the relative velocity of the particles is denoted by $\underline{U}^{j,k}_n = (\underline{U}_j - \underline{U}_k)\underline{r}^{j,k}/\|\underline{r}^{j,k}\|_2$. An illustration can be found in Figure 4.7a. The tangential fraction of the force is either modelled as stiction, similar to the normal component or as *Coulomb friction*

Charles-Augustin de Coulomb ☾ *(1736–1806) French physicist*

$$\underline{F}^t = \min(\|\underline{h}^t_{j,k}\|_2, \mu^t \|\underline{F}^n\|_2) \frac{\underline{h}^t_{j,k}}{\|\underline{h}^t_{j,k}\|_2} \,,$$

where $\mu^t \|\underline{F}^n\|_2$ is the *Coulomb friction* with friction parameter $\mu^t > 0$ and

$$\underline{h}^t_{j,k} = -2M(K^t \underline{U}^t_{j,k} \Delta_t - \eta^t \underline{U}^t_{j,k})$$

is the stiction part with the tangential spring constant $K^t > 0$ and the tangential damping constant $\eta^t > 0$. $\underline{U}^t_{j,k} = (\underline{U}_j - \underline{U}_k)\underline{t} + (R_j \underline{\omega}_j + R_k \underline{\omega}_k)$ is the projection of the relative velocity on the tangential plane plus the velocity originating in the particle rotation. The time since first particle contact is denoted by Δ_t and hence $\underline{U}^t_{j,k}\Delta_t$ is the distance travelled since particle contact. An illustration can be found in Figure 4.7b.

The torque on particle j is then computed as

$$\underline{T} = R_j \frac{\underline{h}^t_{j,k} \times \underline{F}^t}{\|\underline{h}^t_{j,k}\|_2} \,.$$

All 5 parameters of the model are material specific and have to be determined experimentally.

For non-spherical particles the model can be adapted considering the overlapping areas instead of Δ_d [63]. The introduced model was implemented in `OpenLB` and used during the simulations of an air/water two-phase flow through the engine compartment of a car in the diploma thesis by Wünsche [193].

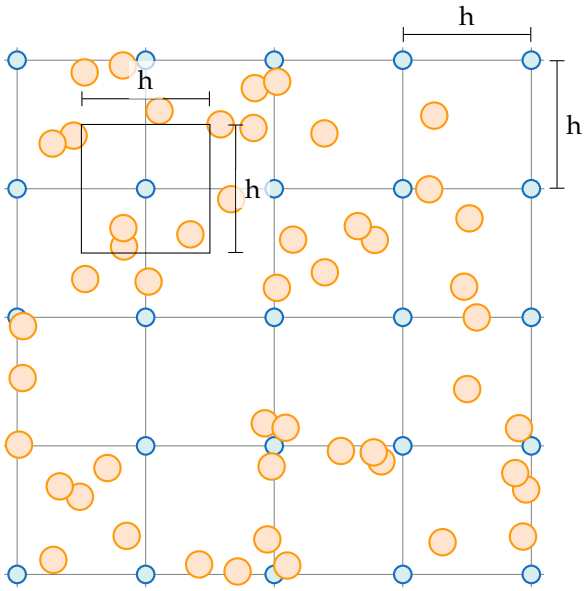


Figure 4.8: Schematic illustration of subgrid scale particle (red circles) and the lattice nodes (blue circles). The square indicates the control volume.

4.2 SUBGRID SCALE PARTICLES

This section is devoted to the simulation of two and four way coupled particulate flows of particles with diameter less than the lattice spacing of the underlying LBM. One lattice node can contain several Lagrangian particles, as illustrated in Figure 4.8. In this setup, while for particle–particle interaction their radius is respected, in their action on the fluid the particles are interpreted as mass points. Literature on this approach is scarce, known to the author are the articles by Wang et al. [187], Zhang et al. [202], and Xiong, Madadi-Kandjani and Lorenzini [196] only. The authors of the three articles use different approaches to couple particles and fluid. We shortly introduce their methods, and subsequently propose a new method using the HLBM as well as a method using a direct forcing scheme. The ability of the newly proposed method to simulate subgrid scale particles is then determined and analysed using simulations of one sedimenting particle.

Wang et al. [187] use a source term in the discrete Boltzmann Equation (BE), based on the IBM by Noble and Torczynski [140] respecting the saturation of the cell. The modified Lattice Boltzmann Equation (LBE) reads

$$f_i(\underline{x} + h^2\xi_i, t + h^2) = f_i(\underline{x}, t) - \frac{1}{\tau}(1-\gamma)\left(f_i(\underline{x}, t) - M_i(\underline{x}, t)\right) + \gamma\Omega_i^S \ ,$$

where Ω_i^S is the additional source term, which depends besides the distribution functions f_i also on the averaged velocity of the particles contained in the respective fluid cell. Furthermore the weighting function γ depends on the solid volume fraction of the cell and the relaxation time τ. For the momentum transfer on the particles a "structure-dependent drag based on the energy minimisation multi-scale model is used" [187], which alters Stokes drag (3.7) by multiplying a drag correction factor that depends on the saturation. In the article, the proposed method is used to simulate two dimensional gas-solid fluidised beds.

In the article by Xiong, Madadi-Kandjani and Lorenzini [196] the LBE is formulated as

$$f_i(\underline{x} + h^2\xi_i, t + h^2) = f_i(\underline{x}, t) - \frac{1}{\tau}\left(f_i(\underline{x}, t) - M_i(\underline{x}, t)\right) + F_i(\underline{F}, \underline{x}, t) \ ,$$

where $F_i(\underline{F}, \underline{x}, t)$ is the forcing term described in Equation (2.54). The applied force is the sum of the forces exerted by all solid particles in the vicinity of the node as well as gravitation

$$\underline{F}(\underline{x}, t) = \underline{M}^S(\underline{x}, t) + \alpha_f(\underline{x}, t)\underline{g} \ ,$$

with $\underline{M}^S(\underline{x}, t) = 1/h^3 \sum \underline{F}_k(t)$ and $\alpha_f(\underline{x}, t)$ is the "volume fraction"[196]. The gas–solid drag force $\underline{F}_k(t)$ is again a variation of Stokes drag depending on the volume fraction, but differs from the one used by Wang et al. [187]. The proposed solver is used to simulate a three dimensional gas-solid fluidised bed. In both articles the particle–particle interaction is modelled by variations of the spring-dashpot algorithm (Section 4.1.2).

Finally Zhang et al. [202] compute the hydrodynamic force acting on a discrete particle using the Momentum Exchange Algorithm (MEA) (see Section 4.15) on a fictitious lattice node at the particle position. Therefore the distribution function f_i is interpolated at the particle positions $f_i(\underline{X}_k, t) = L(\underline{X}_k, \underline{x}) f_i(\underline{x}, t)$ using a second order Lagrangian interpolation denoted by $L(\underline{X}_k, \underline{x})$. The force density g_f on particle k is then calculated by

$$g_f(\underline{X}_k, t) = \sum_i \xi_{i^*}\big(f_i(\underline{X}_k, t - h^2) - f_i(\underline{X}_k, t) - 2\rho^F(\underline{X}_k, t - h^2)\frac{\xi_i \underline{U}_k(t - h^2)}{c_s^2}\big) \ ,$$

where ξ_{i^*} is the discrete velocity opposing ξ_i. With it the force on the particle by

$$\underline{F}(t) = -g_f(\underline{X}_k, t)\pi R_k^2 \ .$$

On the other hand the force acting on the fluid is implemented by the same forcing term as in the method introduced by Xiong, Madadi-Kandjani and Lorenzini [196]. The force is interpolated by an approximate δ-function function over the lattice nodes within a distance of $2h$ to the particle. The method is illustrated by simulations of one sedimenting particle and simulations of thousands of sedimenting particles in two and three dimensions.

4.2.1 HLBM *for Subgrid Particulate Flows*

In the following we propose a method to simulate subgrid scale particles using the HLBM as proposed in Section 2.3.5. In HLBM the velocity parameter $\underline{u}^*$ of the Equilibrium Distribution Function (EDF) $M_i[\rho^F, \underline{u}^*](\underline{x}, t)$ is computed by a convex combination of the fluid velocity $\underline{u}^F(\underline{x}, t)$ and a rigid body velocity $\underline{u}^B(\underline{x}, t)$

$$\underline{u}^*(\underline{x}, t) = d(\underline{x}, t)\underline{u}^F(\underline{x}, t) + (1 - d(\underline{x}, t))\underline{u}^B(\underline{x}, t) \qquad \text{in } \Omega_h \times I_h.$$

The porosity parameter $d(\underline{x}, t)$ is computed as the fraction of all volumes $V_s = 4/3\ \pi R_s^3$ of particles $s \in S(\underline{x}, t) := \{k \,|\, \underline{X}_k(t) \in \mathbb{V}(\underline{x})\}$ contained in the vicinity $\mathbb{V}(\underline{x}) := [\underline{x} - h/2, \underline{x} + h/2]^d, \underline{x} \in \Omega_h$, of a lattice node $\underline{x} \in \Omega_h$ and the volume h^3 of the lattice node

$$d(\underline{x}, t) = 1 - \frac{1}{h^3} \sum_{s \in S(\underline{x}, t)} V_s \ .$$

Listing 4.1: HLBM for Subgrid Particulate Flows

```
for t ∈ I_h {
  foreach particle k {
    compute forces F_k(t) = F_k^F(t) + F_k^pp(t) + F_k^g(t)
    U_k(t + h²) = U_k(t) + h² F_k(t)/M
    X_k(t + h²) = X_k(t) + h² U_k(t + h²)
    find closest lattice node x* ∈ Ω_h
    V(x*) += V_k^P
    u^B(x*) += V_k^P U_k(t + h²)
  }
  for x ∈ Ω_h {
    u^B(x) /= V(x)
    d(x, t) = V(x)/h³
    compute u*(x, t) = d(x, t) u^F(x, t) + (1 − d(x, t)) u^B(x, t)
    for i = 0, ..., q − 1 {
      compute M_i[ρ^F, u*](x, t)
      f̃_i(x, t) = −(1/τ) (f_i(x, t) − M_i[ρ^F, u*](x, t))
      f_i(x + h² ξ_i, t + h²) = f̃_i(x, t)
    }
    reset V(x) = 0; u^B(x) = 0
  }
  communicate overlap
}
```

The rigid body velocity is set to be the average over the velocities of the particles in the vicinity $\mathbb{V}(\underline{x})$ of lattice node $\underline{x} \in \Omega_h$, weighted by the particle volume

$$\underline{u}^B(\underline{x}, t) = \frac{1}{\sum_{s \in S(\underline{x},t)} V_s} \sum_{s \in S(\underline{x},t)} V_s \underline{U}_s(t) \, .$$

The complete algorithm is outlined in Listing 4.1.

4.2.2 Application: *Single Particle Sedimentation*

We now determine the ability of the proposed method to simulate subgrid scale particles by simulation of one sedimenting particle. For easy comparison, the simulation setup is identical to the one found in [202]. The particle sediments in a cuboidal domain of length and width 10^{-3} m and height $5 \cdot 10^{-3}$ m, which is discretised with $h = 10^{-5}$ m. The domain is framed by bounceback nodes.

The particle is initially positioned at $\underline{X}(0) = (5 \cdot 10^{-4}\,\text{m}, 5 \cdot 10^{-4}\,\text{m}, 4.9 \cdot 10^{-3}\,\text{m})$ and resting. The simulation is repeated for particles of radius $R = \{h/2, 3h/8, h/4, h/8\}$. The particle timestep has been set to $\delta_t^P = h^2/10$, to overcome stability issues of the forward Euler Method (fEM) as explained above. The simulations ended after 90 seconds. Additional simulation parameters are listed in Table 4.4.

With beginning simulation the particle starts falling. Gravitation, buoyancy (3.5) and Stokes drag (3.7) are acting on the particle

$$M d_t \underline{U}(t) = (\rho^P - \rho^F) V^P \underline{g} + 6\pi\mu R(\underline{u}^F(X(t), t) - \underline{U}(t)) \, . \tag{4.8}$$

	SYMBOL	VALUE
particle density	ρ^F	$1010\,\mathrm{kg\,m^{-3}}$
fluid density	ρ^P	$1000\,\mathrm{kg\,m^{-3}}$
kin. viscosity	ν	$10^{-7}\,\mathrm{m^2\,s^{-1}}$
gravitational acc.	g	$9.8\,\mathrm{m\,s^{-2}}$

Table 4.4: Simulations parameters for simulation of the sedimentation of one subgrid scale particle.

τ	Δ_t [s]	$\underline{U}/\widehat{\underline{U}}$
0.71	0.0007	4.67
0.8	0.0001	1.58
0.95	0.0015	1.24
1.1	0.002	1.14
1.25	0.0025	1.11

$2R/h$	$\widehat{\underline{U}}$ [ms^{-1}]	$\underline{U}/\widehat{\underline{U}}$
1	$5.44 \cdot 10^{-5}$	4.67
3/4	$3.06 \cdot 10^{-5}$	1.29
1/2	$1.36 \cdot 10^{-5}$	1.06
1/4	$3.4 \cdot 10^{-6}$	1.007

Table 4.5: Errors for one sedimenting particle. The left table lists results for several values of the relaxation time τ and fixed particle radius $R = h/2$. The right table lists results for varying particle radii and fixed $\tau = 0.71$.

Assuming that $\underline{u}^F(X(t), t) = 0$ for all times and $U(0) = 0$, the particle velocity $\underline{U}(t)$ can be determined analytically to

$$\underline{U}(t) = \frac{2}{9}\frac{R^2}{\mu}(\rho^P - \rho^F)\underline{g}\left(1 - e^{-\frac{9}{2}\frac{\mu}{R^2}t}\right) . \tag{4.9}$$

In the limit of $t \to \infty$, when the gravitational force equals Stokes drag, the sedimentation velocity $\widehat{\underline{U}}$ is obtained to

$$\widehat{\underline{U}} = \frac{2}{9}\frac{R^2}{\mu}(\rho^P - \rho^F)\underline{g} .$$

Figure 4.9 illustrates the computed results. Shown is the particle velocity normalised by the analytical sedimentation velocity $\underline{U}(t)/\widehat{\underline{U}}$ against the time for several particle radii and fixed relaxation time $\tau = 0.71$. One can see that the deviation of the simulated sedimentation velocity from the analytically determined velocity decreases with decreasing particle size. For $2R = h$ the greatest deviations of 3 to 3.5 times the analytic velocity are found. However, for $R = h/4$ the deviation reduces to about 6%, for $R = h/8$ even to less then 1%. The maximal error over the entire simulation time can also be found in Table 4.5.

One can also see, that for the largest simulated particle ($2R = h$), strong oscillations occur. These oscillations can be connected to the particles position relative to the lattice nodes. As the backcoupling is added to the closest lattice point its influence abruptly changes every time the particle moves into the vicinity of another lattice node, leading to discontinuities in the force development. Additionally, the backcoupling is temporarily applied to nodes in course of the particle, which may lead to an unwanted particle acceleration, as Stokes drag is computed using the velocity of the surrounding lattice nodes. This may be overcome by a velocity interpolation respecting higher moments of the backcoupling force, e. g. the one explained in Sierou [163]. In fact, a first simulation using the δ-function interpolation introduced in

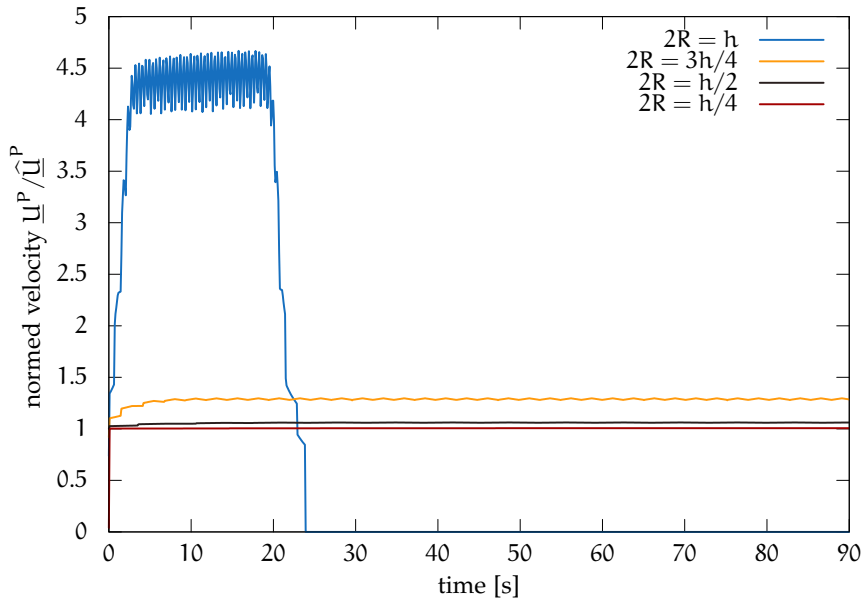


Figure 4.9: Results for one sedimenting subgrid scale particle. Shown is the particle velocity normalised by the analytical sedimentation velocity against the time for several particle radii and fixed $\tau = 0.71$.

Section 4.3 shows a reduction of the error in the case $2R = h$ and $\tau = 0.71$ from $\max_t \underline{U}(t)/\hat{\underline{U}} = 4.67$ to 1.31.

Figure 4.10 shows the obtained results for a particle of fixed radius $R = h/2$. Shown is again the normalised particle velocity $\hat{\underline{U}}$ against the time for several values of the relaxation time τ. The graphs for all values of τ coincide in their general course. After a short acceleration, the normalised sedimentation velocity fluctuates around a certain value until the particle reaches the cavity floor, decelerates and finally comes to rest.

It can be seen that with increasing τ also the accuracy of the simulated sedimentation velocity increases. As before, for $\tau = 0.71$ the same deviation of 3 to 3.5 times the analytic velocity is found, but for $\tau = 1.25$ the deviation reduces to 8–10%. However, with increasing τ the flow simulation becomes non-physical, as can be seen in Figure 4.11. Especially comparing Figure 4.11a where $\tau = 0.71$ and Figure 4.11b where $\tau = 1.25$. Both figures show the z-component of the fluid velocity in a plane with normal in y-direction through the center of the domain. In Figure 4.11a one can see that the z-component of the flow smoothly tends towards zero with increasing distance to the particle, while Figure 4.11b shows strong spatial fluctuations. A similar effect can be seen in Figure 4.11c and 4.11d, displaying the fluid velocity magnitude. Besides that the maximal fluid velocity in Figure 4.11c is $2.05 \cdot 10^{-4}\,\mathrm{m\,s^{-1}}$ and thus greater than the expected particle velocity therefore indicating an overestimation of the backcoupling force.

Therefore, a compromise between the accuracy of the simulated particle velocity and the accuracy of the simulated fluid characteristics must be found. We conclude, that within strong limitations, the HLBM may be used to simulate two way coupled flow of subgrid scale particles, but further investigation is necessary.

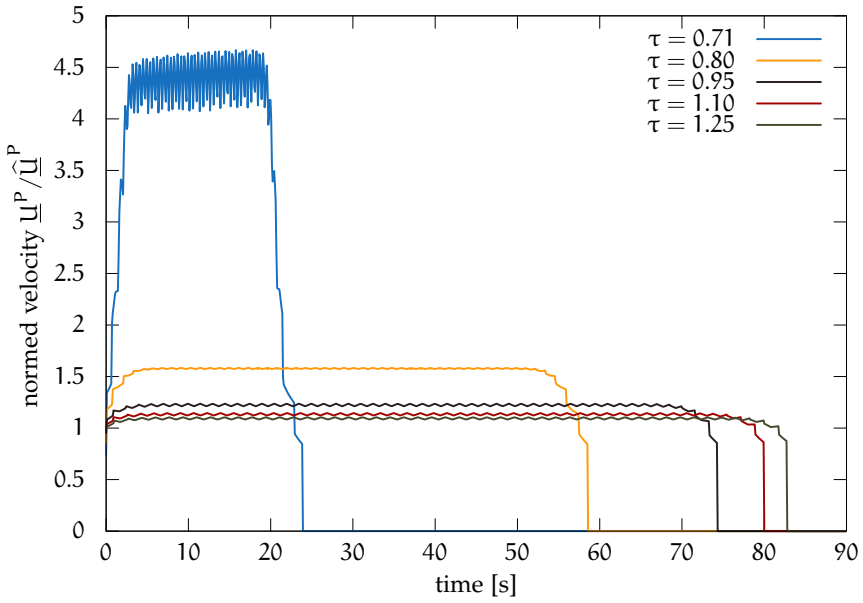
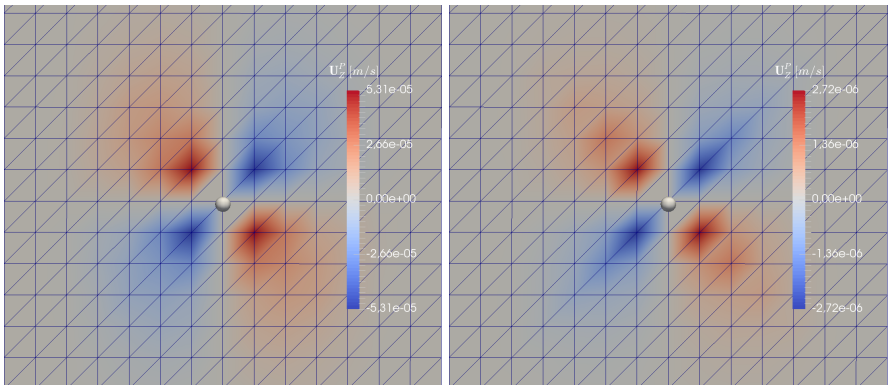
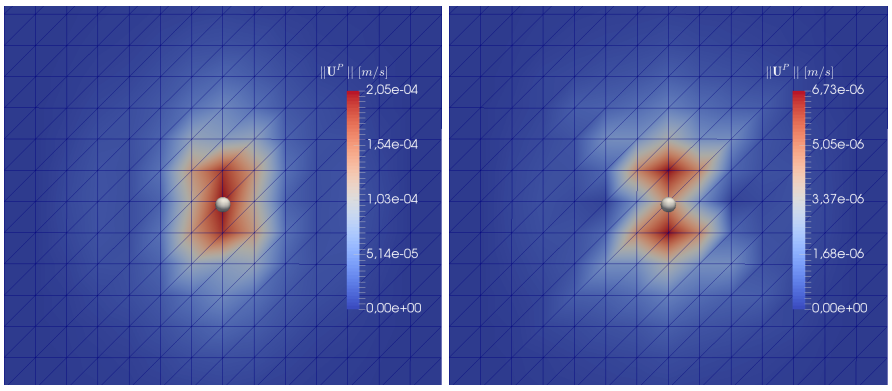


Figure 4.10: Results for one sedimenting subgrid scale particle of radius $2R = h$. Shown is the particle velocity normalised by the analytical sedimentation velocity against the time for several values of the relaxation time τ.



(a) $\tau = 0.71$ (b) $\tau = 1.25$



(c) $\tau = 0.71$ (d) $\tau = 1.25$

Figure 4.11: The upper images show the z-component of the velocity, the lower images the velocity magnitude. The left images are results for $\tau = 0.71$, the right images for $\tau = 1.25$. The particle radius is $2R = h$.

Listing 4.2: LBM for direct forcing subgrid scale particles.

```
for t ∈ I_h {
  for x ∈ Ω_h {
    ρ^F_h u*_h = Σ_i ξ_i f_i + (h²/2) F^St(x, t)
    for i = 0, ..., q − 1 {
      F_i = (1 − 1/(2τ)) w_i (h(ξ_i − u*_h) + h²(ξ_i · u*_h ξ_i)) · F_i
      f̃_i(x, t) = f_i(x, t) − (1/τ)(f_i(x, t) − M[ρ^F_h, u*_h](x, t)) + F_i
      f_i(x + h²ξ_i, t + h²) = f̃_i(x, t)
    }
  }
  communicate overlap

  foreach particle k {
    interpolate u^F(X(t), t) = δ_h(X(t))
    compute F^St(X(t), t + h²) = 6πμR_k(u^F(X(t), t) − U(t))
    interpolate −F^St(X(t), t + h²) on surrounding lattice nodes
    F(t + h²) = F^St(X(t), t + h²) + M(ρ^P − ρ^F)g
    U_k(t + h²) = U_k(t) + h²F(t + h²)/M
    X_k(t + h²) = X_k(t) + h²U(t + h²)
    compute particle boundaries
  }
  communicate particles
}
```

4.2.3 *Direct Forcing Scheme*

In the following we propose a second approach for subgrid particulate flow. Fluid flow is simulated using a D3Q19 BGK–LBM. Lagrangian particle trajectories are computed by a backward Euler scheme. Rotation is once again neglected. The idea of the new scheme is to apply Newton's third law and directly introduce negative Stokes drag (Eq. 3.7) on the fluid. Body force is induced using the forcing scheme by Guo, Zheng and Shi [71] introduced in Section 2.3.3. The force originating from one Lagrangian particle is interpolated onto the Eulerian lattice using the approximate δ-functions stated in [146]

$$\delta_h(x_i) = \begin{cases} \frac{1}{4}(1 + \cos(\frac{\pi|x_i - X_i|}{h}) , & |x_i - X_i| \leqslant 2 , \\ 0 , & |x_i - X_i| > 2 , \end{cases}$$

and further explained in Section 4.3. The fluid velocity contributing to Stokes drag is interpolated in the same manner. The algorithm is summarised in Listing 4.2.

- Compute Stokes drag (3.7) and gravitational force (3.5).
- Compute particle-particle interaction, using a k-d tree (Section 4.1.1) and the spring-dashpot interaction model (Section 4.1.2).
- For each particle apply negative Stokes drag to the fluid.
- Execute particle step.
- Execute collide and stream steps.

τ	Δ_t [s]	$\underline{u}/\widehat{\underline{u}}$
0.52	$7 \cdot 10^{-5}$	1.17
0.65	$5 \cdot 10^{-4}$	1.17
0.71	$7 \cdot 10^{-4}$	1.18
0.8	$1 \cdot 10^{-3}$	1.18
0.95	$1.5 \cdot 10^{-3}$	1.20
1.1	$2 \cdot 10^{-3}$	1.21
1.25	$2.5 \cdot 10^{-3}$	1.24

$2R/h$	$\widehat{\underline{u}}$ [ms^{-1}]	$\underline{u}/\widehat{\underline{u}}$
1	$5.44 \cdot 10^{-5}$	1.18
3/4	$3.06 \cdot 10^{-5}$	1.13
1/2	$1.36 \cdot 10^{-5}$	1.09
1/4	$3.4 \cdot 10^{-6}$	1.17

Table 4.6: Errors for one sedimenting particle and direct forcing approach. The left table lists results for several values of the relaxation time τ and fixed particle radius $R = h/2$. The right table lists results for varying particle radii and fixed τ.

	SYMBOL	VALUE
particle density	ρ^F	$1010\,\mathrm{kg\,m^{-3}}$
fluid density	ρ^P	$1000\,\mathrm{kg\,m^{-3}}$
kin. viscosity	ν	$10^{-7}\,\mathrm{m^2\,s^{-1}}$
gravitational acc.	g	$9.8\,\mathrm{m\,s^{-2}}$
norm. spring const.	K^n	$5\,10^6\,\mathrm{kg\,s^{-2}}$
norm. damping const.	η^n	$50\,\mathrm{kg\,s^{-1}}$
tang. spring const.	K^t	$80\,\mathrm{kg\,s^{-2}}$
tang. damping const.	η^t	$50\,\mathrm{kg\,s^{-1}}$
friction const.	μ^t	0.5

Table 4.7: Simulations parameters for simulation of the sedimentation of 8125 subgrid scale particles.

4.2.4 Application: *Single particle sedimentation*

We simulate one sedimenting particle. The simulation setup is identical to the one described in 4.2.2. The obtained value are listed in Table 4.6. Again the relaxation time τ has been varied for values from 0.52 to 1.25. This time only a weak dependence of the normalised sedimentation velocity was found. Opposing the *HLBM for subgrid scale particles* algorithm the accuracy increases with decreasing τ. For variation of the particle radius no explicit conclusion can be drawn. While the values for $2R = h$, $2R = 3h/4$ and $2R = h/2$ indicate that the accuracy increases with decreasing radius, this assumption is disproven by the result for $2R = h/4$.

4.2.5 Application: 8125 *sedimenting particles*

We now simulate the sedimentation process of 8125 particles. We therefore consider a cubical cavity with side length 0.0015 m. The cavity is discretised by a lattice with spacing $h = 0.0001$ m. The relaxation time is set to $\tau = 0.71$ leading to a time step of 0.0007 s. A bounceback boundary condition is applied to all cavity boundaries. Exactly 8125 particles of radius $R = h/4 = 2.5 \cdot 10^{-4}$ m are positioned in 13 layers of 25×25 particles. The centre particle of the top layer is positioned at $(4.3 \cdot 10^{-5}\,\mathrm{m}, 7.5 \cdot 10^{-4}\,\mathrm{m}, 7.5 \cdot 10^{-4}\,\mathrm{m})$.

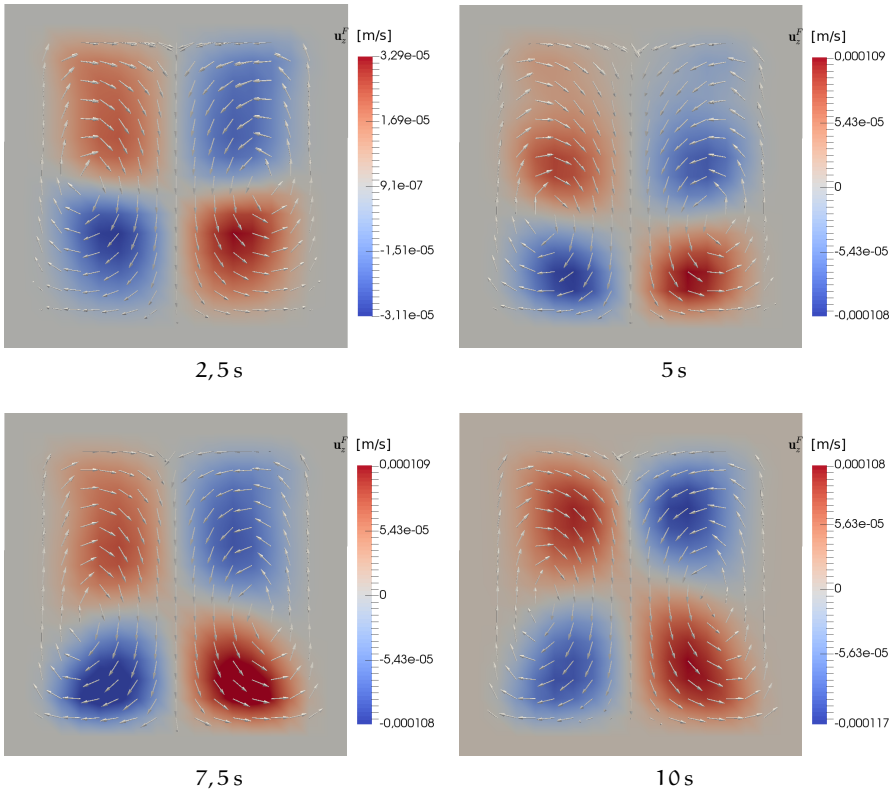


Figure 4.12: z-Component of fluid velocity in a plane through the center and normal in y-direction.

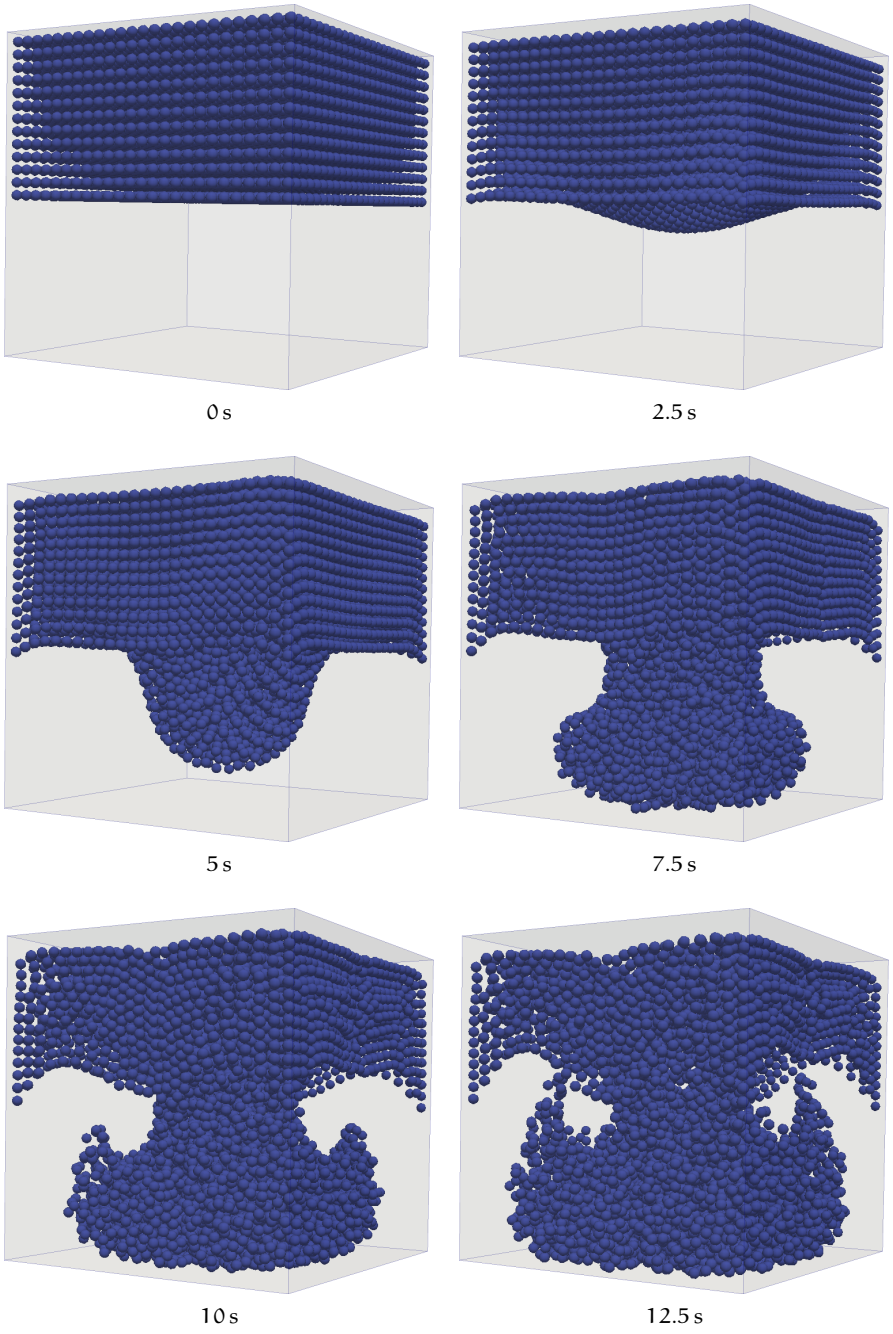


Figure 4.13: Sedimentation of 8125 particles at different timesteps.

The remaining particles are aligned around it with a distance of $0.95 \cdot 10^{-5}$ m in x-direction and $0.95833 \cdot 10^{-5}$ m in y- and z-direction.

Gravitation, bounce and Stokes drag are applied to the particles. Additionally the spring-dashpot collision model from Section 4.1.2 is used. The used parameters are listed in Table 4.7. Particle contact is computed using the k-d tree algorithm (Section 4.1.1).

The particle distribution is shown at several time steps in Figure 4.13. The initial distribution can be seen in 4.13a. During the first seconds of the simulation the particle conglomeration begins sedimentation as a whole, which can be seen at the increasing distance of the top particle layer to the upper cavity boundary.

However the particles at the center of the cavity accelerate faster than the particles close to the boundaries and at 2.5 s a bulge can be seen. The reason is that the particle front accelerates the fluid, so that the subsequent particles experience a smaller resistance and therefore accelerate faster. In addition, the fluid is slowed down at the cavity boundaries due to the no-slip boundary condition of the fluid. This, in turn prevents rapid sedimentation of the boundary particles.

After approximately 7.5 s first contact between the particles and the cavity floor occurs. Due to the fluid reflux in the area closer to the boundary some particles are dragged along and move upwards. The fluid velocity is illustrated in Figure 4.12, showing the z-component of the velocity in a plane through the cavity center with normal in y-direction. The fluid flow is additionally indicated by arrows. One can clearly see the vortexes formed by the sedimenting particles. The fluid flow downwards in the cavity's centre and upwards close to the boundaries. We found a good qualitative match of our results in comparison to the results obtained by Zhang et al. [202].

SUMMARY

We introduced two new methods for simulating flow of subgrid scaled particles. The first approach employs the HLBM introduced in Section 2.3.5. Investigation of one sedimenting particle shows a strong τ-dependency. It is found that by reducing the particle radius the deviation of the sedimentation velocity compared to an analytic solution can be reduced to less then 1%. The second approach employs the forcing scheme introduced in Section 2.3.3. Again sedimentation of one particle is simulated. The τ-dependency of the previous scheme is not regained. Sedimentation of 8125 interacting particles is executed. The simulation shows expected results, and matches the results obtained by [202].

4.3 RESOLVED PARTICLES

We now turn towards the simulation of submerged particles with diameter of similar scale as the geometry, in the sense that one particle covers several lattice nodes. The pioneer work in simulating this type of particles using LBM has been done by Ladd [113, 114]. He used a simple bounceback rule to model the particle boundary. Thereby the boundary nodes have been centred on the links between the interior and exterior regions leading to a staircase approximation. The interior fluid was kept for computational convenience, and nodes inside and outside are treated in an identical manner. This means, while the particle is moving over the grid, lattice nodes spontaneous flip and turn from a fluid node into a solid boundary and vice versa.

This "causes fluctuation on the computation of forces on the particle and further leads to fluctuations on the velocities of the particle"[138].

Two more advanced methods for numerical simulation of submerged particles are the IBM and the FDM, which are introduced in the following.

IMMERSED BOUNDARY METHOD

The IBM has been proposed by Peskin [144, 145] in the 1970s with the aim to simulate blood flow in the human heart. Blood flow is regulated by the heart valve, more precisely by a valvular leaflet which is modelled as an immersed boundary. In a later work Peskin [146] concentrates on the mathematical structure as well as spatial and temporal discretisation of the method. For simplicity we assume that there is only one particle. The main idea of the IBM is to model the rigid structure's boundary as deformable, but with high stiffness. Therefore a Lagrangian curvilinear coordinate system is introduced and attached to the domain Ω^P of a particle, such that $\underline{X}(\underline{q},t) : \Omega^P \times I \to \Omega$ is the position of the Lagrangian point $\underline{q}$ at time t in the Eulerian coordinate system of the fluid. The governing equations listed by Peskin are the incompressible viscous NSE including the forcing term $\underline{f}$ in the entire domain Ω, instead of Equations (4.2), which are stated for the fluid domain Ω^F only. They are combined with

$$\underline{f}(\underline{x},t) = \rho^F \int_{\underline{s}\in\partial\Omega^P} \underline{F}(\underline{q},t)\underline{\delta}(\underline{x}-\underline{X}(\underline{q},t))\,d\underline{s} \qquad \text{on } \partial\Omega^P\,, \tag{4.10}$$

$$\partial_t \underline{X}(\underline{q},t) = \int_\Omega \underline{u}^F(\underline{x},t)\underline{\delta}(\underline{x}-\underline{X}(\underline{q},t))\,d\underline{x} \qquad \text{on } \partial\Omega^P\,. \tag{4.11}$$

The force $\underline{F} : \partial\Omega^P \times I \to\in \mathbb{R}^3$ in the first equation describes the elastic properties of the boundary. The second equation denotes a *no-slip condition*. The transition between the Eulerian and the Lagrangian coordinate system is performed by means of a three dimensional *delta distribution* $\underline{\delta}(\underline{x})$.

The IBM has been introduced to LBM by Feng and Michaelides [57]. Feng and Michaelides discretise the fluid domain into a lattice and the particle boundary into a set of Lagrangian boundary points $\underline{q}_j, j = 1,\dots,J$ that move under the action of the moving fluid. The force $\underline{F}(\underline{q},t)$ is identified as a restoration force modelled by a linear spring term with spring constant $k > 0$,

$$\underline{F}(\underline{q}_j) = -k\Delta_{\underline{q}_j}\,,$$

where $\Delta_{\underline{q}_j}$ denotes a perturbation of the Lagrangian node $\underline{q}_i$ due to the deformation of the boundary by the fluid. This approach is called *penalty method*.

The discrete version of Equation (4.10) is then denoted by

$$\underline{f}(\underline{x},t) = \sum_j \underline{F}(\underline{q}_j)\delta_h(\underline{x}-\underline{X}(\underline{q}_j,t)) \qquad \text{in } \Omega_h \times I_h\,,$$

where the delta distribution is approximated by a continuous kernel distribution function $\underline{\delta_h}(\underline{x}) = \delta_h(x_1)\delta_h(x_2)\delta_h(x_3)$ and

$$\delta_h(x_i) = \begin{cases} \frac{1}{4}(1+\cos(\frac{\pi|x_i-X_i|}{h}))\,, & |x_i - X_i| \leqslant 2\,, \\ 0\,, & |x_i - X_i| > 2\,, \end{cases} \tag{4.12}$$

for a kernel of four lattice nodes. This approximate delta function is subject to certain conditions stipulated by Peskin [146]. However it is possible to

reduce the support to three nodes [154], resulting in a reduction from 64 to 27 participating nodes in three dimensions. The forcing term $\underline{f}$ is then added to the LBE.

The motion of the rigid body inside the boundary is not significant and therefore not explicitly enforced [57].

FICTITIOUS DOMAIN METHOD

As a second common approach to simulate numerically resolved particles we introduce the FDM. Sometimes Glowinski, Pan and Periaux [67] are said to have proposed the FDM [107], however they write it was presented before by Hyman [93].

The basic idea of FDM is to extend a geometric complex domain to a larger, but geometric simpler, possibly time independent domain. However the original boundary conditions must remain, such that the new solution matches the solution of the original problem. For the initially stated problem the fluid domain without the particle domains $\Omega^F = \Omega \setminus \bigcup_k \Omega_k^P$ represents the complex domain and the entire domain Ω can be interpreted as the extended, simple domain. By this approach, the interior of the particle domain is filled with the fluid. However a body force is employed, forcing the fluid to behave as a rigid body. This "allows to update particle velocities explicitly without the need of computing hydrodynamic force and torque on the particles"[138].

A direct forcing FDM has been combined with a two dimensional LBM by Nie and Lin [138] and extended to three dimensions in Nie and Lin [137]. Following the first article for a short introduction, we apply the Momentum Equation (4.2a) to the entire domain Ω and set the fluid velocity inside of a particle to the local particle velocity

$$\rho^F(\underline{x},t)\,d_t\underline{u}^F(\underline{x},t) = \nabla\cdot\underline{\underline{\sigma}}(\underline{u}^F(\underline{x},t),p(\underline{x},t)) + \underline{\lambda}(\underline{x},t) \qquad \text{in } \Omega\times I\,, \tag{4.13}$$

$$\underline{u}^F(\underline{x},t) = \underline{U}(t) + \underline{\omega}(t)\times(\underline{x}-\underline{X}(t)) \qquad \text{in } \Omega^P(t)\times I\,. \tag{4.14}$$

In this notation $\underline{\lambda}$ denotes the pseudo body force originating from the particle. We continue to assume that only one particle exits. Neglecting additional forces and utilising Gauss' theorem we insert Equation (4.13) into Equation (4.4) and obtain

$$\underline{F}_k = \int_{\Omega^P} \rho^F d_t\underline{u}^F - \underline{\lambda}\,d\underline{x} \tag{4.15}$$

$$= -\int_{\Omega^P}\underline{\lambda}\,d\underline{x} + \int_{\Omega^P}\rho(\underline{U}+\underline{\omega}\times(\underline{x}-\underline{X}))\,d\underline{x} \tag{4.16}$$

$$= -\int_{\Omega^P}\underline{\lambda}\,d\underline{x} + M^F d_t\underline{U}\,. \tag{4.17}$$

Here $M^F = \int_{\Omega^P}\rho^F\,d\underline{x}$ denotes the fluid mass within the particle domain. Proceeding in a similar way with Equation (4.5) for the torque one obtains

$$\underline{T} = -\int_{\Omega^P}(\underline{x}-\underline{X})\times\underline{\lambda}\,d\underline{x} + d_t(\underline{\underline{I}}^F\underline{\omega}_k)\,, \tag{4.18}$$

with $\underline{\underline{I}}^F = \int_{\Omega^P}\rho^F(\underline{x}-\underline{X})\times(\underline{x}-\underline{X})\,d\underline{x}$. Both quantities M^F and $\underline{\underline{I}}^F$ have to be computed explicitly during the simulation as ρ^F is not constant during LBM simulations.

Substituting Equation (4.15) into Equation (4.3a) and Equation (4.18) into Equation (4.3c) one obtains

$$(M - M^F)\,d_t \underline{U} = \int_{\Omega^P} \underline{\lambda}\; d\underline{x} \tag{4.19}$$

$$d_t((\underline{\underline{I}} - \underline{\underline{I}}^F)\omega) = -\int_{\Omega^P} (\underline{x} - \underline{X}) \times \underline{\lambda}\; d\underline{x}\,. \tag{4.20}$$

Applying a fractional step time scheme to Equation (4.13) we begin to decouple the system of Equations (4.13), (4.14), (4.19), and (4.20). We denote quantities at the current time step by the index n and at the following by $n+1$

$$\underline{\lambda}^{n+1} = \rho^f \frac{\underline{u}^{F^{n+1}} - \underline{u}^{F^n}}{\Delta_t} + \rho^F \underline{u}^{F^n} \cdot \nabla \underline{u}^{F^n} - \nabla \cdot \underline{\underline{\sigma}}^n\,. \tag{4.21}$$

Introducing a temporary fluid velocity $\underline{u}^*$, that satisfies the homogeneous momentum equation

$$\rho^F \frac{\underline{u}^* - \underline{u}^{F^n}}{\Delta_t} + \rho^F \underline{u}^{F^n} \cdot \nabla \underline{u}^{F^n} - \nabla \cdot \underline{\underline{\sigma}}^n = 0\,, \tag{4.22}$$

we find that

$$\underline{\lambda}^{n+1} = \rho^f \frac{\underline{u}^{F^{n+1}} - \underline{u}^*}{\Delta_t} + \rho^f \frac{\underline{u}^* - \underline{u}^{F^n}}{\Delta_t} + \rho^F \underline{u}^{F^n} \cdot \nabla \underline{u}^{F^n} - \nabla \cdot \underline{\underline{\sigma}}^n \tag{4.23}$$

$$= \rho^f \frac{\underline{u}^{F^{n+1}} - \underline{u}^*}{\Delta_t}\,. \tag{4.24}$$

Plugging Equation (4.14) into the previous one we get

$$\underline{\lambda}^{n+1} = \rho^f \frac{\underline{U}^{n+1} + \underline{\omega} \times (\underline{x} - \underline{X})}{\Delta_t} - \rho^f \frac{\underline{u}^*}{\Delta_t}\,. \tag{4.25}$$

Introducing Equation (4.24) into Equations (4.19) and (4.20) leads to

$$M \frac{U^{n+1}}{\Delta_t} = (M - M^F)\frac{U^n}{\Delta_t} + \int_{\Omega^P} \rho^F \frac{\underline{u}^*}{\Delta_t}\; d\underline{x}\,, \tag{4.26}$$

$$\frac{\underline{\underline{I}}\omega^{n+1}}{\Delta_t} = \frac{(\underline{\underline{I}} - \underline{\underline{I}}^F)\omega^n}{\Delta_t} + \int_{\Omega^P} (\underline{x} - \underline{X}) \times \rho^F \frac{\underline{u}^*}{w_i \Delta_t}\; d\underline{x} \tag{4.27}$$

For the computation the whole particle domain is discretised by a Lagrangian grid and the interpolation between the two grid is again obtained by δ-functions. The algorithm can be summarised as follows

- Compute an intermediate distribution function f_i^* by the LBM collision step disregarding the body force λ in the entire domain Ω.
- Compute the temporal velocity $\rho \underline{u}^* = \sum_i \underline{\xi}_i f_i^*$.
- Update the particle velocity $\underline{U}^{n+1}$ and angular velocity $\underline{\omega}^{n+1}$ from Equations (4.26) and (4.27).
- Update the body force $\underline{\lambda}^{n+1}$ in the particle domain Ω^P using Equation (4.25)
- Compute $\widetilde{f}_i = f_i^* + \frac{\Delta_t}{c_s^2} \underline{\lambda} \underline{\xi}_i$ in Ω^P and execute the LBM streaming step in Ω.
- Recompute the fluid density and velocity.

SUMMARY
We introduced the IBM and the FDM both applied to LBM as two examples of algorithms to simulate numerically resolved particles in fluid. While the IBM only acts on the particle boundary, the FDM appears as body force on the fluid in the entire domain of a particle. Both methods occupy a Lagrangian mesh that moves with the particle next to the Eulerian lattice of the LBM to approximate the particle boundaries and to handle the fluctuations in Ladd's method [113, 114]. Initial mesh generation as well as recalculation during each time step can be cost intensive. In the following section we will overcome this issue by using the Eulerian lattice to compute the particle properties.

4.3.1 *HLBM for Resolved Particulate Flows*

This section is based on article [81], submitted to *Particuology*.

We employ the HLBM that has been proposed in Section 2.3.5 and newly introduce a method to simulate submersed particles. The method can be categorised as direct forcing method and may be seen as a variant of the FDM. As in the direct forcing FDM we flood the entire domain Ω including the particle domain with fluid and force the fluid in the particle domain to behave like a rigid body. However we do not use a Lagrangian mesh to discretise the particle domain Ω^P, but use the Eulerian mesh instead.

Using HLBM the entire domain is modelled as moving porous medium. The interior of a particle is assumed to have vanishing porosity, meaning that the solid medium is completely displacing the fluid. The porosity is increased over a thin boundary layer becoming a pure fluid. For a lattice node this leads to a smooth transition from a fluid to a solid node, as the particle skims over it and prevents the shock waves occurring in Ladd's approach [113, 114]. Additionally the smooth transition resolves the question of how to initialise a new fluid node released as the particle moves on. The thickness of the boundary layer is chosen to scale with the spatial stepping, such that in the limit of vanishing step size, one obtains a sharp boundary.

On each lattice node $\underline{x} \in \Omega_h$ the design parameter $d(\underline{x}, t)$ is computed by

$$d(\underline{x}, t) := 1 - \prod_{k=0}^{N} (1 - d_k(\underline{x}, t)) \,,$$

where the porosity parameter d_k is modelled depending on the particle type. Nodes with $d = 1$ can be seen as pure fluid, while pure solid nodes are assigned $d = 0$.

For spherical particles (Figure 4.14) with centre of mass $\underline{X}_k(t) = (X_1^k, X_2^k)(t)$ and radius $R_k > 0$, $R_k^+ := R_k + \epsilon_h/2$, $R_k^- := R_k - \epsilon_h/2$ the porosity parameter d_k^S is computed by

$$d_k^S(\underline{x}, t) = \begin{cases} 0, & \|\underline{x} - \underline{X}_k(t)\|_2 \leqslant R_k^- \\ \sin^2\left(\frac{\pi}{2\epsilon_h}(\|\underline{x} - \underline{X}_k(t)\|_2 - R_k^-)\right), & R_k^- < \|\underline{x} - \underline{X}_k(t)\|_2 \leqslant R_k^+ \\ 1, & \|\underline{x} - \underline{X}_k(t)\|_2 > R_k^+ \end{cases} . \tag{4.28}$$

For cuboidal particles (see Figure 4.15a) with half edge length L_k, $L_k^+ = L_k + \epsilon_h/2$, $L_k^- = L_k - \epsilon_h/2$ and $\alpha_j = \left|x_j - X_j\right|, j \in 1, 2$, the porosity parameter d_k^C is computed by

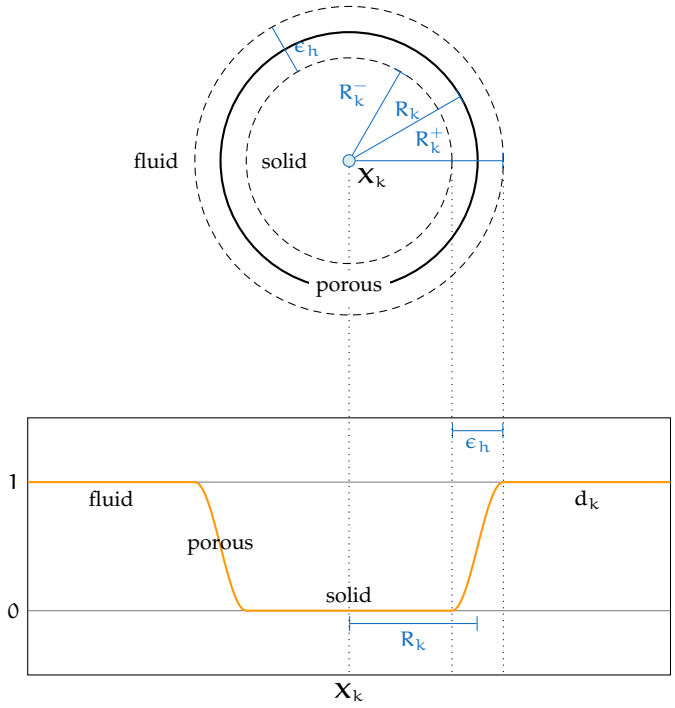


Figure 4.14: Setup for the smooth transition of the porosity d_k for a sphere of radius R_k and centre $\underline{x}_0$ in a fluid of the new fluid structure model. Inside the particle geometry $d_k = 0$, outside $d_k = 1$ and in an ϵ_h wide area around the surface $0 < d_k < 1$, which enables smooth and robust movements.

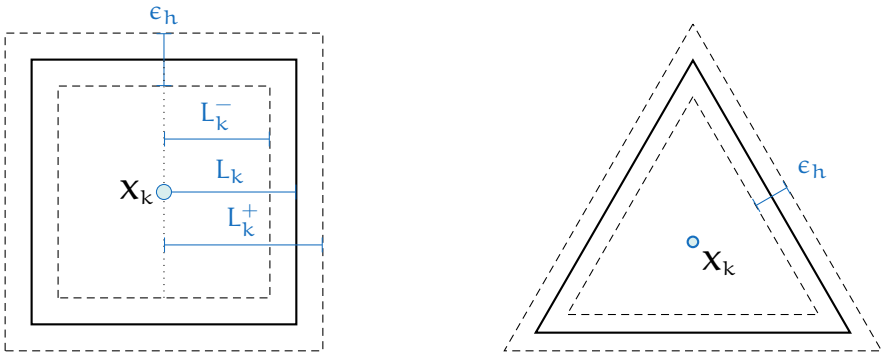
$$d_k^C = \begin{cases} 0, & \alpha_1 < L_k^- \text{ and } \alpha_2 < L_k^- \\ \sin^2\left(\frac{\pi}{2\epsilon_h}\left(\alpha_2 - L_k^-\right)\right), & \alpha_1 < L_k^- \text{ and } L_k^- \leqslant \alpha_2 \leqslant L_k^+ \\ \sin^2\left(\frac{\pi}{2\epsilon_h}\left(\alpha_1 - L_k^-\right)\right), & L_k^- \leqslant \alpha_1 \leqslant L_k^+ \text{ and } \alpha_2 < L_k^- \\ \sin^2\left(\frac{\pi}{2\epsilon_h}\left(\alpha_1 - L_k^+\right)\right) \cdot & \\ \quad \cdot \sin^2\left(\frac{\pi}{2\epsilon_h}\left(\alpha_2 - L_k^+\right)\right), & L_k^- \leqslant \alpha_1 \leqslant L_k^+ \text{ and } L_k^- \leqslant \alpha_2 \leqslant L_k^+ \\ 1, & \text{else} \end{cases}, \tag{4.29}$$

Finally, the porosity parameter d_k^T of a smooth equilateral triangle can be computed by

$$d_k^T = \prod_{l=1}^{3} (1 - \gamma_l(\underline{x})) \,, \tag{4.30}$$

where

$$\gamma_l(\underline{x}) = \begin{cases} 1, & a_l < -\epsilon_h/2, \\ \sin^2\left(\frac{\pi}{2\epsilon_h}(a_l + \frac{\epsilon_h}{2})\right), & -\epsilon_h/2 \leqslant a_l \leqslant \epsilon_h/2 \\ 0, & \epsilon_h/2 < a_l \end{cases},$$



(a) Smooth square (b) Smooth triangle

Figure 4.15: Smooth particles

and a_l is the oriented distance of $\underline{x}$ to the respective sides of the triangle.

Here, the smoothing parameter $\epsilon_h \geqslant 0$ is of the same scale as h, i.e. $\epsilon_h = \epsilon h$, with $\epsilon \geqslant 0$ and describes the width of the transition zone of the parameter d. Hence one obtains a sharp boundary as h tends to 0. The particles' influence on the fluid node at $(\underline{x}, t) \in \Omega_h \times I_h$ is expressed by the velocity parameter $\underline{u}^*$ of the EDF $M_i[\rho^F, \underline{u}^*]$, which is computed according to Equation (2.56)

$$\underline{u}^*(\underline{x}, t) = d(\underline{x}, t)\ \underline{u}^F(\underline{x}, t) + (1 - d(\underline{x}, t))\ \underline{u}^B(\underline{x}, t)\ .$$

The velocity $\underline{u}^B(\underline{x}, t)$ is defined as the weighted average of the particle velocities

$$\underline{u}^B(\underline{x}, t) := \frac{\sum_k d_k(\underline{x}, t)\ \underline{u}_k^P(\underline{x}, t)}{\sum_k d_k(\underline{x}, t)} \tag{4.31}$$

where $\underline{u}_k^P(\underline{x}, t) = \underline{U}_k(t) + \underline{\omega}_k(t) \times (\underline{x} - \underline{X}_k(t))$ is the sum of the translational velocity of particle's centre of mass $\underline{U}_k : I_h \to \mathbb{R}^2$ and the angular velocity $\underline{\omega}_k : I_h \to \mathbb{R}$ of the particle.

The hydrodynamic force $\underline{F}_k^F$ acting on particle k is based on the MEA, first proposed by Ladd [113, 114] to compute the momentum acting on solid bounceback boundaries. In Caiazzo and Junk [25] the MEA is introduced as

$$\begin{aligned} \underline{g}_i(\underline{x}_j, t) &= \underline{\xi}_i \widetilde{f}_i(\underline{x}_j, t) - \underline{\xi}_{i^*} f_{i^*}(\underline{x}_j, t + h^2) \\ &= \underline{\xi}_i \left(f_i(\underline{x}_j + \underline{\xi}_i h^2, t + h^2) + f_{i^*}(\underline{x}_j, t + h^2) \right)\ , \end{aligned} \tag{4.32}$$

where $\underline{\xi}_{i^*} = -\underline{\xi}_i$ and f_{i^*} is the momentum distribution in opposing direction of f_i and $\widetilde{f}_i$ the distribution after collision (see Equation (2.38)). Therefore $\underline{g}_i(\underline{x}_j, t)$ denotes the momentum change in direction i. Using the formulation in the second line of the previous equation in the implementation, one does not need to store the value of $\widetilde{f}_i(\underline{x}_j, t)$, but can take it from the

neighbouring node in direction of $\underline{\xi}_i$. Summation over all directions results in the total momentum change during one timestep

$$\begin{aligned}\underline{G}(\underline{x}_j, t) &= \sum_i \underline{\xi}_i \left(f_i(\underline{x}_j + \underline{\xi}_i h^2, t) + f_{i^*}(\underline{x}_j, t) \right) \\ &= \sum_i \underline{\xi}_i f_i(\underline{x}_j + \underline{\xi}_i h^2, t) - \sum_i \underline{\xi}_{i^*} f_{i^*}(\underline{x}_j, t) \\ &= \sum_i \underline{\xi}_i \left(f_i(\underline{x}_j, t - h^2) + Q(f_i(\underline{x}_j, t)) \right) - \sum_i \underline{\xi}_i f_i(\underline{x}_j, t) \\ &= \rho^F \underline{u}_h^F(\underline{x}_j, t - h^2) - \rho^F \underline{u}_h^F(\underline{x}_j, t) + \sum_i \underline{\xi}_i Q(f_i(\underline{x}_j, t)) \, .\end{aligned}$$

In the conventional LBM the first moment of the BGK collision operator $Q(F)$ is momentum conserving. However in the HLBM, momentum is transferred between the solid and the fluid phase during the collision step according to Equation (2.59) leading to

$$\underline{G}(\underline{x}_j, t) = \underbrace{\rho^F \underline{u}_h^F(\underline{x}_j, t - h^2) - \rho^F \underline{u}_h^F(\underline{x}_j, t)}_{\text{fluid momentum transport}} - \underbrace{\frac{\rho_h^F}{\tau}(1 - d)(\underline{u}_h^F + \underline{u}^B)}_{\text{momentum exchange between phases}} \, .$$

The particles are assumed to be porous, hence momentum transfer between the fluid and the particle occurs throughout the entire particle volume instead of the boundary only. Therefore $\underline{F}_k^F(t)$ is computed by

$$\underline{F}_k^F(t) = \sum_j \theta(d_k(\underline{x}_j, t)) \sum_i \underline{g}_i(\underline{x}_j, t) \, , \tag{4.33}$$

where

$$\begin{aligned}\theta : \mathbb{R} &\to \{0, 1\} \\ d &\mapsto \begin{cases} 1 : d > 0 \\ 0 : d \leqslant 0 \end{cases} \, ,\end{aligned}$$

Oliver Heaviside ☾ *(1850–1925) British mathematician and physicist*

is the Heaviside function. Similarly the torque $\underline{T}_k^F(t)$ is computed by

$$\underline{T}_k^F(t) = \sum_j \theta(d_k(\underline{x}_j, t)) \, (\underline{x}_j - \underline{X}_k(t)) \times \sum_i \underline{g}_i(\underline{x}_j, t) \, .$$

4.4 IMPLEMENTATIONAL ASPECTS

This section focuses on the implementation of the introduced HLBM ansatz for flows of resolved particles in `OpenLB`. We begin by a short outline of the main LBM loop and explain the additional steps in comparison to the basic LBM. We then explain in more detail parallel aspects on the basis of the calculation of the hydrodynamic force acting on one particle.

An outline of the main LBM loop is written in Listing 4.3. Each time step starts with the calculation of the solid phase, by a loop over all particles. First the forces and torques acting on the particle are computed. We later explain the computation of the hydrodynamic force $\underline{F}_k^F$ and it's parallelisation in more detail. With the forces and torques known, the particle velocity, position, angular velocity and angle are updated using a velocity Verlet algorithm as introduced in Section 3.1.2. Lines 4–6 compute and store the

Listing 4.3: Basic HLBM algorithm

```
0  for t ∈ I_h {
     foreach particle k {
       compute forces F_k^F(t), F_k^pp(t), F_k^pw(t), F_k^g(t) and torque T_k^F
       update U_k(t+h^2), X_k(t+h^2), T_k(t+h^2), ω_k(t+h^2)
       for x ∈ Ω_h ∩ Ω_k^P
5        Set d *= 1 − d_k; N += d_k u_k^P(x,t); D += d_k;
       }
     }
     for x ∈ Ω_h {
       compute u*(x,t) = d(x,t)u^F(x,t) + (1 − d(x,t))u^B(x,t)
10     for i = 0,...,q − 1 {
         compute M_i[ρ^F, u*](x,t)
         f̃_i(x,t) = −1/τ (f_i(x,t) − M_i(x,t))
         f_i(x + h^2 ξ_i, t + h^2) = f̃_i(x,t)
       }
15     reset d = 1; N = 0; D = 0;
     }
     communicate overlap
   }
```

porosity parameter $d(x, T)$ and the enumerator N and denominator D of the fraction in Equation 4.31 for each lattice node covered by the current particle. As OpenLB uses a cell based strategy to store lattice data, the class Cell is extended by an external field to hold the additional data.

Subsequent to the iteration over the particles a loop over the lattice nodes follows, which is basically the LBM as introduced in Listing 2.1 expanded by lines 9, 11 and 15. In line 11 the velocity parameter u^* is computed and used in line 11 to compute the EDF. Finally in line 15 the variables d, N and D are resetted.

The rigid particles are specified in classes of type SmoothIndicatorF2D.

```
template <typename T, typename S>
class SmoothIndicatorF2D : public AnalyticalF2D<T,S> {
protected:
  SmoothIndicatorF2D();
  virtual bool operator() (T output[], const S input[])=0;
  Vector<S,2> _myMin;
  Vector<S,2> _myMax;

public:
  virtual Vector<S,2>& getMin();
  virtual Vector<S,2>& getMax();
};
```

All children of class SmoothIndicatorF2D must have an implementation of the function `bool operator()(T output[], const S input[])`, which calculates the value of d_k as defined in Equations (4.28) to (4.30). Additionally the functions `Vector<S,2>& getMin()` and `Vector<S,2>& getMax()` return references to the position of the lower and upper corners of an axis aligned bounding box, illustrated in Figure 4.16.

Respecting the particle extensions and backcoupling poses new challenges concerning the parallelisation of the code. Extended particles can spread over several subdomains on different Processing Unit (PU). Therefore their

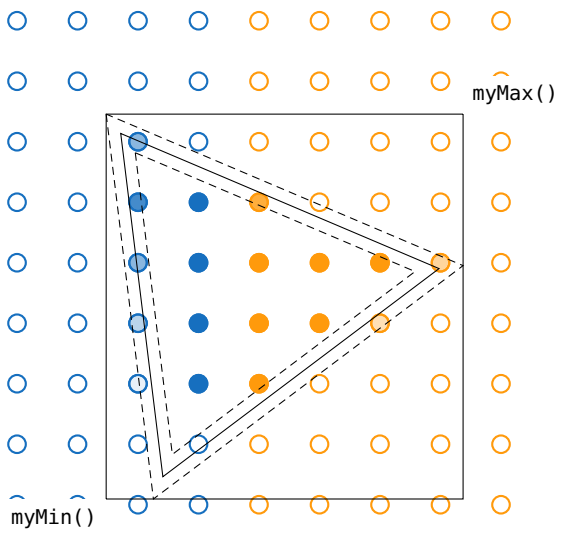


Figure 4.16: A smooth triangle with bounding box spread over two BlockLattice2D indicated by the colour of the nodes. The porosity parameter d_k is indicated by the opacity of the nodes.

basic properties such as geometric position, radius, etc. have to be known to all PUs covering a part of the rigid body. For simplicity's sake this basic information is made available to all PUs. The situation is illustrated in Figure 4.16 for a triangular particle. The two different BlockLattice2D are indicated by the different colours and the porosity parameter d by the opacity of the nodes.

We explain the parallelisation by means of the computation of the hydrodynamic force $\underline{F}^F(t)$, Equation 4.33. The MEA is implemented as written in the second line of Equation 4.32. In this form only the data of directly neighbouring nodes is needed. Using the overlap of the BlockLattice2Ds no additional communication is necessary to compute the force $g_i(\underline{x}_j, t)$ on each single lattice node. Therefore $\underline{F}^F = \sum_{l=0}^{m-1} \underline{F}_l^F$ can be split in m parts $\underline{F}_l^F$, one for each BlockLattice2D. This idea is implemented in the function SuperSumIndicator2D::operator(), which is simultaneously executed on all PUs.

```
template <typename T, template <typename U> class DESCRIPTOR>
bool SuperSumIndicator2D<T,DESCRIPTOR>::operator() (T output[], const
    int input[])
{
  _f.getSuperStructure().communicate();
  LoadBalancer<T>& load = _f.getSuperStructure().getLoadBalancer();
  for (int i = 0; i < this->getTargetDim(); ++i) {
    output[i] = 0.;
  }
  T physR[2], d_k[1], outputTmp[_f.getTargetDim();
  int start[DESCRIPTOR::d] = {0}, span[DESCRIPTOR::d] = {0};
  Cuboid2D<T>* cub = nullptr;

  for (int iC = 0; iC < load.size(); ++iC) {
    int globiC = load.glob(iC);
    cub = &_superGeometry.getCuboidGeometry().get(globiC);

    if (! (cub->get_globPosX() > _indicator.getMax()[0] ||
           cub->get_globPosY() > _indicator.getMax()[1] ||
```

```
          _indicator.getMin()[0] > cub->get_globPosX() + cub->
              getExtend()[0] * cub->getDeltaR() ||
          _indicator.getMin()[1] > cub->get_globPosY() + cub->
              getExtend()[1] * cub->getDeltaR())) {

      for (int k=0; k<DESCRIPTOR<T>::d; k++) {
        start[k] = (_indicator.getMin()[k] - cub->getOrigin()[k]) / cub
            ->getDeltaR() + 1;
        if (start[k] < 0) start[k] = 0;
        cub->getPhysR(physR,start);
        span[k] = (_indicator.getMax()[k] - physR[k]) / cub->getDeltaR
            () + 1;
        if (span[k] + start[k] > cub->getExtend()[k]) span[k] = cub->
            getExtend()[k] - start[k];
      }
```

First the needed variables are initialised. Noteworthy are `T d_k[1];` which represents the porosity parameter d_k and `T output[]` which holds the part $\underline{F}_l^F$ of the current PU. After that the function starts with a loop over all local `Cuboid2D`s and tests whether an intersection of the particles bounding box and the `Cuboid2D`'s extensions exists. If an intersection exists, the coordinates of the first lattice node in the bounding box are determined and stored in the variable `start`. Then the number of lattice nodes in each direction of the intersection is computed and stored in the variable `span`.

```
      for (int iX = start[0]; iX < start[0]+span[0]; iX++) {
        for (int iY = start[1]; iY < start[1]+span[1]; iY++) {
          if (_superGeometry.get(globiC, iX, iY) == 1) {
            cub->getPhysR(physR,iX,iY);
            _indicator(d_k, physR);
            if (d_k[0]) {
              _f(outputTmp,globiC,iX,iY);
              for (int i = 0; i < this->getTargetDim()-1; ++i) {
                output[i] += outputTmp[i];
              }
            }
          }
        }
      }
    }
  }
```

This is followed by a loop over the lattice nodes $\underline{x}$ in the intersection, which results in a major speedup compared to iterating over all lattice nodes. If the node is a fluid node i.e. its material number is one, the porosity parameter $d_k(\underline{x}, t)$ is computed by `_indicator(d_k, physR);` which is an instantiation of `SmoothIndicatorF2D::operator()`. If $d_k \neq 0$ the contribution to $\underline{F}_l^F$ of this lattice node is computed by `_f(outputTmp,globiC,iX,iY);` and added to the previous results.

```
#ifdef PARALLEL_MODE_MPI
  for (int i = 0; i < this->getTargetDim()-1; ++i) {
    singleton::mpi().reduceAndBcast(output[0], MPI_SUM);
  }
#endif
  return true;
}
```

After completing the main loop the single parts $\underline{F}_l^F$ are known on the respective PUs. They are submitted to and totalled on the master node via an

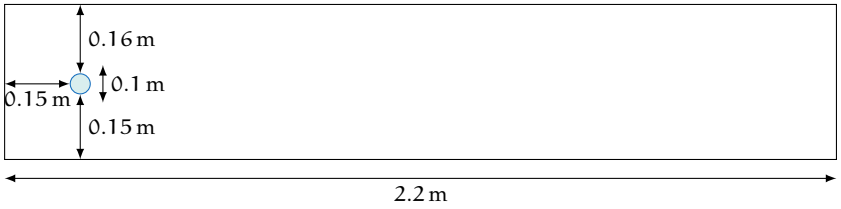


Figure 4.17: Geometry for flow around cylinder.

`MPI_reduce()` command and distributed to all PUs by an `MPI_broadcast()` command, which are wrapped by the function `singleton::mpi().reduceAndBcast (output[i], MPI_SUM)`.

4.5 APPLICATION: NUMERICAL EXPERIMENTS

In this section, the Homogeneous Lattice Boltzmann code is validated by comparison to results existing in literature and convergence analysis. The *flow around a cylinder* test case is used as a first test and can be interpreted as flow around a resting particle. Then, by simulating the sedimentation of a single spherical particle, the behaviour of the method for moving media is investigated. Sedimentation of two spherical particles additionally includes particle-particle interaction. Finally, sedimentation of two square particles and 24 differently shaped particles is simulated to show the feasibility for more complex shapes.

This section is based on article [81], submitted to *Particuology*.

4.5.1 *Flow Around a Cylinder*

The familiar problem of viscous flow around a cylinder is revisited. For the probably most comprehensive discussion of this setup we point to the books by Zdravkovich [199, 200]. Based on the simulation setup in Schäfer et al. [157], drag and lift coefficients are computed and compared with the presented results. An incompressible Newtonian fluid with kinematic viscosity $\nu = 10^{-3}\,m^2/s$ and density $\rho^F = 1000\,kg/m^3$ is considered. A porous cylinder with diameter $D = 0.1\,m$ is placed at $(0.16, 0.2)\,m$, in a rectangular domain of dimension $2.2 \times 0.41\,m^2$ (see Figure 4.17). Velocity boundary conditions are used at the inlet, pressure boundary conditions at the outlet and bounceback boundaries at the upper and lower edge. The inflow velocity has a Poiseuille profile with

$$\underline{u}^F((0,y),t) = \frac{1.2}{0.41}y - \frac{1.2}{0.41^2}y^2\,,$$

which yields a Reynolds number of $Re = 20$. The quantities of interest are the drag coefficient c^D and the lift coefficient c^L, which are defined by

$$c^D = \frac{2\underline{F}^D}{\rho^F \overline{\underline{u}}^2 DH}, \qquad c_L = \frac{2\underline{F}^L}{\rho^F \overline{\underline{u}}^2 DH}\,,$$

where the drag force $\underline{F}^D$ and lift force $\underline{F}^L$ are

$$\underline{F}^D = \int_S \left(\rho^F \nu \frac{\partial \underline{u}_t}{\partial \underline{n}} n_y - p n_x\right)\, dS\,,$$
$$\underline{F}^L = -\int_S \left(\rho^F \nu \frac{\partial \underline{u}_t}{\partial \underline{n}} n_x + p n_y\right)\, dS\,,$$

with the cylinder boundary S, the normal $\underline{n}$ on S, and the tangential velocity $\underline{u}_t$ on S. Results for the computation of the drag coefficients are listed in Table 4.8 and corresponding Figure 4.18 for different resolutions $N = 1, 2, \ldots, 8$ with $220N \times 41N$ lattice nodes and chosen values of the smoothing parameter $\epsilon_h = \epsilon h$ with $\epsilon \in \{1, 0.5, 0.25, 0\}$.

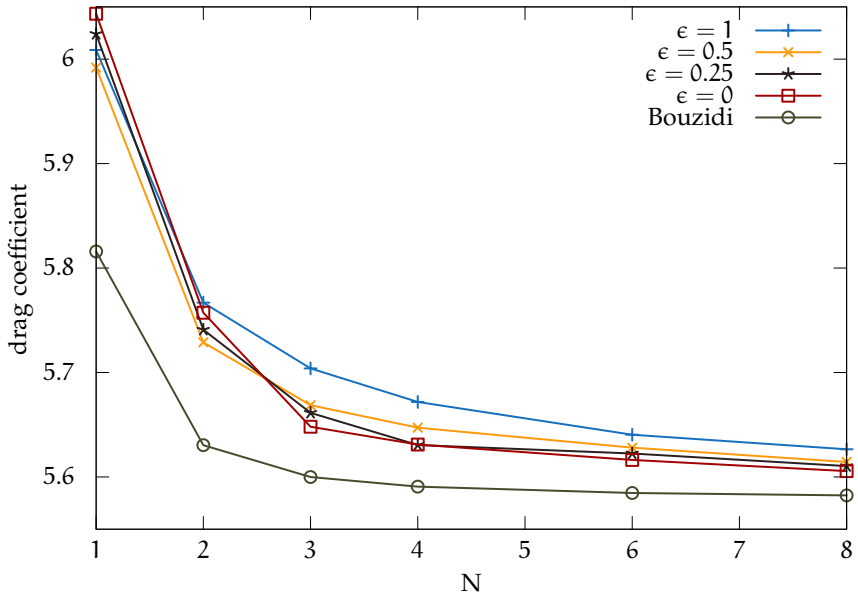


Figure 4.18: Drag coefficients for 'Flow around a cylinder' for different resolutions N and smoothing parameter ϵ.

c_D^N	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N = 6$	$N = 8$
$\epsilon = 1$	6.01	5.77	5.70	5.67	5.64	5.63
$\epsilon = 0.5$	5.99	5.73	5.67	5.65	5.63	5.61
$\epsilon = 0.25$	6.02	5.74	5.66	5.63	5.62	5.61
$\epsilon = 0$	6.04	5.76	5.65	5.63	5.62	5.61
Bouzidi	5.82	5.63	5.60	5.59	5.58	5.58

Table 4.8: Drag coefficient c_D for different resolutions and ϵ for Poiseuille flow around a cylinder, where the cylinder is a two dimensional representation of a spherical particle.

For comparison, also results obtained using Bouzidi's boundary condition [19] for the cylinder are listed. For the highest resolution ($N = 8$) we obtained values between 5.61 for $\epsilon = 0$ and 5.63 for $\epsilon = 1$. Schäfer et al. [157] found a lower bound for the drag coefficient of 5.58 and an upper bound of 5.59, which agrees well with our results.

Results for the lift coefficients are shown in Table 4.9 and Figure 4.19. Our results vary between 0.0108 and 0.0113, which is again in good agreement to those given by Schäfer et al. [157], who state 0.0080 as lower and 0.0100 as upper bound, as well as to the computation of the cylinder with Bouzidi's boundary.

The Experimental Order of Convergence (EOC) of the discretisation parameter N is defined as in Equation (3.20). The EOC of the drag coefficient is computed using $\mathrm{Err}(N) = |c_D^N - c_D^*|$, where c_D^N is the drag coefficient

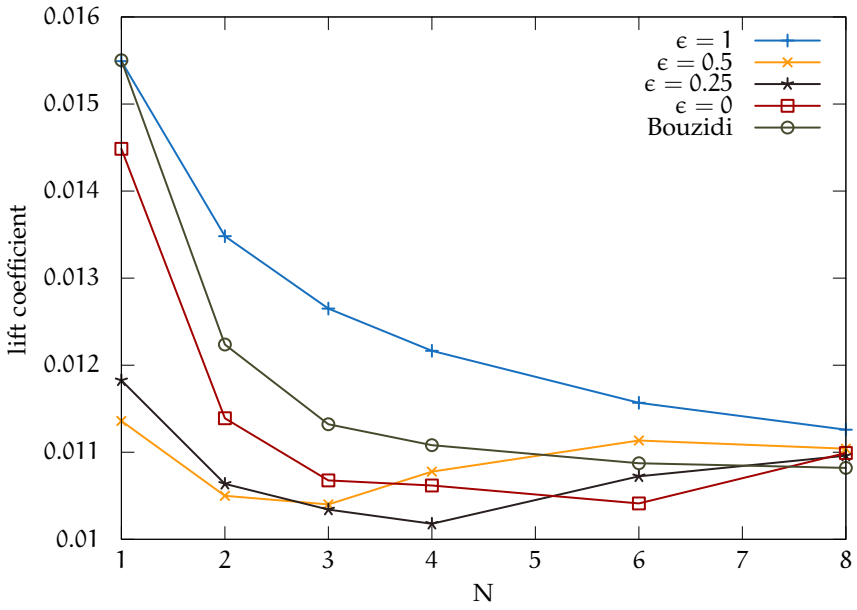


Figure 4.19: Lift coefficients for 'Flow around a cylinder' for different resolutions and smoothing parameter ϵ.

$\times 10^{-2}$	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N = 6$	$N = 8$
$\epsilon = 1$	1.55	1.35	1.27	1.22	1.16	1.13
$\epsilon = 0.5$	1.14	1.05	1.04	1.08	1.11	1.10
$\epsilon = 0.25$	1.18	1.06	1.03	1.02	1.07	1.10
$\epsilon = 0$	1.45	1.14	1.07	1.06	1.04	1.10
Bouzidi	1.55	1.22	1.13	1.11	1.09	1.08

Table 4.9: Lift coefficients for different resolutions and ϵ for Poiseuille flow around a cylinder, where the cylinder is a two dimensional representation of a spherical particle.

obtained for resolution N and c_D^* is the reference coefficient obtained for the highest resolution. The resulting values are listed in Table 4.10 and illustrated in Figure 4.20. The proposed method shows approximately linear EOC for all tested values of ϵ, whereas the simulations using Bouzidi boundary condition are second order accurate, as expected.

4.5.2 *Sedimentation of One Particle*

In order to further validate the HLBM, applications for particulate flows are simulated. In this setup, one circular particle sediments in a domain $\Omega = [0,2] \times [0,8]\,\mathrm{cm}^2$. The particle of radius $R = 0.125\,\mathrm{cm}$ and density $\rho^P = 1.25\,\mathrm{g/cm}^3$ is located at $x^P = (1,4)\,\mathrm{cm}$. Bounceback conditions are used at the boundaries of the domain. The fluid's density and kinematic viscosity are set to $\rho^F = 1\,\mathrm{g/cm}^3$ and $\nu = 0.1\,\mathrm{g/cm}^3$. Initially the particle and the fluid are at rest. The domain is discretised with a step size of $h = 0.01/N\,\mathrm{cm}$, leading to a lattice of $(200N \times 600N)$ nodes. The simulations are repeated for the smoothing parameter $\epsilon_h = \epsilon h, \epsilon \in \{0, 0.5, 1\}$. Quantitative results are plotted in Figure 4.21. Figures 4.21a and 4.21b show the

	$\epsilon = 1$	$\epsilon = 0.5$	$\epsilon = 0.25$	$\epsilon = 0$	Bouzidi
EOC(1,2)	1.20	1.47	1.46	1.39	2.23
EOC(2,3)	1.01	1.28	1.68	2.36	2.28
EOC(3,4)	1.04	0.96	1.65	1.00	2.14
EOC(4,6)	1.03	0.83	0.43	0.84	2.05
EOC(6,8)	0.91	1.17	1.16	1.22	2.41
Average	≈ 1.04	≈ 1.14	≈ 1.28	≈ 1.36	≈ 2.22

Table 4.10: EOC of drag coefficient c_D for flow around cylinder.

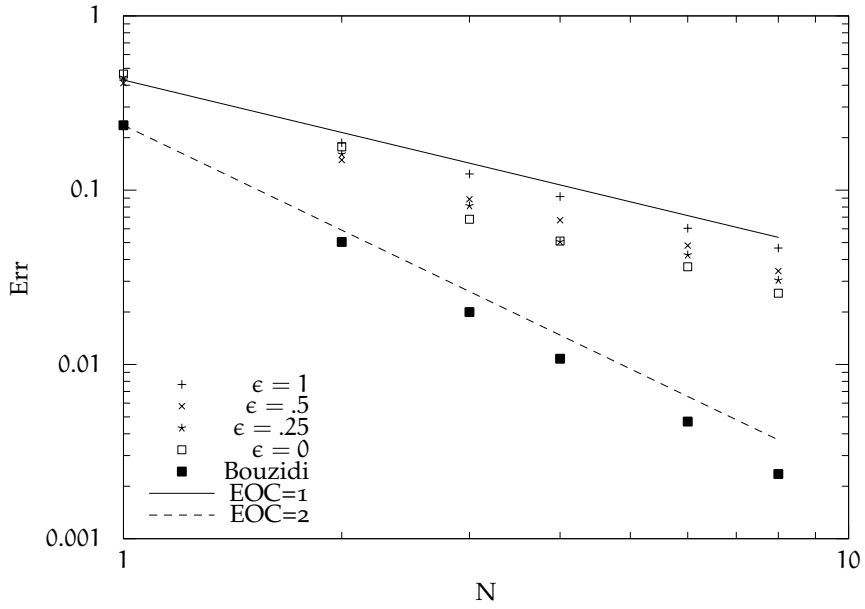


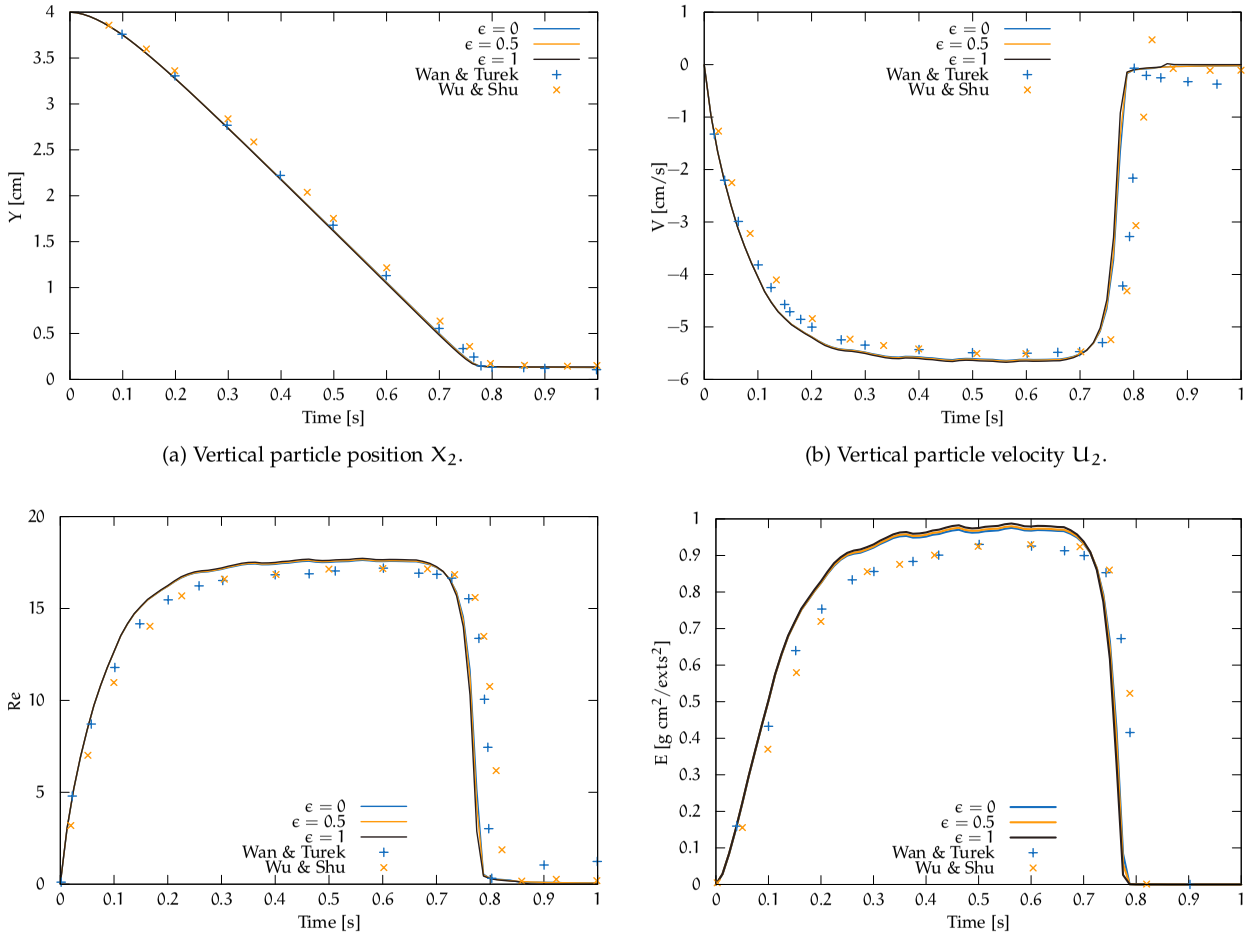
Figure 4.20: EOC of the drag coefficients for flow around a cylinder for different grid resolutions.

evolution of the particle's vertical position and velocity. Figure 4.21c contains the particle's Reynolds number, defined as $Re^P = 2\rho^P R\|\underline{u}^P\|_2/\nu$ and Figure 4.21d the particle's kinetic energy $E = 0.5M\|\underline{u}^P\|_2^2$. One can see that the curves for the varying values of ϵ collapse, therefore the smooth boundary has only a minimal influence on the particle's behaviour. Additionally, the figures show an excellent agreement with the results obtained by Wan and Turek [185] and Wu and Shu [192].

Figure 4.22 illustrates the pressure distribution for the three different values of ϵ. Looking at the isobars one can clearly see that fluctuations are smoothed out for increasing ϵ. The EOCs for $\epsilon_h = 0.5h$ are computed using $\mathrm{Err}(N) = \frac{1}{81}\left(\sum_{k=0}^{80}\left|X_1^N(0.125k) - X_1^{32}(0.125k)\right|^2\right)^{1/2}$ and the results of a simulation with $N = 32$ as reference. They show approximately linear behaviour and can be found in Table 4.11.

4.5.3 *Sedimentation of Two Particles*

The sedimentation of two particles, usually referred to as Drafting, Kissing, Tumbling (DKT) phenomenon, is widely used as a reference setup for the



(a) Vertical particle position X_2.

(b) Vertical particle velocity U_2.

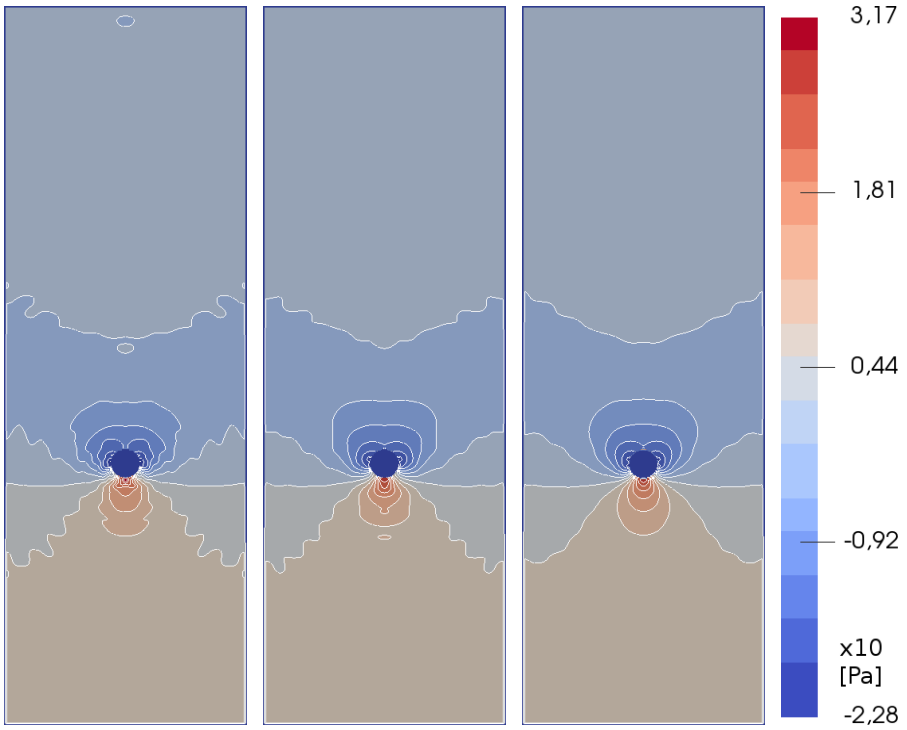
(c) Particle Reynolds number Re.

(d) Kinetic particle energy E.

Figure 4.21: Quantitative results for the simulation of a single sedimenting particle over time. For comparison, the figures contain the results obtained by Wan and Turek [185] and Wu and Shu [192]

(N′, N)	(1, 2)	(2, 4)	(4, 8)	(8, 16)
EOC(N′, N)	1.72	0.81	1.09	1.53

Table 4.11: EOC of sedimentation of one particle and $\epsilon_h = 0.5h$.



(a) $\epsilon = 0$ (b) $\epsilon = 0.5$ (c) $\epsilon = 1$

Figure 4.22: Pressure contours for a single particle sedimenting in a rectangular domain for three different values for the smoothing parameter ϵ. The figure clearly indicates a strong reduction of fluctuation with increasing ϵ.

simulation of particle dynamics submerged in a fluid. In general, the course of the simulations can be divided in three stages, namely the drafting, the kissing and the tumbling stage. Initially, the two particles are positioned on top of each other and accelerate in the direction of gravity, whereby the leading particle creates a wake, in which the trailing particle is caught and therefore experiences less flow resistance, resulting in faster sedimentation. This is called the drafting stage. Next, the two particles collide and form a long body (kissing stage). This long body is unstable and eventually begins to rotate, such that the particles drift away from each other (tumbling stage), with the trailing particle overtaking the leading particle. This process was examined in depth in numerical simulations as well as physical experiments [44, 57, 68, 138]. The test is repeated two times, once using spherical particles and once using square particles. Compared to the simulations in the last section, the following also covers the behaviour of the HLBM for particle-particle collisions.

An incompressible Newtonian fluid with density $\rho^F = 1000\,\mathrm{kg/m^2}$ and kinematic viscosity of $\nu = 10^{-6}\,\mathrm{m^2/s}$ is considered in a rectangular domain of height $H = 0.02$ m in x-direction and length $L = 0.08$ m in y-direction. The particles' radius $R^P = 0.001$ m and density $\rho^P = 1010\ \mathrm{kg/m^2}$. The ϵ—boundary is chosen with a width of h. The particles initial positions are $\underline{X}_1(0) = (0.00999, 0.072)$ m and $\underline{X}_2(0) = (0.01, 0.068)$ m. They move under the influence of gravity and buoyancy in negative y-direction with an acceleration of $g = 9.81(1 - 1000/1010)\ \mathrm{m/s^2}$. The simulations cover a time interval of 5 s.

SPHERICAL PARTICLES

A widely used model in particle-particle, as well as particle-wall collisions is the repulsive force proposed by Wan and Turek [185]. The force acting on particle k due to collision with particle l reads

$$\underline{F}^{pp}_{k,l} = \begin{cases} 0, & D_{kl} > R_{k+l} + \xi \\ \frac{1}{\gamma}\left(\underline{X}_k - \underline{X}_l\right)\left(R_{k+l} - D_{kl}\right), & D_{kl} \leqslant R_{k+l} \\ \frac{1}{\gamma}\left(\underline{X}_k - \underline{X}_l\right)\left(R_{k+l} + \xi - D_{kl}\right)^2, & R_{k+l} \leqslant D_{kl} \leqslant R_{k+l} + \xi \end{cases}, \tag{4.34}$$

with $R_{k+l} = R_k + R_l$ and the distance $D_{kl} = \|\underline{X}_k - \underline{X}_l\|_2$. A particle-wall collision is computed by

$$\underline{F}^{pw}_{k} = \begin{cases} 0, & D_{kk'} > 2R_k + \xi \\ \frac{1}{\gamma_w}\left(\underline{X}_k - \underline{X}'_k\right)\left(2R_k - D_{kk'}\right), & D_{kk'} \leqslant 2R_k \\ \frac{1}{\gamma_w}\left(\underline{X}_k - \underline{X}'_k\right)^2, & 2R_k \leqslant D_{kk'} \leqslant R_k + \xi \end{cases}, \tag{4.35}$$

with $D_{kk'} = \left\|\underline{X}_k - \underline{X}'_k\right\|_2$ the distance $\underline{X}_k$ to the closest point on the wall $\underline{X}'_k$. The parameters are chosen to be $\gamma = 10^{-7}$ and $\gamma_w = 0.5 \cdot 10^{-7}$.

Figure 4.26 shows the simulation at several times. The three stages of the simulation are clearly visible. The drafting stage in Figs. 4.26a and 4.26b, the kissing stage in Figs. 4.26c and 4.26d and the tumbling stage in Figs. 4.26e and 4.26f. The evolution of the horizontal position of the particles is illustrated in Figure 4.23 in comparison to and in good agreement with the results by Feng and Michaelides [57], Niu et al. [139] and Wang, Guo and Mi [186]. The simulation was executed for resolutions of $(200N \times 800N)$, lattice nodes $N = 1, \ldots, 16$. Figure 4.24 shows the evolution of the horizontal positions for these simulations. One can clearly see that the curves merge

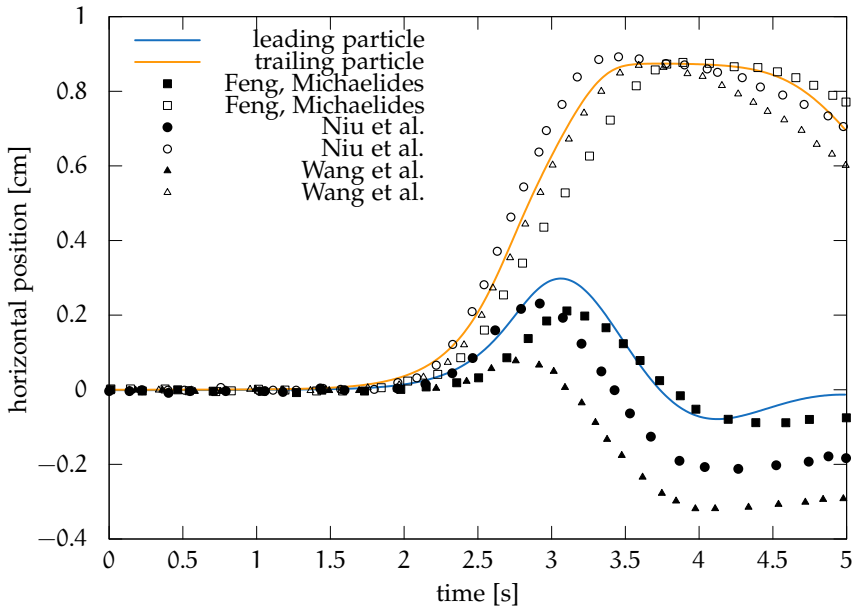


Figure 4.23: Evolution of horizontal position of spherical particles in comparison to results by Feng and Michaelides [57], Niu et al. [139].

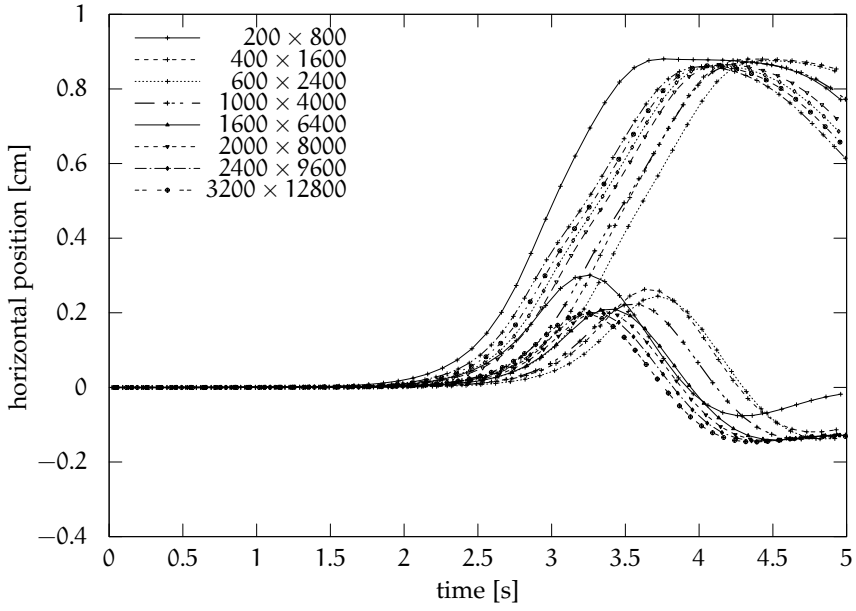


Figure 4.24: Evolution of horizontal position of spherical particles. For increasing resolution the curves adapt to a certain solution.

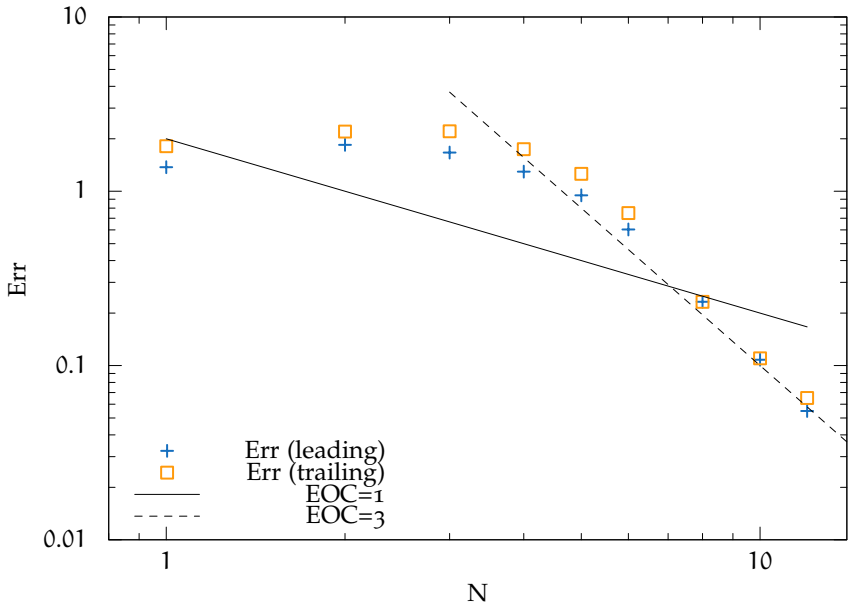


Figure 4.25: Error plot of the horizontal position of leading and trailing particle.

towards the one obtained for the highest resolution. It also indicates, that the resolution commonly used in literature (200×800) might be insufficient. The errors

$$\mathrm{Err}(N) = \frac{1}{501}\left(\sum_{k=0}^{500}\left|X_2^N(0.01k) - X_2^{16}(0.01k)\right|^2\right)^{1/2}$$

have been computed, using $N = 16$ as reference solution. They are illustrated in Figure 4.25 and indicate an EOC between one and three.

SQUARE PARTICLES

To investigate the potential of the proposed method the simulation is repeated using particles of cubical shape, without an additional collision model. Different to the simulations above are the mass and moment of inertia of the particles, as well as the porosity parameter d_k given in Equation (4.29). The particles have edge length $L = 0.001\,\mathrm{m}$ and mass $M = 4.04 \cdot 10^{-3}\,\mathrm{kg}$ and a moment of inertia $J = \frac{2}{3}ML^2$, with its rotation axis at the centre of mass. Most interesting is that although no explicit collision model is used, during the kissing stage a certain momentum transfer occurs and the general behaviour of the DKT stages can be recovered. Figs. 4.27 and 4.28 display the horizontal and vertical particle positions over time. No significant changes of the particles trajectories are seen for resolutions $N = 1, 2, 3$. Figure 4.29 shows the magnitude of the fluid velocity at several points in time.

4.5.4 *Sedimentation of 24 Particles*

In order to investigate the intrinsic collision model of HLBM, sedimentation of triangles, squares and circles is simulated. Eight particles of each type are used in the simulation. The domain Ω, fluid viscosity ν and fluid and particle densities are chosen as in Subsection 4.5.3. The radius of the circumscribed circle of all particles is $R^P = 0.001$. The two topmost particles in the centre are positioned as before. The remaining particles are positioned

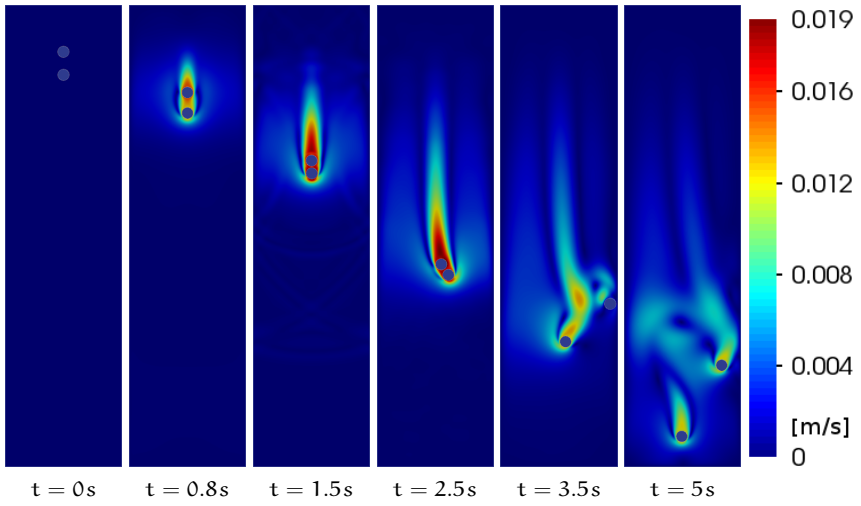


Figure 4.26: Fluid velocity $\underline{u}^F$ in m/s for DKT of spherical particles at different times t.

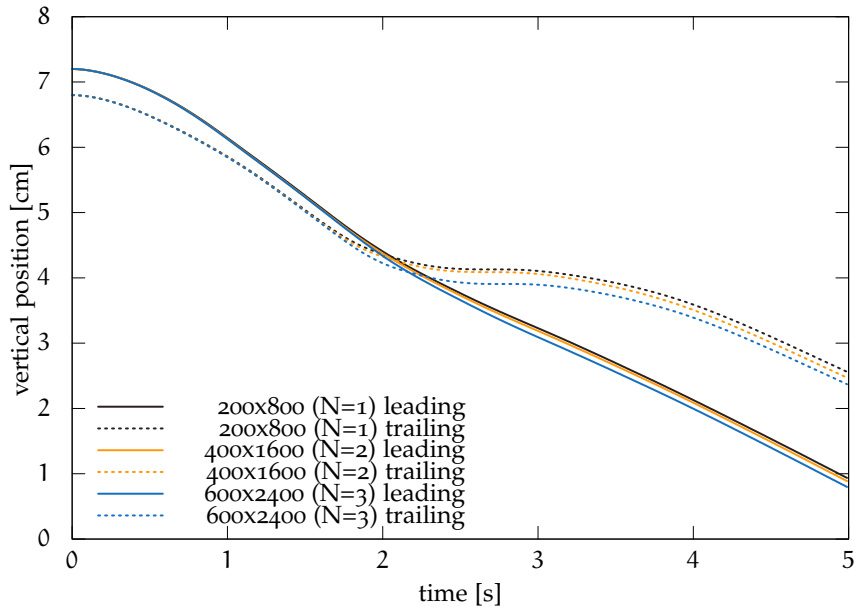


Figure 4.27: Evolution of vertical position of DKT for square particles without collision model.

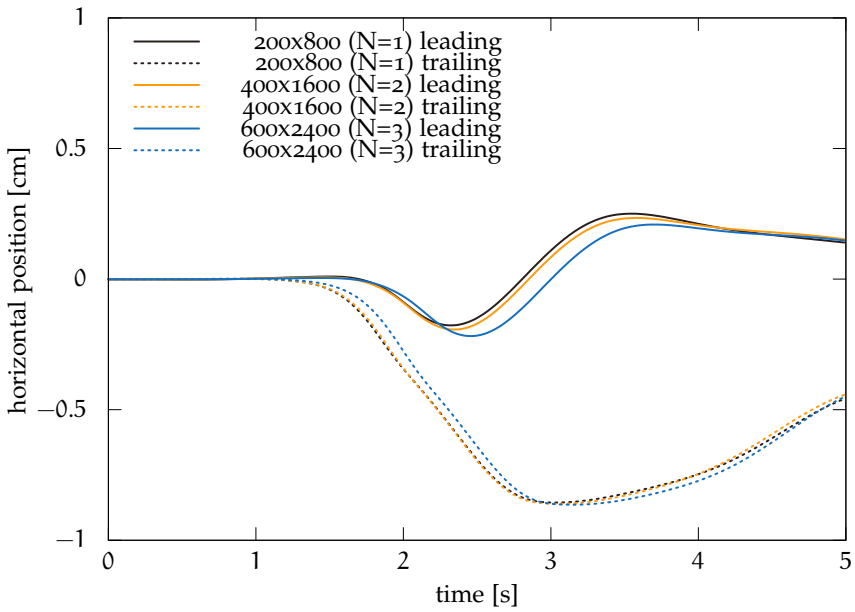


Figure 4.28: Evolution of horizontal position of DKT for square particles without collision model.

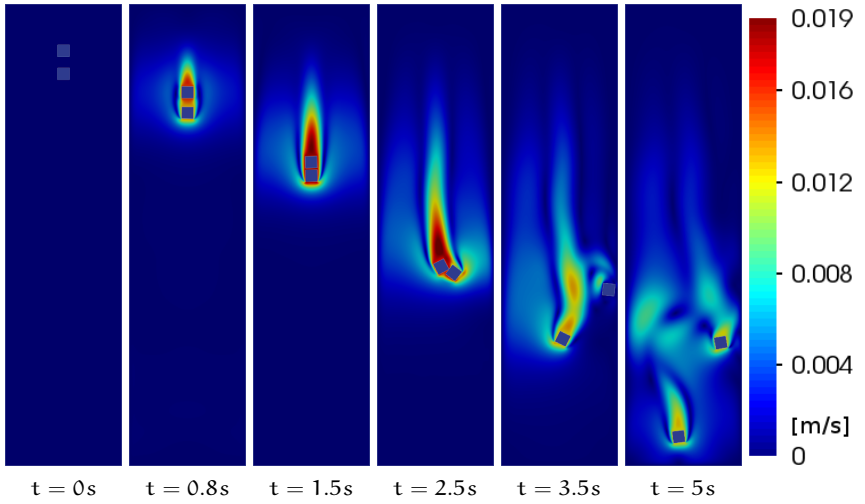


Figure 4.29: Fluid velocity $\underline{u}^F$ in m/s for DKT of cubical particles at different times t.

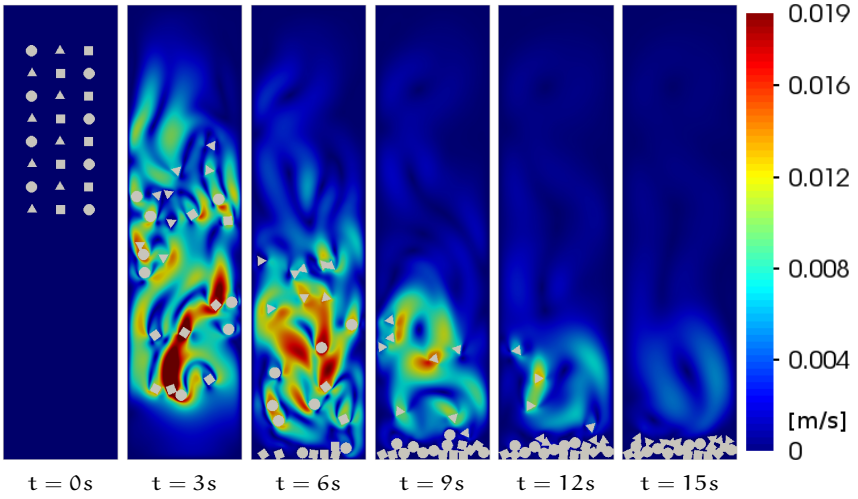


Figure 4.30: Fluid velocity $\underline{u}_h^F$ in m/s for sedimentation of 24 particles at different times t.

with a distance of 0.004 m in each direction. The shape of the particles is displayed in Figure 4.30. One can find a small overlapping area of touching particles, especially for triangles. This is expected for the method. The size of the overlapping area is influenced by the systems relaxation parameter τ.

4.5.5 *Summary*

The HLBM for simulation of moving porous media is proposed. It is applied to simulation of particulate flows, where particles are assumed to have a solid centre with a thin boundary layer of increasing porosity. The method is used to simulate flow around a cylinder as a first scenario to determine its behaviour for a single fixed particle. It is found to be in good agreement with literature and of linear EOC. Sedimentation of a single particle is used to test the method on a moving obstacle. Again good agreement with literature is observed. Additionally the results suggest that pressure fluctuations that remained challenging in Ladd's approach [113] are greatly reduced. Finally, sedimentation of two particles is simulated to include particle-particle interaction. The results for spherical particles also agree with literature. To show the capabilities of the method, sedimentation of two square particles is realised. Most interesting is that although no explicit collision model is used, at the particle collision a certain momentum transfer occurs and the general behaviour of the drafting, kissing, tumbling process can be recovered. Finally sedimentation of 24 particles of different shapes is simulated to strengthen the position, that the method is capable of simulating arbitrarily shaped particles.

In summary the newly proposed method for simulation of particulate flows shows at least a linear EOC. It does not need an additional Lagrangian mesh as used e. g. by the IBM and can therefore omit cost-intensive interpolations. The implementation of particles of arbitrary shape is easy and straightforward and finally the intrinsic parallel efficiency of LBM is conserved.

5 SUMMARY AND CONCLUSIONS

Particulate flows are permanently present in our everyday lifes. Their simulation can help to advance and understand numerous natural and technical as well as biological processes. Conventional numerical methods that are based on the Navier–Stokes Equation (NSE) have been found to be of limited efficiency for the simulation of particulate flows. Therefore, the aim of this thesis was the study of Lattice Boltzmann Methods (LBMs) with respect to particulate flows. We have investigated a broad range of mathematical models and numerical methods. The following contains a dense summary of the work, its results and achievements, as well as some perspective challenges.

The work was split in three parts of increasingly complex models. The first part covered one phase flows with LBMs, the second part covered dilute, one way coupled particle flows, while the third part covered dense two and four way coupled particle flows.

In the first part the basic LBM was introduced as well as several schemes to simulate turbulence, forces, and porous media, including the newly proposed Homogenised Lattice Boltzmann Method (HLBM) as a new scheme to simulate moving porous media. We gave insight into a newly implemented voxeliser that obtains a structured grid from a complex triangulated surface and explained the parallelisation concept of domain decomposition for the LBM. The first part finished by a conference article on simulation of non-steady blood flow of a patient specific aortic arch with a thoracic coarctation, executed in parallel using 512 cores. This demonstrates the outstanding ability of the LBM to efficiently use massively parallel systems as well as its capabilites of handling flow simulations in highly complex domains.

The second part covered dilute particles, which where connected to the carrier fluid by a one-way coupling. We first introduced an Euler–Lagrange (EL) approach and faced the problem, that the domain decomposition used for fluid simulation is not optimal for the parallel computation of particle trajectories. Therefore, we proposed a *load optimal* and a *communication optimal strategy*. Both strategies are based on the given fixed domain decomposition from the underlying fluid computation. Under the assumption of homogeneous distribution of particles the communication optimal strategy was found to be more efficient and therefore implemented and validated. Convergence of the implemented method was shown for varying lattice resolutions and particle numbers. Speedup tests were conducted and led to super-linear results. We therefore found a parallelisation strategy that scales excellently for dilute flows of evenly distributed Lagrangian particles.
The method was then used to investigate particle deposition in the human nasal cavity for unsteady airflow for one respiratory cycle. Particles of several radii were injected once, as well as continuously during inspiration. It was found that particles of smaller Stokes number tended to exit through the trachea, while particles of greater Stokes number mostly deposited. Repeated injection showed higher deposition for all simulated Stokes numbers. 10 µm particles were investigated in more detail. A significant amount of particles got lost through the nostrils in consequence of a backflow. In both cases most particles deposited in the anterior region and remained in the sinuses after one completed breathing cycle. Therefore, transient simulations

are necessary, when simulating particle flow in nasal cavities.
A disadvantage of EL approaches is that computational effort grows with the number of simulated particles. As an adequate amount of particles has to be considered to reach a desired level of accuracy, the computational requirements of EL can be enormous. However, the computational complexity of Euler–Euler (EE) methods increases only with the numerical resolution, with the drawback that information on single particle trajectories gets lost. An EE approach was extended to the domain of dilute and inertial particles, to take advantage of its easy parallelisability in the framework of LBM. Since this method allowed to formulate boundary conditions on a mesoscopic scale, a new capture boundary was introduced. The problem of distorted results caused by the stabilisation approach has been solved by normalising the escape rate with a reference value.
Using the newly developed EE scheme the previous simulations of the tracheobronchial bifurcation have been repeated. The results of the two different simulation approaches were found to be congruent. By considering the artificial diffusion by a proposed reasonable correction of the obtained data with computed reference values, the solution was found to fit the physical model. We therefore newly developed an EE approach and for the first time simulated dilute particulate flows using LBM for both phases.

The third part covered two- and four-way coupled flows. We started by introducing two algorithms with the aim to reduce the complexity of contact detection. The first one was based on a k-d tree, the second one on a grid. After implementation the algorithm runtimes were compared for three different scenarios. Particle contact was computed for two structured particle distribution and one random particle distribution. We found that the runtime of both algorithms was comparable for the tested structured particle distributions and up to one million particles. In case of the random particle distribution the grid based algorithm clearly outruned the k-d tree algorithm and should therefore be prefered.
We introduced two new methods for simulation of flow of subgrid scaled particles. The first approach employed the HLBM introduced in Section 2.3.5. Investigation of one sedimenting particle showed a strong τ-dependency. It was found that by reducing the particle radius the deviation of the sedimentation velocity compared to an analytic solution could be reduced to less then 1%. The second approach employed the forcing scheme introduced in Section 2.3.3. Again sedimentation of one particle was simulated. The τ-dependency of the previous scheme was not regained. Sedimentation of 8125 interacting particles was realised. The simulation showed expected results and qualitatively matches the results found in the literature. However, for grid refinement the quality of the results decreases and further research is necessary. Also concerning the HLBM for subgrid sclae particles further research is necessary.
For simulation of particles with a size comparable to the characteristic length of the system the HLBM for simulation of moving porous media was proposed and applied to simulation of particulate flows, where particles are assumed to have a solid centre with a thin boundary layer of increasing porosity. As a first scenario the method was used to simulate flow around a cylinder in order to determine its behaviour for a single fixed particle. It was found to be in good agreement with the literature and of linear Experimental Order of Convergence (EOC). Sedimentation of a single particle was used to test the method on a moving obstacle. Again good agreement with the literature was observed. Additionally the results suggested that

pressure fluctuations which remained challenging in Ladd's approach were greatly reduced. Sedimentation of two particles was simulated to include particle–particle interaction. The results for spherical particles also agreed with the literature. To show the capabilities of the method, sedimentation of two square particles was realised. Most interesting is that although no explicit collision model was used, during particle contact a certain momentum transfer occured and the general behaviour of the drafting, kissing, tumbling process was recovered. Finally, sedimentation of 24 particles of different shape was simulated to strengthen the position, that the method is capable of simulating arbitrarily shaped particles.
In summary the newly proposed method for simulation of particulate flows showed at least a linear EOC. It does not need an additional Lagrangian mesh as used e. g. by the Immersed Boundary Method (IBM) and can therefore omit cost-intensive interpolations. The implementation of particles of arbitrary shape is easy and straightforward and finally the intrinsic parallel efficiency of LBM is conserved.

This brings us a step closer to the initially mentioned great challenge of finding an efficient numerical algorithm to predict the dynamics of millions of arbitrarily shaped submerged particles.

[1] T. Abe. "Derivation of the lattice Boltzmann method by means of the discrete ordinate method for the Boltzmann equation". In: *Journal of Computational Physics* 131.1 (1997), pp. 241–246.

[2] C. Aidun and J. Clausen. "Lattice-Boltzmann method for complex flows". In: *Annual Review of Fluid Mechanics* 42 (2010), pp. 439–472. DOI: 10.1146/annurev-fluid-121108-145519.

[3] C. Aidun, Y. Lu and E.-J. Ding. "Direct analysis of particulate suspensions with inertia using the discrete Boltzmann equation". In: *Journal of Fluid Mechanics* 373 (Oct. 1998), pp. 287–311. DOI: 10.1017/S0022112098002493.

[4] B. J. Alder and T. E. Wainwright. "Studies in Molecular Dynamics. I. General Method". In: *The Journal of Chemical Physics* 31.2 (1959), pp. 459–466. DOI: http://dx.doi.org/10.1063/1.1730376.

[5] F. J. Alexander, H. Chen, S. Chen and G. D. Doolen. "Lattice Boltzmann model for compressible fluids". In: *Phys. Rev. A* 46 (4 Aug. 1992), pp. 1967–1970. DOI: 10.1103/PhysRevA.46.1967.

[6] M. P. Allen and D. J. Tildesley. *Computer simulation of liquids*. Oxford university press, 1989.

[7] M. Augustin, A. Caiazzo, A. Fiebach, J. Fuhrmann, V. John, A. Linke and R. Umla. "An assessment of discretizations for convection-dominated convection–diffusion equations". In: *Computer Methods in Applied Mechanics and Engineering* 200.47–48 (2011), pp. 3395–3409. DOI: 10.1016/j.cma.2011.08.012.

[8] T. Auton, J. Hunt and M. Prud'Homme. "The force exerted on a body in inviscid unsteady non-uniform rotational flow". In: *Journal of Fluid Mechanics* 197 (1988), pp. 241–257.

[9] H. Babovsky. *Die Boltzmann-Gleichung*. Teubner Verlag, 2002.

[10] C. Bardos, F. Golse and D. Levermore. "Fluid dynamic limits of kinetic equations. I. Formal derivations". In: *Journal of Statistical Physics* 63 (Apr. 1991), pp. 323–344. DOI: 10.1007/BF01026608.

[11] C. Bardos, F. Golse and C. D. Levermore. "Fluid dynamic limits of kinetic equations II convergence proofs for the boltzmann equation". In: *Communications on Pure and Applied Mathematics* 46.5 (1993), pp. 667–753. DOI: 10.1002/cpa.3160460503.

[12] A. B. Basset. *A treatise on hydrodynamics: with numerous examples*. Vol. 2. Deighton, Bell and Company, 1888.

[13] G. K. Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[14] N. Bellomo and R. Gatignol. *Lecture notes on the discretization of the Boltzmann equation*. Vol. 63. World Scientific, 2003.

[15] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi and E. Kaxiras. "A flexible high-performance Lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries". In: *Concurrency and Computation: Practice and Experience* 22.1 (2010), pp. 1–14. DOI: 10.1002/cpe.1466.

[16] 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". In: *Phys. Rev.* 94 (1954), pp. 511–525. DOI: 10.1103/PhysRev.94.511.

[17] C.-U. Böttner. "Über den Einfluss der elektrostatischen Feldkraft auf turbulente Zweiphasenströmungen". PhD thesis. Martin-Luther - Universität Halle-Wittenberg, 2002.

[18] J. Boussinesq. *Théorie analytique de la chaleur: mise en harmonie avec la thermodynamique et avec la théorie mécanique de la lumière*. Vol. 2. Gauthier-Villars, 1903.

[19] M. Bouzidi, M. Firdaouss and P. Lallemand. "Momentum transfer of a Boltzmann-lattice fluid with boundaries". In: *Physics of Fluids* 13.11 (2001), pp. 3452–3459. DOI: 10.1063/1.1399290.

[20] H. Brinkman. "A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles". In: *Applied Scientific Research* 1.1 (1949), pp. 27–34.

[21] R. A. Brown. "Building a Balanced k-d Tree in $O(kn \log n)$ Time". In: *Journal of Computer Graphics Techniques (JCGT)* 4.1 (Mar. 2015), pp. 50–68.

[22] Y. A. Buevich. "Motion resistance of a particle suspended in a turbulent medium". In: *Fluid Dynamics* 1.6 (1966), pp. 119–119. DOI: 10.1007/BF01022298.

[23] F. Bülow. *Numerical simulation of destabilizing heterogeneous suspensions at vanishing Reynolds numbers*. Karlsruhe, 2015.

[24] A. Caiazzo and M. Junk. "Asymptotic analysis of lattice Boltzmann methods for flow-rigid body interaction". In: *Progress in Computational Physics* 3 (2013), p. 91.

[25] A. Caiazzo and M. Junk. "Boundary forces in lattice Boltzmann: Analysis of momentum exchange algorithm". In: *Computers & Mathematics with Applications* 55.7 (2008), pp. 1415–1423. DOI: 10.1016/j.camwa.2007.08.004.

[26] C. Cercignani. *Mathematical methods in kinetic theory*. Springer, 1969.

[27] C. Cercignani. *The Boltzmann Equation and its Applications*. Springer-Verlag, New York, 1988.

[28] Z. Chai and B. Shi. "A novel lattice Boltzmann model for the Poisson equation". In: *Applied Mathematical Modelling* 32.10 (2008), pp. 2050–2058. DOI: http://dx.doi.org/10.1016/j.apm.2007.06.033.

[29] H. Chen, S. Chen and W. H. Matthaeus. "Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method". In: *Physical Review A* 45.8 (1992), R5339.

[30] S. Chen and G. Doolen. "Lattice Boltzmann Method for Fluid Flows". In: *Annual Review of Fluid Mechanics* 30 (1998), pp. 329–364.

[31] X. B. Chen, H. P. Lee, V. F. H. Chong and D. Y. Wang. "A Computational Fluid Dynamics Model for Drug Delivery in a Nasal Cavity with Inferior Turbinate Hypertrophy". In: *Journal of Aerosol Medicine and Pulmonary Drug Delivery* 23.5 (2010), pp. 329–338. DOI: 10.1089/jamp.2009.0776.

[32] Y. Chen, Q. Cai, Z. Xia, M. Wang and S. Chen. "Momentum-exchange method in lattice Boltzmann simulations of particle-fluid interactions". In: *Phys. Rev. E* 88 (1 July 2013), p. 013303. DOI: 10.1103/PhysRevE.88.013303.

[33] K.-H. Cheng, Y.-S. Cheng, H.-C. Yeh, R. A. Guilmette, S. Q. Simpson, Y.-H. Yang and D. L. Swift. "In vivo measurements of nasal airway dimensions and ultrafine aerosol deposition in the human nasal and oral airways". In: *Journal of Aerosol Science* 27.5 (1996), pp. 785–801. DOI: 10.1016/0021-8502(96)00029-8.

[34] Y. Cheng, T. Holmes, J. Gao, R. Guilmette, S. Li, Y. Surakitbanharn and C. Rowlings. "Characterization of Nasal Spray Pumps and Deposition Pattern in a Replica of the Human Nasal Airway". In: *Journal of Aerosol Medicine* 14.2 (2001), pp. 267–280. DOI: 10.1089/08942680152484199.

[35] R. Clift, J. Grace and M. Weber. *Bubbles, Drops, and Particles*. Academic Press, 1978.

[36] S. Corrsin and J. Lumley. "On the equation of motion for a particle in turbulent fluid". In: *Applied Scientific Research, Section A* 6.2 (1956), pp. 114–116. DOI: 10.1007/BF03185030.

[37] R. Courant, E. Isaacson and M. Rees. "On the solution of nonlinear hyperbolic differential equations by finite differences". In: *Communications on Pure and Applied Mathematics* 5.3 (1952), pp. 243–255. DOI: 10.1002/cpa.3160050303.

[38] C. Crowe, M. Sommerfeld and Y. Tsuji. *Multiphase Flows with Droplets and Particles*. CRC Press LLC, 1998.

[39] P. A. Cundall. "A computer model for simulating progressive large scale movements in blocky rock systems". In: *Proceedings of the international symposium on rock fracture (ISRM), Nancy*. Vol. 1. II–8. 1971, pp. 129–136.

[40] P. A. Cundall and O. D. Strack. "A discrete numerical model for granular assemblies". In: *Geotechnique* 29.1 (1979), pp. 47–65.

[41] E. Cunningham. "On the Velocity of Steady Fall of Spherical Particles through Fluid Medium". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 83.563 (1910), pp. 357–365. DOI: 10.1098/rspa.1910.0024.

[42] E. L. Cussler. *Diffusion: mass transfer in fluid systems*. Cambridge university press, 2009.

[43] T. Darden, D. York and L. Pedersen. "Particle mesh Ewald: An N log(N) method for Ewald sums in large systems". In: *The Journal of Chemical Physics* 98.12 (1993), pp. 10089–10092. DOI: http://dx.doi.org/10.1063/1.464397.

[44] S. M. Dash, T.-S. Lee and H. Huang. "A novel flexible forcing hybrid IB-LBM scheme to simulate flow past circular cylinder". In: *International Journal of Modern Physics C* 25.01 (2014), p. 1340014. DOI: 10.1142/S0129183113400147.

[45] P. Deuflhard, F. Bornemann and A. Hohmann. *Numerische Mathematik*. De Gruyter Lehrbuch Bd. 1. De Gruyter, 2002.

[46] D. d'Humieres, I. Ginzburg, M. Krafczyk, P. Lallemand and L.-S. Luo. "Multiple-Relaxation-Time Lattice Boltzmann Models in Three Dimensions". In: *Philosophical Transactions: Mathematical, Physical and Engineering Sciences* 360.1792 (2002), pp. 437–451.

[47] S. Donath, J. Götz, C. Feichtinger, K. Iglberger and U. Rüde. "waLBerla: Optimization for Itanium-based Systems with Thousands of Processors". In: *High Performance Computing in Science and Engineering, Garching/Munich 2009: Transactions of the Fourth Joint HLRB and KONWIHR Review and Results Workshop, Dec. 8-9, 2009, Leibniz Supercomputing Centre, Garching/Munich, Germany*. Ed. by S. Wagner, M. Steinmetz, A. Bode and M. M. Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 27–38. DOI: 10.1007/978-3-642-13872-0_3.

[48] F. Durst. *Fluid Mechanics: An Introduction to the Theory of Fluid Flows*. SpringerLink: Springer e-Books. Springer Berlin Heidelberg, 2008.

[49] S. Elghobashi. "Particle-laden turbulent flows: direct simulation and closure models". In: *Applied Scientific Research* 48.3 (1991), pp. 301–314. DOI: 10.1007/BF02008202.

[50] A. Eshghinejadfard, A. Abdelsamie, G. Janiga and D. Thévenin. "Direct-forcing immersed boundary lattice Boltzmann simulation of particle/fluid interactions for spherical and non-spherical particles ". In: *Particuology* (2015). DOI: 10.1016/j.partic.2015.05.004.

[51] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee and L. G. Pedersen. "A smooth particle mesh Ewald method". In: *The Journal of Chemical Physics* 103.19 (1995), pp. 8577–8593. DOI: http://dx.doi.org/10.1063/1.470117.

[52] P. P. Ewald. "Die Berechnung optischer und elektrostatischer Gitterpotentiale". In: *Annalen der Physik* 369.3 (1921), pp. 253–287. DOI: 10.1002/andp.19213690304.

[53] M. Farazmand and G. Haller. "The Maxey–Riley equation: Existence, uniqueness and regularity of solutions". In: *Nonlinear Analysis: Real World Applications* 22 (2015), pp. 98–106.

[54] E. Fattahi, C. Waluga, B. Wohlmuth and U. Rüde. "Large Scale Lattice Boltzmann Simulation for the Coupling of Free and Porous Media Flow". In: *High Performance Computing in Science and Engineering: Second International Conference, HPCSE 2015, Soláň, Czech Republic, May 25-28, 2015, Revised Selected Papers*. Ed. by T. Kozubek, R. Blaheta, J. Šístek, M. Rozložník and M. Čermák. Cham: Springer International Publishing, 2016, pp. 1–18. DOI: 10.1007/978-3-319-40361-8_1.

[55] C. Feichtinger, J. Götz, S. Donath, K. Iglberger and U. Rüde. "WaLBerla: Exploiting Massively Parallel Systems for Lattice Boltzmann Simulations". In: *Parallel Computing: Numerics, Applications, and Trends*. Ed. by R. Trobec, M. Vajteršic and P. Zinterhof. London: Springer London, 2009, pp. 241–260. DOI: 10.1007/978-1-84882-409-6_8.

[56] Z.-G. Feng and E. E. Michaelides. "Proteus: a direct forcing method in the simulations of particulate flows". In: *Journal of Computational Physics* 202.1 (2005), pp. 20–51. DOI: 10.1016/j.jcp.2004.06.020.

[57] Z.-G. Feng and E. E. Michaelides. "The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems". In: *Journal of Computational Physics* 195.2 (2004), pp. 602–628. DOI: 10.1016/j.jcp.2003.10.013.

[58] J. Fietz, M. J. Krause, C. Schulz, P. Sanders and V. Heuveline. "Optimized hybrid parallel lattice Boltzmann fluid flow simulations on complex geometries". In: *Euro-Par 2012 Parallel Processing*. Springer, 2012, pp. 818–829.

[59] M. Finck, D. Hänel and I. Wlokas. "Simulation of nasal flow by lattice Boltzmann methods". In: *Comput. Biol. Med.* 37.6 (2007), pp. 739–749. DOI: http://dx.doi.org/10.1016/j.compbiomed.2006.06.013.

[60] E. G. Flekkøy. "Lattice Bhatnagar-Gross-Krook models for miscible fluids". In: *Phys. Rev. E* 47 (6 June 1993), pp. 4247–4257. DOI: 10.1103/PhysRevE.47.4247.

[61] U. Frisch, B. Hasslacher and Y. Pomeau. "Lattice-Gas Automata for the Navier-Stokes Equation". In: *Phys. Rev. Lett.* 56.14 (Apr. 1986), pp. 1505–1508. DOI: 10.1103/PhysRevLett.56.1505.

[62] U. Frisch, D. d'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, J.-P. Rivet et al. "Lattice gas hydrodynamics in two and three dimensions". In: *Complex systems* 1.4 (1987), pp. 649–707.

[63] X. Fu, Z. Yao and X. Zhang. "Numerical investigation of polygonal particle separation in microfluidic channels". In: *Microfluidics and Nanofluidics* 20.7 (2016), pp. 1–14. DOI: 10.1007/s10404-016-1772-8.

[64] I. Gallagher, L. Saint-Raymond and B. Texier. *From Newton to Boltzmann: hard spheres and short-range potentials*. European mathematical society, 2013.

[65] M. A. Gallivan, D. R. Noble, J. G. Georgiadis and R. O. Buckius. "An Evaluation of the Bounce-Back Boundary Condition for Lattice Boltzmann Simulations". In: *International Journal for Numerical Methods in Fluids* 25.3 (1997), pp. 249–263. DOI: 10.1002/(SICI)1097-0363(19970815)25:3<249::AID-FLD546>3.0.CO;2-7.

[66] R. Glowinski, T. Pan, T. Hesla, D. Joseph and J. Periaux. "A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow". In: *Journal of Computational Physics* 169.2 (2001), pp. 363–426. DOI: 10.1006/jcph.2000.6542.

[67] R. Glowinski, T.-W. Pan and J. Periaux. "A fictitious domain method for Dirichlet problem and applications". In: *Computer Methods in Applied Mechanics and Engineering* 111.3 (1994), pp. 283–303. DOI: http://dx.doi.org/10.1016/0045-7825(94)90135-X.

[68] R. Glowinski, T.-W. Pan, T. I. Hesla and D. D. Joseph. "A distributed Lagrange multiplier/fictitious domain method for particulate flows". In: *International Journal of Multiphase Flow* 25.5 (1999), pp. 755–794. DOI: 10.1016/S0301-9322(98)00048-2.

[69] J. Götz, C. Feichtinger, K. Iglberger, S. Donath and U. Rüde. "Large scale simulation of fluid structure interaction using Lattice Boltzmann methods and thephysics engine". In: *ANZIAM Journal* 50 (2008), pp. 166–188.

[70] H. Grad. "On the kinetic theory of rarefied gases". In: *Communications on Pure and Applied Mathematics* 2.4 (1949), pp. 331–407. DOI: 10.1002/cpa.3160020403.

[71] Z. Guo, C. Zheng and B. Shi. "Discrete lattice effects on the forcing term in the lattice Boltzmann method". In: *Phys. Rev. E* 65 (2002), p. 046308. DOI: 10.1103/PhysRevE.65.046308.

[72] Z. Guo and C. Shu. *Lattice Boltzmann Method and Its Applications in Engineering (Advances in Computational Fluid Dynamics)*. 1st ed. World Scientific Publishing Company, Mar. 2013.

[73] Z. Guo and T. Zhao. "Lattice Boltzmann model for incompressible flows through porous media". In: *Physical Review E* 66.3 (2002), p. 036304. DOI: 10.1103/PhysRevE.66.036304.

[74] D. Hänel. *Molekulare Gasdynamik*. Springer, 2004.

[75] M. Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Mathematische Leitfäden. Teubner, 2006.

[76] J. Hardy, Y. Pomeau and O. de Pazzis. "Time Evolution of a Two-Dimensional Classical Lattice System". In: *Phys. Rev. Lett.* 31 (5 July 1973), pp. 276–279. DOI: 10.1103/PhysRevLett.31.276.

[77] X. He and L.-S. Luo. "Lattice Boltzmann Model for the Incompressible Navier Stokes Equation". In: *Journal of Statistical Physics* 88 (1997), pp. 927–944.

[78] X. He and L.-S. Luo. "Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation". In: *Phys. Rev. E* 56.6 (Dec. 1997), pp. 6811–6817. DOI: 10.1103/PhysRevE.56.6811.

[79] X. He, X. Shan and G. D. Doolen. "Discrete Boltzmann equation model for nonideal gases". In: *Phys. Rev. E* 57.1 (Jan. 1998), R13–R16. DOI: 10.1103/PhysRevE.57.R13.

[80] T. Henn, V. Heuveline, M. J. Krause and S. Ritterbusch. "Statistical Atlases and Computational Models of the Heart. Imaging and Modelling Challenges: Third International Workshop, STACOM 2012, Held in Conjunction with MICCAI 2012, Nice, France, October 5, 2012, Revised Selected Papers". In: ed. by O. Camara, T. Mansi, M. Pop, K. Rhode, M. Sermesant and A. Young. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. Chap. Aortic Coarctation Simulation Based on the Lattice Boltzmann Method: Benchmark Results, pp. 34–43. DOI: 10.1007/978-3-642-36961-2_5.

[81] T. Henn, F. Klemens, G. Thäter and M. J. Krause. "Particle Flow Simulations with Homogenised Lattice Boltzmann Methods". In: *Submitted to Particuology* (2016).

[82] T. Henn, G. Thäter, W. Dörfler, H. Nirschl and M. J. Krause. "Parallel dilute particulate flow simulations in the human nasal cavity". In: *Computers & Fluids* 124 (2016), pp. 197–207. DOI: 10.1016/j.compfluid.2015.08.002.

[83] V. Heuveline and M. Krause. "OpenLB: towards an efficient parallel open source library for lattice Boltzmann fluid flow simulations". In: vol. 9. PARA, International Workshop on State-of-the-Art in Scientific and Parallel Computing. accepted for. 2010.

[84] V. Heuveline, M. Krause and J. Latt. "Towards a hybrid parallelization of lattice Boltzmann methods". In: *Computers & Mathematics with Applications* (2009). DOI: 10.1016/j.camwa.2009.04.001.

[85] M. Hochbruck. *Numerische Mathematik I und II, Numerische Methoden für Differentialgleichungen, Finite Elemente Methoden*. Skriptum. June 2016.

[86] D. J. Holdych, D. R. Noble, J. G. Georgiadis and R. O. Buckius. "Truncation error analysis of lattice Boltzmann methods". In: *Journal of Computational Physics* 193.2 (2004), pp. 595–619. DOI: http://dx.doi.org/10.1016/j.jcp.2003.08.012.

[87] S. Hou, J. Sterling, S. Chen and G. Doolen. "A lattice Boltzmann subgrid model for high Reynolds number flows". In: *arXiv preprint comp-gas/9401004* (1994).

[88] H. Hu. "Direct simulation of flows of solid-liquid mixtures". In: *International Journal of Multiphase Flow* 22.2 (1996), pp. 335–352. DOI: http://dx.doi.org/10.1016/0301-9322(95)00068-2.

[89] H. Hu, D. Joseph and M. Crochet. "Direct simulation of fluid particle motions". In: *Theoretical and Computational Fluid Dynamics* 3.5 (1992), pp. 285–306. DOI: 10.1007/BF00717645.

[90] Y. Hu, H. Yuan, S. Shu, X. Niu and M. Li. "An improved momentum exchanged-based immersed boundary lattice Boltzmann method by using an iterative technique". In: *Computers & Mathematics with Applications* 68.3 (2014), pp. 140–155. DOI: 10.1016/j.camwa.2014.05.013.

[91] H. Huang, X. Yang, M. Krafczyk and X.-Y. Lu. "Rotation of spheroidal particles in Couette flows". In: *Journal of Fluid Mechanics* 692 (2012), pp. 369–394. DOI: 10.1017/jfm.2011.519.

[92] H.-B. Huang, X.-Y. Lu and M. Sukop. "Numerical study of Lattice Boltzmann methods for a convection–diffusion equation coupled with Navier–Stokes equations". In: *Journal of Physics A: Mathematical and Theoretical* 44.5 (2011), p. 055001. DOI: 10.1088/1751-8113/44/5/055001.

[93] M. A. Hyman. "Non-iterative numerical solution of boundary-value problems". In: *Applied Scientific Research, Section B* 2.1 (1952), pp. 325–351.

[94] T. Inamuro, M. Yoshina and F. Ogino. "A non-slip boundary condition for lattice Boltzmann simulations". In: *Phys. Fluids* 7 (1995), pp. 2928–2930. DOI: 10.1063/1.868766.

[95] K. Inthavong, Z. Tian, H. Li, J. Tu, W. Yang, C. Xue and C. Li. "A Numerical Study of Spray Particle Deposition in a Human Nasal Cavity". In: *Aerosol Science and Technology* 40.11 (2006), pp. 1034–1045. DOI: 10.1080/02786820600924978.

[96] K. Inthavong, Z. Tian, J. Tu, W. Yang and C. Xue. "Optimising nasal spray parameters for efficient drug delivery using computational fluid dynamics". In: *Computers in Biology and Medicine* 38.6 (2008), pp. 713–726. DOI: 10.1016/j.compbiomed.2008.03.008.

[97] S. Jafari, R. Yamamoto and M. Rahnama. "Lattice-Boltzmann method combined with smoothed-profile method for particulate suspensions". In: *Physical Review E* 83.2 (2011), p. 026702. DOI: 10.1103/PhysRevE.83.026702.

[98] V. John and E. Schmeyer. "Finite element methods for time-dependent convection–diffusion–reaction equations with small diffusion". In: *Computer Methods in Applied Mechanics and Engineering* 198.3–4 (2008), pp. 475–494. DOI: 10.1016/j.cma.2008.08.016.

[99] J. E. Jones. "On the Determination of Molecular Fields. II. From the Equation of State of a Gas". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 106.738 (1924), pp. 463–477. DOI: 10.1098/rspa.1924.0082. eprint: http://rspa.royalsocietypublishing.org/content/106/738/463.full.pdf.

[100] M. Junk and Z. Yang. "Outflow boundary conditions for the lattice Boltzmann method". In: *Progress in Computational Fluid Dynamics,* 8.1–4 (2008), pp. 38–38. DOI: 10.1504/PCFD.2008.018077.

[101] M. Junk and A. Klar. "Discretizations for the Incompressible Navier-Stokes Equations Based on the Lattice Boltzmann Method". In: *SIAM J. Sci. Comput.* 22.1 (2000), pp. 1–19. DOI: http://dx.doi.org/10.1137/S1064827599357188.

[102] A. Kaufmann, M. Moreau, O. Simonin and J. Helie. "Comparison between Lagrangian and mesoscopic Eulerian modelling approaches for inertial particles suspended in decaying isotropic turbulence". In: *Journal of Computational Physics* 227.13 (2008), pp. 6448–6472. DOI: 10.1016/j.jcp.2008.03.004.

[103] 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". In: *Aerosol Science and Technology* 38.11 (Nov. 2004), pp. 1063–1071. DOI: 10.1080/027868290883360.

[104] D. Kim and H. Choi. "Immersed boundary method for flow around an arbitrarily moving body". In: *Journal of Computational Physics* 212.2 (2006), pp. 662–680. DOI: 10.1016/j.jcp.2005.07.010.

[105] C. Kleinstreuer and Z. Zhang. "Airflow and Particle Transport in the Human Respiratory System". In: *Annual Review of Fluid Mechanics* 42.1 (2010), p. 301. DOI: 10.1146/annurev-fluid-121108-145453.

[106] C. Kleinstreuer and Z. Zhang. "An Adjustable Triple-Bifurcation Unit Model for Air-Particle Flow Simulations in Human Tracheobronchial Airways". In: *Journal of Biomechanical Engineering* 131.3 (2008), p. 021007. DOI: 10.1115/1.3005339.

[107] F. Klemens. "Simulation of Fluid-Particle Dynamics with a Porous Media Lattice Boltzmann Method". MA thesis. Karlsruher Institut für Technologie, 2016.

[108] J. Kolbe. "Simulation of the Agglomeration Process for Solid Particles in Fluids Based on Lattice Boltzmann Methods". MA thesis. Karlsruhe Institute of Technology, 2015.

[109] M. J. Krause, T. Gengenbach, S. Zimny, R. Mayer and V. Heuveline. "How to Breathe Life into CT-Data". In: *Computer Aided Medical Engineering* 4 (2011), pp. 29–33.

[110] M. J. Krause. “Fluid Flow Simulation and Optimisation with Lattice Boltzmann Methods on High Performance Computers: Application to the Human Respiratory System”. eng. PhD thesis. 2010.

[111] W. Kutta. “Beitrag zur näherungsweisen Integration totaler Differentialgleichungen”. In: *Z. Math. Phys.* 46 (1901), pp. 435–453.

[112] A. Ladd and R. Verberg. “Lattice-Boltzmann simulations of particle-fluid suspensions”. In: *Journal of Statistical Physics* 104.5-6 (2001), pp. 1191–1251.

[113] A. J. Ladd. “Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation”. In: *Journal of Fluid Mechanics* 271 (1994), pp. 285–309. DOI: 10.1017/S0022112094001771.

[114] A. J. Ladd. “Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results”. In: *Journal of Fluid Mechanics* 271 (1994), pp. 311–339. DOI: 10.1017/S0022112094001783.

[115] U. Lantermann. “Simulation der Transport- und Depositionsvorgänge von Nanopartikeln in der Gasphase mittels Partikel-Monte-Carlo- und Lattice-Botzmann-Methoden”. PhD thesis. Universität Duisburg-Essen, July 2006.

[116] J. Latt. *Choice of units in lattice Boltzmann simulations*. Apr. 2008.

[117] A. Latz and A. Wiegmann. “Simulation of fluid particle separation in realistic three dimensional fiber structures”. In: *Filtech Europa, Düsseldorf* (2003).

[118] E. Laurien and H. O. jr. *Numerische Strömungsmechanik: Grundgleichungen und Modelle - Lösungsmethoden - Qualität und Genauigkeit (German Edition)*. 5th ed. Springer Vieweg, Aug. 2013.

[119] D.-T. Lee and C. Wong. “Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees”. In: *Acta Informatica* 9.1 (1977), pp. 23–29.

[120] K. Lee and B. Liu. “Theoretical Study of Aerosol Filtration by Fibrous Filters”. In: *Aerosol Science and Technology* 1.2 (1982), pp. 147–161. DOI: 10.1080/02786828208958584.

[121] L.-S. Luo. “Analytic solutions of linearized Lattice Boltzmann equation for simple flows”. In: *Journal of Statistical Physics* 88.3-4 (Aug. 1997), pp. 913–926.

[122] H. Lv, S. Tang and W. Zhou. “Numerical simulation of sedimentation of rectangular particle in Newtonian fluid ”. In: *Particuology* 10.1 (2012), pp. 79–88. DOI: 10.1016/j.partic.2011.04.008.

[123] F. Massaioli and G. Amati. “Achieving high performance in a LBM code using OpenMP”. In: EWOMP 2002, 2002.

[124] J. Mathieu and J. Scott. *An Introduction to Turbulent Flow*. Cambridge University Press, 2000.

[125] H.-G. Matuttis and J. Chen. *Understanding the Discrete Element Method: Simulation of Non-spherical Particles for Granular and Multi-body Systems*. John Wiley & Sons, 2014.

[126] M. R. Maxey. "The equation of motion for a small rigid sphere in a nonuniform or unsteady flow". In: *ASME-PUBLICATIONS-FED* 166 (1993), pp. 57–57.

[127] M. R. Maxey and J. J. Riley. "Equation of motion for a small rigid sphere in a nonuniform flow". In: *Physics of Fluids* 26.4 (1983), pp. 883–889. DOI: http://dx.doi.org/10.1063/1.864230.

[128] G. R. McNamara and G. Zanetti. "Use of the Boltzmann Equation to Simulate Lattice-Gas Automata". In: *Phys. Rev. Lett.* 61.20 (Nov. 1988), pp. 2332–2335. DOI: 10.1103/PhysRevLett.61.2332.

[129] E. E. Michaelides. *Particles, Bubbles & Drops*. World scientific Publishing Co. Pte. Ltd., 2006.

[130] A. Mink, G. Thäter, H. Nirschl and M. J. Krause. "A 3D Lattice Boltzmann method for light simulation in participating media". In: *Journal of Computational Science* (2016).

[131] H. Mirzaee, T. Henn, M. J. Krause, L. Goubergrits, C. Schumann, M. Neugebauer, T. Kuehne, T. Preusser and A. Hennemuth. "MRI-based computational hemodynamics in patients with aortic coarctation using the lattice Boltzmann methods: Clinical validation study". In: *Journal of Magnetic Resonance Imaging* (2016). DOI: 10.1002/jmri.25366.

[132] A. A. Mohamad. *Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes*. London / New York: Springer-Verlag, 2011.

[133] A. Munjiza and K. R. F. Andrews. "NBS contact detection algorithm for bodies of similar size". In: *International Journal for Numerical Methods in Engineering* 43.1 (1998), pp. 131–149. DOI: 10.1002/(SICI)1097-0207(19980915)43:1<131::AID-NME447>3.0.CO;2-S.

[134] A. A. Munjiza. *The Combined Finite-Discrete Element Method*. 1st ed. Wiley, Apr. 2004.

[135] Y. Nakayama and R. Yamamoto. "Simulation method to resolve hydrodynamic interactions in colloidal dispersions". In: *Phys. Rev. E* 71 (3 Mar. 2005), p. 036707. DOI: 10.1103/PhysRevE.71.036707.

[136] 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.

[137] D. Nie and J. Lin. "A lattice Boltzmann-direct forcing/fictitious domain model for brownian particles in fluctuating fluids". In: *Communications in Computational Physics* 9.04 (2011), pp. 959–973. DOI: 10.4208/cicp.181109.300610a.

[138] D. Nie and J. Lin. "A LB-DF/FD method for particle suspensions". In: *Communications in Computational Physics* 7.3 (2010), p. 544. DOI: 10.4208/cicp.2009.08.155.

[139] X. Niu, C. Shu, Y. Chew and Y. Peng. "A momentum exchange-based immersed boundary-lattice Boltzmann method for simulating incompressible viscous flows". In: *Physics Letters A* 354.3 (2006), pp. 173–182. DOI: 10.1016/j.physleta.2006.01.060.

[140] D. R. Noble and J. R. Torczynski. "A Lattice-Boltzmann Method for Partially Saturated Computational Cells". In: *International Journal of Modern Physics C* 09.08 (1998), pp. 1189–1201. DOI: 10.1142/S0129183198001084.

[141] C. W. Oseen. *Hydrodynamik*. Vol. 1. Akad. Verl.-Ges., 1927.

[142] C. W. Oseen. *Über die Stoke'sche Formel und über eine verwandte Aufgabe in der Hydrodynamik*. Almqvist & Wiksell, 1911.

[143] J.-H. Park, G. von Maltzahn, L. Zhang, M. P. Schwartz, E. Ruoslahti, S. N. Bhatia and M. J. Sailor. "Magnetic iron oxide nanoworms for tumor targeting and imaging". In: *Advanced Materials* 20.9 (2008), pp. 1630–1635. DOI: 10.1002/adma.200800004.

[144] C. S. Peskin. "Flow patterns around heart valves: a numerical method". In: *Journal of computational physics* 10.2 (1972), pp. 252–271. DOI: 10.1016/0021-9991(72)90065-4.

[145] C. S. Peskin. "Numerical analysis of blood flow in the heart". In: *Journal of computational physics* 25.3 (1977), pp. 220–252. DOI: 10.1016/0021-9991(77)90100-0.

[146] C. S. Peskin. "The immersed boundary method". In: *Acta Numerica* 11 (Jan. 2002), pp. 479–517. DOI: 10.1017/S0962492902000077.

[147] G. Pingen, A. Evgrafov and K. Maute. "Topology optimization of flow domains using the lattice Boltzmann method". In: *Structural and Multidisciplinary Optimization* 34.6 (2007), pp. 507–524. DOI: 10.1007/s00158-007-0105-7.

[148] T. Pohl, F. Deserno, N. Thurey, U. Rude, P. Lammers, G. Wellein and T. Zeiser. "Performance Evaluation of Parallel Large-Scale Lattice Boltzmann Applications on Three Supercomputing Architectures". In: *Supercomputing 2004, Proceedings of the ACM/IEEE SC2004 Conference*. 2004, p. 21.

[149] Y. Qian, D. d'Humières and P. Lallemand. "Lattice BGK models for Navier-Stokes equation". In: *EPL (Europhysics Letters)* 17.6 (1992), p. 479.

[150] R. Rannacher. *Einführung in die Numerische Mathematik (Numerik 0)*. Vorlessungsskriptum SS 2005. Universität Heidelberg, 2006.

[151] R. Rannacher. *Numerische Mathematik 3 (Numerik von Problemen der Kontinuumsmechanik)*. Tech. rep. Universität Heidelberg, 2008.

[152] N. Rao and M. Faghri. "Computer Modeling of Aerosol Filtration by Fibrous Filters". In: *Aerosol Science and Technology* 8.2 (2007), pp. 133–156. DOI: 10.1080/02786828808959178.

[153] O. Reynolds. *Papers on mechanical and physical subjects*. Cambridge University Press, 1903.

[154] A. M. Roma, C. S. Peskin and M. J. Berger. "An adaptive version of the immersed boundary method". In: *Journal of computational physics* 153.2 (1999), pp. 509–534.

[155] T. Rosén, F. Lundell and C. Aidun. "Effect of fluid inertia on the dynamics and scaling of neutrally buoyant particles in shear flow". In: *Journal of Fluid Mechanics* 738 (Jan. 2014), pp. 563–590. DOI: 10.1017/jfm.2013.599.

[156] C. Runge. "Über die numerische Auflösung von Differentialgleichungen". In: *Mathematische Annalen* 46.2 (1895), pp. 167–178.

[157] M. Schäfer, S. Turek, F. Durst, E. Krause and R. Rannacher. "Benchmark Computations of Laminar Flow Around a Cylinder". English. In: *Flow Simulation with High-Performance Computers II*. Ed. by E. Hirschel. Vol. 48. Notes on Numerical Fluid Mechanics (NNFM). Vieweg+Teubner Verlag, 1996, pp. 547–566. DOI: 10.1007/978-3-322-89849-4_39.

[158] F. Schwabl. *Statistische Mechanik: mit 26 Tabellen und 186 Aufgaben*. Springer-Lehrbuch. Springer, 2000.

[159] M. Se, K. Inthavong and J. Tu. "Unsteady particle deposition in a human nasal cavity". In: *Proceedings of the Seventh International Conference on CFD in the Minerals and Process Industries*. Ed. by P. Witt and M. Schwarz. Melbourne, Australia: CSIRO, Dec. 2009, pp. 1–6.

[160] X. Shan and H. Chen. "Lattice Boltzmann model for simulating flows with multiple phases and components". In: *Physical Review E* 47.3 (1993), p. 1815. DOI: 10.1103/PhysRevE.47.1815.

[161] K. T. Shanley, P. Zamankhan, G. Ahmadi, P. K. Hopke and Y.-S. Cheng. "Numerical simulations investigating the regional and overall deposition efficiency of the human nasal cavity". In: *Inhalation toxicology* 20.12 (2008), pp. 1093–1100.

[162] X. Shi and N. Phan-Thien. "Distributed Lagrange multiplier/fictitious domain method in the framework of lattice Boltzmann method for fluid structure interactions". In: *Journal of Computational Physics* 206.1 (2005), pp. 81–94. DOI: 10.1016/j.jcp.2004.12.017.

[163] A. Sierou. "Accelerated Stokesian Dynamics: Development and application to sheared non-Brownian suspensions". PhD thesis. California Institute of Technology, 2002.

[164] P. Skordos. "Initial and boundary conditions for the Lattice Boltzmann Method". In: *Phys. Rev. E* 48(6).6 (1993), pp. 4823–4842.

[165] J. Smagorinsky. "General Circulation Experiments with the Primitive Equations". In: *Mon. Wea. Rev.* 91.3 (1963), pp. 99–164. DOI: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[166] R. van der Sman. "Galilean invariant lattice Boltzmann scheme for natural convection on square and rectangular lattices". In: *Physical Review E* 74.2 (2006), p. 026705. DOI: 10.1103/PhysRevE.74.026705.

[167] A. Sokolichin, G. Eigenberger, A. Lapin and A. Lübbert. "Dynamic numerical simulation of gas-liquid two-phase flows Euler/Euler versus Euler/Lagrange". In: *Chemical Engineering Science* 52.4 (1996), pp. 611–626. DOI: 10.1016/S0009-2509(96)00425-3.

[168] M. A. Spaid and F. R. Phelan Jr. "Lattice Boltzmann methods for modeling microscale flow in fibrous porous media". In: *Physics of Fluids (1994-present)* 9.9 (1997), pp. 2468–2474. DOI: 10.1063/1.869392.

[169] M. Stiebler, J. Tölke and M. Krafczyk. "Advection–diffusion lattice Boltzmann scheme for hierarchical grids". In: *Computers & Mathematics with Applications* 55.7 (2008). Mesoscopic Methods in Engineering and Science, pp. 1576–1584. DOI: http://dx.doi.org/10.1016/j.camwa.2007.08.024.

[170] G. G. Stokes. *On the effect of the internal friction of fluids on the motion of pendulums*. Vol. 9. Pitt Press, 1851.

[171] S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford: Clarendon Press, 2001.

[172] M. C. Sukop and D. T. Thorne. *Lattice Boltzmann modeling*. Springer, 2006.

[173] K. Suzuki and T. Inamuro. "Effect of internal mass in the simulation of a moving body by the immersed boundary method ". In: *Computers & Fluids* 49.1 (2011), pp. 173–187. DOI: 10.1016/j.compfluid.2011.05.011.

[174] T. Swaminathan, K. Mukundakrishnan and H. Hu. "Sedimentation of an ellipsoid inside an infinitely long tube at low and intermediate Reynolds numbers". In: *Journal of Fluid Mechanics* 551 (Mar. 2006), pp. 357–385. DOI: 10.1017/S0022112005008402.

[175] C.-M. Tchen. "Mean Value and Correlation Problems connected with the Motion of Small Particles suspended in a turbulent fluid". PhD thesis. TU Delft, 1947. DOI: 10.1007/978-94-017-6101-7.

[176] R. Trunk, T. Henn, W. Dörfler, H. Nirschl and M. J. Krause. "Inertial dilute particulate fluid flow simulations with an Euler–Euler lattice Boltzmann method". In: *Journal of Computational Science* (2016). DOI: http://dx.doi.org/10.1016/j.jocs.2016.03.013.

[177] T. Tsuji, K. Yabumoto and T. Tanaka. "Spontaneous structures in three-dimensional bubbling gas-fluidized bed by parallel DEM–CFD coupling simulation". In: *Powder Technology* 184.2 (2008), pp. 132–140. DOI: 10.1016/j.powtec.2007.11.042.

[178] J. Tu, K. Inthavong and G. Ahmadi. *Computational fluid and particle dynamics in the human respiratory system*. Springer, 2012.

[179] T. F. de Vasconcelos, B. Sapoval, J. S. Andrade, J. B. Grotberg, Y. Hu and M. Filoche. "Particle capture into the lung made simple?" In: *Journal of Applied Physiology* 110.6 (2011), pp. 1664–1673. DOI: 10.1152/japplphysiol.00866.2010. eprint: http://jap.physiology.org/content/110/6/1664.full.pdf.

[180] R. Verberg and A. J. C. Ladd. "Lattice-Boltzmann Model with Sub-Grid-Scale Boundary Conditions". In: *Phys. Rev. Lett.* 84 (10 Mar. 2000), pp. 2148–2151. DOI: 10.1103/PhysRevLett.84.2148.

[181] L. Verlet. "Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules". In: *Phys. Rev.* 159 (1 July 1967), pp. 98–103. DOI: 10.1103/PhysRev.159.98.

[182] L. Verlet. "Computer "Experiments" on Classical Fluids. II. Equilibrium Correlation Functions". In: *Phys. Rev.* 165 (1 Jan. 1968), pp. 201–214. DOI: 10.1103/PhysRev.165.201.

[183] E. J. W. Verwey. "Theory of the Stability of Lyophobic Colloids." In: *The Journal of Physical and Colloid Chemistry* 51.3 (1947), pp. 631–636. DOI: 10.1021/j150453a001. eprint: http://dx.doi.org/10.1021/j150453a001.

[184] K. Vollmari, R. Jasevičius and H. Kruggel-Emden. "Experimental and numerical study of fluidization and pressure drop of spherical and non-spherical particles in a model scale fluidized bed". In: *Powder Technology* 291 (2016), pp. 506–521. DOI: 10.1016/j.powtec.2015.11.045.

[185] D. Wan and S. Turek. "Direct numerical simulation of particulate flow via multigrid FEM techniques and the fictitious boundary method". In: *International Journal for Numerical Methods in Fluids* 51.5 (2006), pp. 531–566. DOI: 10.1002/fld.1129.

[186] L. Wang, Z. Guo and J. Mi. "Drafting, kissing and tumbling process of two particles with different sizes". In: *Computers & Fluids* 96 (2014), pp. 20–34. DOI: 10.1016/j.compfluid.2014.03.005.

[187] L. Wang, B. Zhang, X. Wang, W. Ge and J. Li. "Lattice Boltzmann based discrete simulation for gas–solid fluidization". In: *Chemical Engineering Science* 101 (2013), pp. 228–239. DOI: http://dx.doi.org/10.1016/j.ces.2013.06.019.

[188] E. R. Weibel. *Morphometry of the human lung*. eng. Bibliography: p. 143-148. Berlin: Springer, 1963, XI, 151 S.

[189] J.-P. Weiß. *Numerical analysis of Lattice Boltzmann Methods for the heat equation on a bounded interval*. Universitätsverlag Karlsruhe, Karlsruhe, 2006.

[190] D. A. Wolf–Gladrow. *Lattice-Gas, Cellular Automata and Lattice Boltzmann Models, An Introduction*. Lecture Notes in Mathematics. Heidelberg, Berlin: Springer, 2000.

[191] J. Wu and C. Shu. "Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications". In: *Journal of Computational Physics* 228.6 (2009), pp. 1963–1979. DOI: 10.1016/j.jcp.2008.11.019.

[192] J. Wu and C. Shu. "Particulate flow simulation via a boundary condition-enforced immersed boundary-lattice Boltzmann scheme". In: *Communications in Computational Physics* 7.4 (2010), p. 793. DOI: 10.4208/cicp.2009.09.054.

[193] M. Wünsche. "Mehrphasige Strömungssimulation mit OpenLB am Teilfahrzeugmodell". MA thesis. TU Dresden, 2015.

[194] Z. Xia, K. Connington, S. Rapaka, P. Yue, J. Feng and S. Chen. "Flow patterns in the sedimentation of an elliptical particle". In: *Journal of Fluid Mechanics* 625 (Apr. 2009), pp. 249–272. DOI: 10.1017/S0022112008005521.

[195] Q. Xiong, B. Li, J. Xu, X. Fang, X. Wang, L. Wang, X. He and W. Ge. "Efficient parallel implementation of the lattice Boltzmann method on large clusters of graphic processing units". In: *Chinese Science Bulletin* 57.7 (2012), pp. 707–715. DOI: 10.1007/s11434-011-4908-y.

[196] Q. Xiong, E. Madadi-Kandjani and G. Lorenzini. "A LBM–DEM solver for fast discrete particle simulation of particle–fluid flows". In: *Continuum Mechanics and Thermodynamics* 26.6 (2014), pp. 907–917.

[197] K. Yokoi. "Numerical Method for Interaction Among Multi-particle, Fluid and Arbitrary Shape Structure". In: *Journal of Scientific Computing* 46.2 (2011), pp. 166–181. DOI: 10.1007/s10915-010-9385-y.

[198] H. Yu, S. Girimaji and L.-S. Luo. "DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method". In: *Journal of Computational Physics* 209.2 (2005), pp. 599–616. DOI: 10.1016/j.jcp.2005.03.022.

[199] M. M. Zdravkovich. *Flow around Circular Cylinders: Volume 1: Fundamentals*. Oxford University Press, 1997.

[200] M. M. Zdravkovich. *Flow around Circular Cylinders: Volume 2: Applications*. Oxford University Press, 2003.

[201] 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: *Proceedings of 3rd Russian-German Workshop on High Performance Computing, Novosibirsk, July 2007, Springer*. Ed. by N. S. e. a. M. Resch Y. Shokin. 2008.

[202] H. Zhang, F. X. Trias, A. Oliva, D. Yang, Y. Tan, S. Shu and Y. Sheng. "PIBM: Particulate immersed boundary method for fluid–particle interaction problems". In: *Powder Technology* 272 (2015), pp. 1–13. DOI: 10.1016/j.powtec.2014.11.025.

[203] Z. Zhang, C. Kleinstreuer, J. Donohue and C. Kim. "Comparison of micro- and nano-size particle depositions in a human upper airway model". In: *Journal of Aerosol Science* 36.2 (2005), pp. 211–233. DOI: 10.1016/j.jaerosci.2004.08.006.

[204] Q. Zou and X. He. "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model". In: *Phys. Fluids* 9 (1997), pp. 1591–1598. DOI: 10.1063/1.869307.