A Viscosity Adaptive Lattice Boltzmann Method

Vom Fachbereich Maschinenbau und Verfahrenstechnik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)
genehmigte
Dissertation

von

Herrn

Dipl.-Ing. Daniel Conrad
aus Kaiserslautern

2015

Tag der mündlichen Prüfung:	23. März 2015
Dekan:	Prof. Dr.-Ing. Christian Schindler
Vorsitzender:	Prof. Dr.-Ing. Sergiy Antonyuk
Berichterstatter:	Prof. Dr.-Ing. Martin Böhle
	Prof. Dr.-Ing. habil. Uwe Janoske

D 386

In Gedenken an meinen Bruder

Christopher Conrad
(14.04.1992 - 08.07.1996)

Danksagung

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Strömungsmechanik und Strömungsmaschinen der Technischen Universität Kaiserslautern.

Zuerst möchte ich mich bei meinem Doktorvater Prof. Dr.-Ing. Martin Böhle dafür bedanken, dass er uns die größt mögliche Freiheit zum Erreichen unserer Ziele eingeräumt hat. Mit seinem Vertrauen in unser Lattice Boltzmann Team hat er mir gezeigt was Forschung bedeutet.

Ferner danke ich Herrn Prof. Dr.-Ing. habil. Uwe Janoske für die Übernahme der Zweitkorrektur sowie Herrn Prof. Dr.-Ing. Sergiy Antonyuk für den Vorsitz der Prüfungskommission.

Den gesamten Mitarbeitern des SAM Lehrstuhls danke ich für die gute Zusammenarbeit und die Kollegialität sowohl innerhalb als auch außerhalb des Institutes.

Mein Dank gilt auch dem Regionalen Hochschulrechenzentrum Kaiserslautern für die infrastrukturelle und fachliche Unterstützung. Namentlich möchte ich hier speziell Herrn Dr.-Ing. Markus Hillenbrand und Herrn PD Dr. habil. Josef Schüle erwähnen.

Tiefer Dank gilt meiner Oma Erna und meiner Mutter Angelika, die nie an mir gezweifelt und mich in jeder Situation unterstützt haben. Insbesondere möchte ich mich bei meiner Mutter dafür bedanken, dass sie mich schon immer moralisch auf meinem Weg begleitet hat.

Bei meinem SAM-Lattice Mitentwickler, Kollegen und Freund Andreas Schneider möchte ich mich für die gemeinsamen Jahre herzlich bedanken. Ich hätte mir keinen besseren Mitstreiter wünschen können.

Letztlich danke ich meiner Frau Lena von ganzen Herzen für die vielen Entbehrungen, ihre uneingeschränkte Unterstützung und Liebe. Auch möchte ich mich bei unserem Sohn Johann bedanken, der mein Leben so sehr bereichert.

Kaiserslautern, im April 2015. Daniel Conrad

Contents

Abstract

The present thesis describes the development and validation of a viscosity adaption method for the numerical simulation of non-Newtonian fluids on the basis of the Lattice Boltzmann Method (LBM), as well as the development and verification of the related software bundle *SAM-Lattice*.

By now, Lattice Boltzmann Methods are established as an alternative approach to classical computational fluid dynamics methods. The LBM has been shown to be an accurate and efficient tool for the numerical simulation of weakly compressible or incompressible fluids. Fields of application reach from turbulent simulations through thermal problems to acoustic calculations among others. The transient nature of the method and the need for a regular grid based, non body conformal discretization makes the LBM ideally suitable for simulations involving complex solids. Such geometries are common, for instance, in the food processing industry, where fluids are mixed by static mixers or agitators. Those fluid flows are often laminar and non-Newtonian.

This work is motivated by the immense practical use of the Lattice Boltzmann Method, which is limited due to stability issues. The stability of the method is mainly influenced by the discretization and the viscosity of the fluid. Thus, simulations of non-Newtonian fluids, whose kinematic viscosity depend on the shear rate, are problematic. Several authors have shown that the LBM is capable of simulating those fluids. However, the vast majority of the simulations in the literature are carried out for simple geometries and/or moderate shear rates, where the LBM is still stable. Special care has to be taken for practical non-Newtonian Lattice Boltzmann simulations in order to keep them stable. A straightforward way is to truncate the modeled viscosity range by numerical stability criteria. This is an effective approach, but from the physical point of view the viscosity bounds are chosen arbitrarily. Moreover, these bounds depend on and vary with the grid and time step size and, therefore, with the simulation Mach number, which is freely chosen at the start of the simulation. Consequently, the modeled viscosity range may not fit to the actual range of the physical problem, because the correct simulation Mach number is unknown *a priori*. A way around is, to perform precursor simulations on a fixed grid to determine a possible time step size and simulation Mach number, respectively. These precursor simulations can be time consuming and expensive, especially for complex cases

and a number of operating points. This makes the LBM unattractive for use in practical simulations of non-Newtonian fluids.

The essential novelty of the method, developed in the course of this thesis, is that the numerically modeled viscosity range is consistently adapted to the actual physically exhibited viscosity range through change of the simulation time step and the simulation Mach number, respectively, while the simulation is running. The algorithm is robust, independent of the Mach number the simulation was started with, and applicable for stationary flows as well as transient flows. The method for the viscosity adaption will be referred to as the "'viscosity adaption method (VAM)"' and the combination with LBM leads to the "'viscosity adaptive LBM (VALBM)"'.

Besides the introduction of the VALBM, a goal of this thesis is to offer assistance in the spirit of a theory guide to students and assistant researchers concerning the theory of the Lattice Boltzmann Method and its implementation in *SAM-Lattice*. In Chapter 2, the mathematical foundation of the LBM is given and the route from the BGK approximation of the Boltzmann equation to the Lattice Boltzmann (BGK) equation is delineated in detail. The derivation is restricted to isothermal flows only. Restrictions of the method, such as low Mach number flows are highlighted and the accuracy of the method is discussed.

SAM-Lattice is a C++ software bundle developed by the author and his colleague Dipl.-Ing. Andreas Schneider. It is a highly automated package for the simulation of isothermal flows of incompressible or weakly compressible fluids in 3D on the basis of the Lattice Boltzmann Method. By the time of writing of this thesis, *SAM-Lattice* comprises 5 components. The main components are the highly automated lattice generator *SamGenerator* and the Lattice Boltzmann solver *SamSolver*. Postprocessing is done with *ParaSam*, which is our extension of the open source visualization software *ParaView*. Additionally, domain decomposition for MPI parallelism is done by *SamDecomposer*, which makes use of the graph partitioning library MeTiS. Finally, all mentioned components can be controlled through a user friendly GUI (*SamLattice*) implemented by the author using QT, including features to visually track output data. In Chapter 3, some fundamental aspects on the implementation of the main components, including the corresponding flow charts will be discussed. Actual details on the implementation are given in the comprehensive programmers guides to *SamGenerator* and *SamSolver*.

In order to ensure the functionality of the implementation of *SamSolver*, the solver is verified in Chapter 4 for Stokes's First Problem, the suddenly accelerated plate, and for Stokes's Second Problem, the oscillating plate, both for Newtonian fluids. Non-Newtonian fluids are modeled in *SamSolver* with the power-law model according to Ostwald de Waele. The implementation for non-Newtonian fluids is verified for the Hagen-Poiseuille channel flow in conjunction with a convergence analysis of the method. At the same time, the local grid refinement as it is implemented in *SamSolver*, is verified. Finally, the verification of higher order boundary conditions is done for the 3D Hagen-Poiseuille pipe flow for both Newtonian and non-Newtonian fluids.

In Chapter 5, the theory of the viscosity adaption method is introduced. For the adaption process, a target collision frequency or target simulation Mach number must be chosen and the distributions must be rescaled according to the modified time step size. A convenient choice is one of the stability bounds. The time step size for the adaption step is deduced from the target collision frequency Ω_t and the currently minimal or maximal shear rate in the system, while obeying auxiliary conditions for the simulation Mach number. The adaption is done in the collision step of the Lattice Boltzmann algorithm. We use the transformation matrices of the MRT model to map from distribution space to moment space and vice versa. The actual scaling of the distributions is conducted on the back mapping, because we use the transformation matrix on the basis of the new adaption time step size. It follows an additional rescaling of the non-equilibrium part of the distributions, because of the form of the definition for the discrete stress tensor in the LBM context. For that reason it is clear, that the VAM is applicable for the SRT model as well as the MRT model, where there is virtually no extra cost in the latter case. Also, in Chapter 5, the multi level treatment will be discussed.

Depending on the target collision frequency and the target Mach number, the VAM can be used to optimally use the viscosity range that can be modeled within the stability bounds or it can be used to drastically accelerate the simulation. This is shown in Chapter 6. The viscosity adaptive LBM is verified in the stationary case for the Hagen-Poiseuille channel flow and in the transient case for the Wormersley flow, i.e., the pulsatile 3D Hagen-Poiseuille pipe flow. Although, the VAM is used here for fluids that can be modeled with the power-law approach, the implementation of the VALBM is straightforward for other non-Newtonian models, e.g., the Carreau-Yasuda or Cross model. In the same chapter, the VALBM is validated for the case of a propeller viscosimeter developed at the chair SAM. To this end, the experimental data of the torque on the impeller of three shear thinning non-Newtonian liquids serve for the validation. The VALBM shows excellent agreement with experimental data for all of the investigated fluids and in every operating point. For reasons of comparison, a series of standard LBM simulations is carried out with different simulation Mach numbers, which partly show errors of several hundred percent. Moreover, in Chapter 7, a sensitivity analysis on the parameters used within the VAM is conducted for the simulation of the propeller viscosimeter.

Finally, the accuracy of non-Newtonian Lattice Boltzmann simulations with the SRT and the MRT model is analyzed in detail. Previous work for Newtonian fluids indicate that depending on the numerical value of the collision frequency Ω, additional artificial viscosity is introduced due to the finite difference scheme, which negatively influences the accuracy. For the non-Newtonian case, an error estimate in the form of a functional is derived on the basis of a series expansion of the Lattice Boltzmann equation. This functional can be solved analytically for the case of the Hagen-Poiseuille channel flow of non-Newtonian fluids. The estimation of the error minimum is excellent in regions where the Ω error is the dominant source of error as opposed to the compressibility error.

Result of this dissertation is a verified and validated software bundle on the basis of the viscosity adaptive Lattice Boltzmann Method. The work restricts itself on the simulation of isothermal, laminar flows with small Mach numbers. As further research goals, the testing of the VALBM with minimal error estimate and the investigation of the VALBM in the case of turbulent flows is suggested.

Kurzfassung auf Deutsch

Die vorliegende Dissertation beschreibt die Entwicklung und Validierung eines Viskositätsadaptierungsverfahrens zur numerischen Strömungssimulation von nicht-Newtonschen Fluiden auf Basis der Lattice Boltzmann Methode (LBM), sowie die Entwicklung und Verifizierung des dazugehörigen Softwarepaketes *SAM-Lattice*. Die Lattice Boltzmann Methode hat sich als Alternative zu klassischen Methoden der numerischen Strömungsmechanik zur Simulation von schwach kompressiblen und inkompressiblen Fluiden etabliert. Sie wird bereits in vielen technischen Bereichen eingesetzt. Diese reichen unter anderem von turbulenten Strömungen, über thermischen Problemstellungen, bis hin zur Berechnung akustischer Wellen. Insbesondere die transiente Natur des Verfahrens und die bei der LBM notwendige, nicht randkonforme Diskretisierung mit strukturierten Gittern eignet sich hervorragend zur Simulation von Strömungen in komplexen Geometrien. Solche Geometrien sind beispielsweise in der Nahrungsmittelindustrie vorzufinden, wo Flüssigkeiten mit statischen Mischern oder Rührern bearbeitet werden. Solche Strömungen sind häufig laminar und die Flüssigkeiten weisen nicht-Newtonsches Verhalten auf.

Diese Arbeit ist durch den großen praktischen Nutzen dieser Vorteile der Lattice Boltzmann Methode motiviert, deren Einschränkung jedoch die numerische Stabilität darstellt. Diese hängt im Wesentlichen von der Diskretisierung und der Viskosität des Fluides ab und ist daher für Simulationen nicht-Newtonscher Fluide, deren kinematische Viskosität von der Scherrate abhängt, problematisch. Mehrere Autoren haben bereits gezeigt, dass die Simulation von nicht-Newtonschen Fluiden mit der Lattice Boltzmann Methode möglich ist. Die überwiegende Mehrheit der in der Literatur vorzufindenden nicht-Newtonschen LBM Simulationen finden jedoch in einfachen Geometrien und/oder in einem moderaten Scherratenbereich statt so, dass die Lattice Boltzmann Methode stabil bleibt. Für praktische nicht-Newtonsche LBM Simulation ist es zwingend notwendig den modellierten Viskositätsbereich des simulierten Fluides nach Stabilitätskriterien einzuschränken, um die numerische Stabilität zu gewährleisten. Dies ist ein effektiver Ansatz, jedoch werden die Viskositätsgrenzen aus physikalischer Sicht willkürlich gewählt. Außerdem ändern sich diese Grenzen mit der Gitterweite Δx und der Zeitschrittweite Δt und damit mit

der Simulations-Machzahl, welche vor der Simulation frei gewählt wird. Die Konsequenz daraus ist, dass der simulierte Viskositätsbereich nicht mit dem tatsächlich physikalisch auftretenden Viskositätsbereich zusammenfällt, da eine angepasste Simulations-Machzahl *a priori* unbekannt ist. Eine geeignete Simulations-Machzahl kann durch eine Reihe von Vorlaufsimulationen bei fester Gitterweite abgeschätzt werden. Dies ist aber mithin sehr zeit- und kostenintensiv, insbesondere für komplexe Geometrien und mehreren Betriebspunkten. Dieser Umstand macht die Lattice Boltzmann Methode unattraktiv für Simulationen praktischer Problemstellungen.

Wesentliche Neuheit des in dieser Dissertation entwickelten Verfahrens ist, dass der numerisch modellierte Viskositätsbereich konsistent durch Änderung des Zeitschrittes bzw. der globalen Simulations-Machzahl während der Simulation an den physikalisch auftretenden Viskositätsbereich adaptiert wird. Der Algorithmus ist robust, unabhängig von der Ausgangs-Machzahl beim Start der Simulation und sowohl für stationäre als auch für transiente Strömungen einsetzbar. Die Methode zur Viskositätsanpassung wird als „viscosity adaption method (VAM)“ bezeichnet und die Kombination mit LBM führt auf die „viscosity adaptive Lattice Boltzmann Method (VALBM)“.

Neben der Einführung der VALBM, ist es die Aufgabe der vorliegenden Arbeit, Studenten und wissenschaftlichen Mitarbeitern eine Hilfestellung zur Theorie der Lattice Boltzmann Methode und deren Implementierung in *SAM-Lattice* zu bieten. In Kapitel 2 werden die mathematischen Grundlagen der LBM und der Weg von der BGK-Approximation der Boltzmann Gleichung zur Lattice Boltzmann (BGK) Gleichung ausführlich beschrieben. Die Herleitung beschränkt sich auf isotherme Strömungen. Einschränkungen der Methode, wie die Voraussetzung kleiner Machzahlen werden aufgezeigt und die Genauigkeit der Methode diskutiert.

SAM-Lattice ist ein C++ Softwarepaket, dass von dem Autor und seinem Kollegen Dipl.-Ing Andreas Schneider entwickelt wurde. Zum Zeitpunkt des Verfassens dieser Arbeit umfasst *SAM-Lattice* 5 Komponenten. Die Hauptkomponenten sind der stark automatisierte Gittergenerator *SamGenerator* und der Lattice Boltzmann Löser *SamSolver*. Die Verarbeitung der Simulationsergebnisse geschieht in *ParaSam*, unserer Erweiterung der open-source Software *ParaView*. Zusätzlich steht das Programm *SamDecomposer* zur Verfügung, welches mit Hilfe der Graph-Partitionierungsbibliothek MeTiS, die Gebietszerlegung für eine MPI Parallelisierung durchführt. Schließlich können die zuvor genannten Komponenten mit einer benutzerfreundlichen GUI (*SamLattice*) gesteuert werden, welche von dem Autor mittels QT implementiert wurde und die visuelle Verfolgung der Simulationsergebnisse gestattet. In Kapitel 3 werden neben den Ablaufdiagrammen beider Hauptkomponenten fundamentale Aspekte der Implementierung des Gittergenerators und des Lösers aufgezeigt. Die eigentlichen Implementierungsdetails sind den umfangreichen Programmierdokumentationen zu *SamGenerator* und *SamSolver* zu entnehmen.

Um die Funktionalität der Implementierung von *SamSolver* sicher zu stellen, wird der Löser in Kapitel 4 anhand des Ersten Stokes'schen Problems, der plötzlich in Gang

gesetzten Platte und des Zweiten Stokes'schen Problems, der oszillierenden Platte, für Newtonsche Fluide verifiziert. Nicht-Newtonsche Fluide werden in *SamSolver* mit dem Potenzansatz nach Ostwald de Waele modelliert. Die Implementierung für nicht-Newtonsche Fluide wird am Beispiel der Hagen-Poiseuille Kanalströmung in Verbindung mit einer Konvergenzstudie des Verfahrens verifiziert. Gleichzeitig wird die in *SamSolver* implementierte lokale Gitterverfeinerung verifiziert. Es folgt schließlich eine Verifizierung von Randbedingungen höherer Ordnung am Beispiel der 3D Hagen-Poiseuille Rohrströmung, sowohl für Newtonsche als auch für nicht-Newtonsche Fluide.

In Kapitel 5 wird die Theorie zur *viscosity adaption method* vorgestellt. Zur Adaptierung muss eine Ziel-Kollisionsfrequenz bzw. Ziel-Machzahl gewählt und die Verteilungsfunktionen gemäß der veränderten Zeitschrittweite skaliert werden. Zweckmäßig ist die Wahl einer der Stabilitätsgrenzen. Die Zeitschrittweite für einen Adaptierungsschritt wird aus der Zielkollisionsfrequenz Ω_t und der aktuellen maximalen bzw. minimalen Scherrate im System unter Einhaltung von Nebenbedingungen an die Simulations-Machzahl bestimmt. Die Adaptierung geschieht im Kollisionsschritt des Lattice Boltzmann Algorithmus. Dabei bedienen wir uns der Abbildungsmatrizen des MRT Modells für die Transformation aus dem Verteilungsraum in den Momentenraum und umgekehrt. Die eigentliche Skalierung der Verteilungen geschieht bei der Rücktransformation aus dem Momentenraum zurück in den Verteilungsraum, da hierfür die Transformationsmatrix auf der Grundlage des neuen Adaptionszeitschrittes benutzt wird. Es schließt sich eine zusätzliche Skalierung des Nichtgleichgewichtsanteils der Verteilungen aufgrund der Definition des diskreten Spannungstensors im LBM Kontext an. Aus diesem Grund ist es klar, dass die VAM sowohl für das SRT, also auch für das MRT Model anwendbar ist, wobei für letzteres nahezu kein Mehraufwand besteht. Es wird in Kapitel 5 auch auf die Vorgehensweise bei lokaler Gitterverfeinerung eingegangen.

Abhängig von der Zielkollisionsfrequenz bzw. der Ziel-Machzahl kann die VAM dazu genutzt werden, den in den Stabilitätsgrenzen modellierbaren Viskositätsbereich optimal auszunutzen oder die Simulation drastisch zu beschleunigen. Dies wird in Kapitel 6 gezeigt. Die *viscosity adaptive LBM* wird für den stationären Fall mit der Hagen-Poiseuille Kanalströmung verifiziert. Zur Verifizierung im transienten Fall dient eine Womersley Strömung, d.h. eine pulsierende 3D Hagen-Poiseuille Rohrströmung. Obwohl die VAM hier für Fluide eingesetzt wird, die mit dem Potenzansatz beschrieben werden können, ist die Übertragung auf andere nicht-Newtonsche Modelle, wie dem Carreau-Yasuda oder dem Cross Modell, unkompliziert. Im selben Kapitel wird die VALBM für den komplexen Fall eines am SAM entwickelten Propellerviskosimeters validiert. Hierzu dienen Messdaten des Drehmomentverlaufes für 3 verschiedene scherverdünnende nicht-Newtonsche Flüssigkeiten. Die VALBM zeigt für alle Simulationen der untersuchten Fluide in allen Betriebspunkten exzellente Übereinstimmungen mit den Messdaten.

Zum Vergleich werden eine Reihe von Standard LBM Simulationen mit verschiedenen Machzahlen vorgenommen, welche teilweise Fehler von mehreren hundert Prozent aufweisen.

Weiter wird in Kapitel 7 eine Sensitivitätsanalyse der in der VAM verwendeten Parameter für den Fall des Propellerviskosimeters durchgeführt.

Schließlich wird die Genauigkeit von nicht-Newonschen Lattice Boltzmann Simulationen sowohl für das SRT Modell also auch für das MRT Modell detailliert analysiert. Vorarbeiten für Newtonsche Fluide zeigen, dass abhängig von dem numerischen Wert der Kollisionsfrequenz Ω zusätzliche künstliche Viskosität aufgrund des Finite Differenzen Schemas eingeführt wird, was negativen Einfluss auf die Genauigkeit des Verfahrens hat. Für den nicht-Newtonschen Fall wird hier auf Basis einer Reihenentwicklung der Lattice Boltzmann Gleichung eine Abschätzung für diesen Fehler in Form eines Funktionales hergeleitet. Für die Hagen-Poiseuille Kanalströmung nicht-Newtonscher Fluide kann das Funktional analytisch gelöst werden. Die Vorhersage des Fehlerminimums ist exzellent in Bereichen bei denen der Ω Fehler gegenüber dem Kompressibilitätsfehler dominiert.

Ergebnis der Dissertation ist ein verifiziertes und validiertes Softwarepaket auf Basis der viscosity adaptive Lattice Boltzmann Method. Dabei beschränkt sich die Arbeit auf isotherme, laminare Strömungen kleiner Machzahlen. Als weiterführende Forschungsziele werden die Erprobung der VALBM mit minimalen Fehlerschätzer und die Untersuchung der VALBM für turbulente Strömungen vorgeschlagen.

CHAPTER 1

Introduction

At the Institute of Fluid Mechanics and Fluid Machinery (SAM) at TU Kaiserslautern, the wish emerged to start research in the field of computational fluid dynamics on the basis of the Lattice Boltzmann Method. To put this into practice, it was necessary to create a software framework from scratch. In the course of this thesis, such a framework with the name *SAM-Lattice* accrued.

SAM-Lattice

SAM-Lattice is a C++ software bundle developed by the author and his colleague Dipl.-Ing. Andreas Schneider. It is a highly automated package for the simulation of isothermal flows of incompressible or weakly compressible fluids in 3D on the basis of the Lattice Boltzmann Method. By the time of writing of this thesis, *SAM-Lattice* comprises five components. The main components are the lattice generator *SamGenerator* and the Lattice Boltzmann solver *SamSolver*. Some details on the implementation of these are given in Chapter 3. Postprocessing is done with *ParaSam*, which is an extension of the open source visualization software ParaView. Additionally, domain decomposition for MPI parallelism is done by *SamDecomposer*, which makes use of the MeTiS library. Finally, all mentioned components can be controlled through a user friendly GUI (*SamLattice*) implemented by the author using QT, including features to visually track output data.

Scope of this Work

Although capabilities for the computation of turbulent fluid flows are implemented in *SAM-Lattice*, this thesis restricts itself to the investigation of laminar flows. Central result of this thesis is an algorithm termed Viscosity Adaption Method (VAM) for generalized Newtonian flows. It enables the simulation to automatically and consistently adjust the modeled viscosity range to the physical range of the system, which will be shown in Chapter 5. This ultimately results in a Viscosity Adaptive Lattice Boltzmann Method (VALBM).

In the spirit of a theory guide, a goal of this thesis is to offer assistance to students and assistant researchers working on *SAM-Lattice*. Unlike most implementations of the LBM, we do not rely on a dimensionless representation of the internal variables. Therefore, results from the literature are often not directly applicable. Although this approach may seem unconventional at heart, it has didactic benefits. Internal variables can directly be interpreted with their physical meaning and, therefore, be checked for validity already in the debugging process. Formulas can be checked more intuitively through dimensional analysis. Especially time step sizes and grid sizes are chosen to be unity in a dimensionless formulation. It is common practice in the literature to replace such quantities with their scalar value. Once this is done, the information that this scalar has a physical meaning is lost in the plain formula. Hence, it is important to understand the derivation of the individual correlations frequently used in the Lattice Boltzmann community.

Outline

Chapter 2 is intended to delineate the route from the BGK approximation of the Boltzmann equation to the Lattice Boltzmann (BGK) equation. The derivation is restricted to isothermal flows only. Restrictions of the method such as low Mach number flows are highlighted and the accuracy of the method is discussed. There exists a comprehensive programmers documentation for *SamGenerator* and *SamSolver*. Therefore, in Chapter 3, some fundamental aspects of the corresponding algorithms used for the implementation of the lattice generator and LBM solver are outlined. Verification of the code is done in Chapter 4 for both Newtonian and non-Newtonian fluids. In Chapter 5, the Viscosity Adaption Method, the basis of the VALBM, is presented. The proposed method is verified in Chapter 6 for simple steady state and transient cases as well as validated with a complex application example: A Propeller viscosimeter operating in a shear thinning fluid. Chapter 7 focuses on the accuracy of non-Newtonian LBM simulations with the aim to further improve simulation results through an ideal choice of VAM parameters. The thesis is summarized in Chapter 8 and some perspectives are discussed.

CHAPTER 2

Route to the Lattice Boltzmann Equation

2.1 Preliminaries

2.1.1 Tensor Notation and Calculus

In the style of [40], it is useful to define some frequently used tensor operations along with their notation used throughout this thesis. Vectors will be denoted by bold lower case letters $\mathbf{a}$ and higher order Tensors by bold capital letters $\mathbf{A}$. The corresponding index notation will appear in normal type with indices in lower case Greek letters. Accordingly, a plain scalar quantity is written in normal type lower case letters. Preferably, the tensor notation is used, but to achieve lucidity, we will switch to index notation where appropriate. In conjunction with index notation, the Einstein summation convention is employed. This means if a single term contains the same index repeatedly the sum over that index is implied without explicitly writing a summation sign. To distinguish operations involving the ∇ operator, the divergence operation is denoted by a dot product in contrast to the gradient operation. Furthermore, it is common practice in the Lattice Boltzmann community to denote partial derivatives by an index such that spatial derivatives become $\partial/\partial x_\alpha = \partial_\alpha$ and temporal derivatives become $\partial/\partial t = \partial_t$. The following Table 2.1 illustrates some common operations.

2.1.2 Landau Notation

For the sake of formality, a brief description of the Landau notation is given. Landau symbols are used here to express asymptotic behavior of a function when their argument tends to zero. This is commonly used to describe the residual error due to discretization or

Table 2.1: Tensor Operations

Operation	Tensor notation	Index notation
Scalar product	$c = \mathbf{a} \cdot \mathbf{b}$	$c = a_\alpha b_\alpha$
Tensor contraction	$c = \mathbf{A} : \mathbf{B}$	$c = A_{\alpha\beta} B_{\alpha\beta}$
Dyadic product	$\mathbf{A} = \mathbf{a} \otimes \mathbf{b} = \mathbf{ab}$	$A_{\alpha\beta} = a_\alpha b_\beta$
Gradient (Scalar field)	$\mathbf{a} = \nabla c$	$a_\alpha = \partial_\alpha c$
Divergence of 1D Tensor (Vector)	$c = \nabla \cdot \mathbf{a}$	$c = \partial_\alpha a_\alpha$
Divergence of 2D Tensor (Matrix)	$\mathbf{a} = \nabla \cdot \mathbf{A}$	$a_\alpha = \partial_\beta A_{\alpha\beta}$
Divergence of 3D Tensor	$\mathbf{A} = \nabla \cdot \mathbf{T}$	$A_{\alpha\beta} = \partial_\gamma T_{\alpha\beta\gamma}$

truncation of an infinite expansion. The residual error is expressed in terms of a simpler upper bound function representing the most significant term of the residuum. Table 2.2 illustrates some examples with the application of the two Landau-symbols used in this thesis in the context of $\lim_{\Delta x \to 0}$.

Table 2.2: Landau Symbols

Notation	Meaning
$f(x) \in \mathcal{O}(1)$	$f(x)$ is limited
$f(x) \in \mathcal{O}(\Delta x)$	$f(x)$ falls linearly
$f(x) \in \mathcal{O}(\Delta x^2)$	$f(x)$ falls quadratically
$f(x) \in \Theta(\Delta x)$	$f(x)$ falls as fast as Δx
$f(x) \sim \Delta x$	$f(x)$ is on the order of Δx

Consider the example: $f(x) = g(x) + \mathcal{O}(\Delta x^4)$, where $g(x)$ is an approximation for $f(x)$. This means that $|f(x) - g(x)| \leq |c \cdot \Delta x^4|$ for some constant c. In other words: The residual error tends to zero at least as fast as Δx^4, as Δx tends to zero. The order of accuracy is given by the order of the leading error term. So, the approximation $g(x)$ for $f(x)$ is forth order accurate in the example above. Although it is mathematically incorrect, the equal sign is often used in conjunction with the Landau notation. It is understood that such statements are meant symbolically. In particular, the statement $E = \mathcal{O}(1)$ means that the error E does not fall below a certain constant value. By contrast, the truncation order is given by the order of the last term used in the approximation. Hence, if $g(x)$ was a series expansion of a polynomial in x, the truncation order would be three in the aforementioned example.

2.2 The Boltzmann Equation

The Boltzmann equation expresses the balance of molecules with respect to an infinitesimal volume in phase space. The 6-dimensional phase space is the union of the physical space expressed by space vector $\mathbf{x}$ and the space of absolute molecular velocities expressed by the velocity vector $\boldsymbol{\xi}$. In this context, the molecules are mathematically captured through a probability density function $f(\mathbf{x}, \boldsymbol{\xi}, t)$. At a fixed time t, this function gives the probability to find a molecule moving with velocity $\boldsymbol{\xi} + d\boldsymbol{\xi}$ at a place $\mathbf{x} + d\mathbf{x}$. Hereafter, the probability density function will synonymously be called probability distribution or simply distribution and its arguments will be dropped unless it is useful for comprehensibility.

Our starting point of the route to the Lattice Boltzmann equation is the BGK approximation of the Boltzmann equation without external forces [28, 59, 61]:

$$\partial_t f + \boldsymbol{\xi} \cdot \nabla f = \frac{1}{\tau}(f^{eq} - f). \tag{2.1}$$

As already stated, the Boltzmann equation balances molecules in an infinitesimal volume in phase space. The left hand side is the material derivative of f describing the rate of change of the probability distribution through transport (advection) of the molecules. Molecules may also change their velocity as a consequence of inter-molecular collision and, therefore, drop out of or enter the infinitesimal volume in phase space. Thus, the right hand side of the equation expresses the rate of change of f through collision of the molecules.

In the above equation (2.1), the original mathematically complex operator describing inter-molecular collisions is already replaced by a much simpler collision operator. From an engineer's point of view, collisions between particular molecules are of no importance. What matters is their long term effect on the macroscopic quantities. By experience, we know that a gas in non-equilibrium state relaxes towards its equilibrium state after a fair amount of collisions took place. So, we substitute the original collision operator with the relaxation of the distribution towards its equilibrium expressed by f^{eq} with a characteristic collision time τ where $1/\tau$ is called collision frequency. The time needed to reach equilibrium is different for different kinds of gases. As it will turn out, the collision time must be connected with the viscosity of the gas. This model is known as the BGK approximation of the collision operator due to the originators Bhatnagar, Gross, and Kroog [3] and is the basis of the majority of LBM models used. The BGK collision operator conserves mass, momentum, and energy. As stated in the introduction we will restrict ourselves to isothermal and weakly compressible flows for which the temperature is constant and the energy conservation will not be considered in the sequel.

2.2.1 Equilibrium Distribution

The equilibrium state is characterized by vanishing gradients of macroscopic quantities. Transferred to the microscopic view of a gas in the continuum limit, the equilibrium

state is given if the collision integral vanishes. That is, if for every molecule leaving the infinitesimal volume in phase space, an inverse collision can be found for which an other molecule enters it. When gradients vanish, it is perspicuous that the probability distribution should be spherically symmetric. The equilibrium distribution is given by the well known Maxwell-Boltzmann distribution [24, 28],

$$f^{eq} = \frac{\rho}{(2\pi RT)^{3/2}} \cdot \exp\left(-\frac{(\boldsymbol{\xi} - \mathbf{u})^2}{2RT}\right), \tag{2.2}$$

where R is the specific gas constant, T the Temperature, $\mathbf{u}$ the macroscopic flow velocity, and ρ the density.

2.2.2 Moments of the Continuous Distribution

The link between the microscopic state and the macroscopic variables is established through the calculation of the moments m of the distribution function. A moment is generally defined as the integral of the product of a function $a(\boldsymbol{\xi})_k$ with the probability distribution f over $\boldsymbol{\xi}$

$$m = \int_{\boldsymbol{\xi}} a(\boldsymbol{\xi})_k \cdot f \, \mathrm{d}\boldsymbol{\xi}. \tag{2.3}$$

We will define moments up to order k by which the order is determined through the order of the tensor product in the function $a(\boldsymbol{\xi})_k$. From this definition, an arbitrary number of moments can be defined at which only low order moments can be assigned a physical meaning. For the purpose of the simulation of isothermal and weakly compressible fluid flow, we define moments up to order three. Moments in equilibrium state m^{eq} are build from the equilibrium distribution eq. (2.2) for which the integrals give [11, 28, 59]

$$\rho = \int f^{eq} \, \mathrm{d}\boldsymbol{\xi} \tag{2.4}$$

$$\rho\mathbf{u} = \int \boldsymbol{\xi} f^{eq} \, \mathrm{d}\boldsymbol{\xi} \tag{2.5}$$

$$\mathbf{\Pi^{eq}} = \int \boldsymbol{\xi}\boldsymbol{\xi} f^{eq} \, \mathrm{d}\boldsymbol{\xi} = p\mathbf{I} + \rho\mathbf{u}\mathbf{u} \tag{2.6}$$

$$\mathbf{Q^{eq}} = \int \xi_\alpha \xi_\beta \xi_\gamma f^{eq} \, \mathrm{d}\boldsymbol{\xi} = \rho u_\alpha u_\beta u_\gamma + p(u_\alpha \delta_{\beta\gamma} + u_\beta \delta_{\alpha\gamma} + u_\gamma \delta_{\alpha\beta}), \tag{2.7}$$

where p is the thermodynamic pressure . The quantities mass and momentum do not contain non-equilibrium contributions $f^{neq} = f - f^{eq}$ and are, thus, conserved during collision in eq. (2.1),

$$\int f - f^{eq} \, \mathrm{d}\boldsymbol{\xi} = \int f^{neq} \, \mathrm{d}\boldsymbol{\xi} = 0 \tag{2.8}$$

$$\int \boldsymbol{\xi}(f - f^{eq}) \, \mathrm{d}\boldsymbol{\xi} = \int \boldsymbol{\xi} f^{neq} \, \mathrm{d}\boldsymbol{\xi} = 0.$$

We will call this the *conservation condition* or, more generally, *solvability condition* following the terminology in the literature. Thus, with eq. (2.8) we get the following moments up to the second order from the complete distribution,

$$\begin{aligned}\rho &= \int f \,\mathrm{d}\boldsymbol{\xi} \\ \rho\mathbf{u} &= \int \boldsymbol{\xi} f \,\mathrm{d}\boldsymbol{\xi} \\ \boldsymbol{\Pi} &= \int \boldsymbol{\xi}\boldsymbol{\xi} f \,\mathrm{d}\boldsymbol{\xi} = p\mathbf{I} + \rho\mathbf{u}\mathbf{u} - \boldsymbol{\sigma}.\end{aligned} \tag{2.9}$$

Here, the stress tensor $\boldsymbol{\sigma}$ arises from the non-equilibrium part of the distribution.

2.3 The Discrete Boltzmann Equation

In order to end up with a method that can be implemented in a computer code, the phase space has to be discretized. For reasons of structuring, the result of this step will be given in advance. The discrete Boltzmann equation in contrast to its continuous counterpart reads

$$\partial_t f_i + \boldsymbol{\xi}_i \cdot \nabla f_i = \frac{1}{\tau}(f_i^{eq} - f_i). \tag{2.10}$$

This section is intended to present the basis for the discretization of the continuous Boltzmann equation. To this end, we need to introduce a few correlations and definitions. It is well known that the speed of sound c_s of a gas is temperature dependent and can be calculated using the specific gas constant and the isotropic exponent κ,

$$c_s = \sqrt{\kappa R T}. \tag{2.11}$$

In the context of LBM, κ is simply chosen to be unity. Furthermore, the Mach number Ma is introduced, which is the ratio of a characteristic flow speed U to the speed of sound,

$$Ma = \frac{U}{c_s}. \tag{2.12}$$

Throughout this thesis, we will assume that the macroscopic flow velocity is of the order of the characteristic flow speed,

$$|\mathbf{u}| \sim U. \tag{2.13}$$

Finally, because we simulate isothermal, weakly compressible flows, the equation of state for an ideal gas simplifies to

$$p = c_s^2 \rho. \tag{2.14}$$

2.3.1 Discrete Equilibrium Distribution

For the following investigation, the equilibrium function eq. (2.2) is subdivided into terms involving $\mathbf{u}$ and terms not involving $\mathbf{u}$:

$$\begin{aligned} f^{eq} &= \frac{\rho}{(2\pi RT)^{3/2}} \cdot \exp\left(-\frac{\boldsymbol{\xi}^2}{2RT} + \frac{\boldsymbol{\xi}\cdot\mathbf{u}}{RT} - \frac{\mathbf{u^2}}{2RT}\right) \\ &= \frac{\rho}{(2\pi RT)^{3/2}} \cdot \exp\left(-\frac{\boldsymbol{\xi}^2}{2RT}\right) \cdot \exp\left(\frac{\boldsymbol{\xi}\cdot\mathbf{u}}{RT} - \frac{\mathbf{u^2}}{2RT}\right). \end{aligned} \tag{2.15}$$

The exponential functions may be expanded in a Taylor series around zero up to second order,

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \mathcal{O}(x^3) \qquad |x| \ll 1. \tag{2.16}$$

It can be easily shown through generalized ratio test that this series converges everywhere. However, this truncated series at second order will only yield reasonable results for arguments close enough to zero. Expanding the macroscopic velocity dependent exponential function and using eq. (2.11) together with eq. (2.13) yields

$$f^{eq} = w(\boldsymbol{\xi})\rho \cdot \left(1 + \underbrace{\frac{\boldsymbol{\xi}\cdot\mathbf{u}}{c_s^2}}_{\mathcal{O}(Ma)} - \underbrace{\frac{\mathbf{u}\cdot\mathbf{u}}{2c_s^2}}_{\mathcal{O}(Ma^2)} + \underbrace{\frac{(\boldsymbol{\xi}\cdot\mathbf{u})^2}{2c_s^4}}_{\mathcal{O}(Ma^2)} - \underbrace{\frac{\boldsymbol{\xi}\cdot\mathbf{u}\cdot\mathbf{u}\cdot\mathbf{u}}{2c_s^4}}_{\mathcal{O}(Ma^3)} + \underbrace{\frac{(\mathbf{u}\cdot\mathbf{u})^2}{8c_s^4}}_{\mathcal{O}(Ma^4)}\right), \tag{2.17}$$

where the remaining terms are summarized in a weighting factor $w(\boldsymbol{\xi})$. Truncating at second order in Mach gives the equilibrium function, which is only valid for small Mach numbers because of the second order Taylor expansion as shown above,

$$\begin{aligned} f^{eq} &= w(\boldsymbol{\xi})\rho \cdot \left(1 + \frac{\boldsymbol{\xi}\cdot\mathbf{u}}{c_s^2} + \frac{\boldsymbol{\xi}\cdot\mathbf{u}\cdot\boldsymbol{\xi}\cdot\mathbf{u}}{2c_s^4} - \frac{\mathbf{u}\cdot\mathbf{u}}{2c_s^2}\right) \\ &= w(\boldsymbol{\xi})\rho \cdot \left(1 + \frac{\xi_\alpha u_\alpha}{c_s^2} + \frac{\xi_\alpha u_\alpha \xi_\beta u_\beta}{2c_s^4} - \frac{u_\alpha u_\alpha c_s^2}{2c_s^4}\right) \\ &= w(\boldsymbol{\xi})\rho \cdot \left(1 + \frac{\xi_\alpha u_\alpha}{c_s^2} + \frac{(\xi_\alpha\xi_\beta - c_s^2\delta_{\alpha\beta})u_\alpha u_\beta}{2c_s^4}\right) \\ &= w(\boldsymbol{\xi})\rho \cdot \left(1 + \frac{\boldsymbol{\xi}\cdot\mathbf{u}}{c_s^2} + \frac{1}{2c_s^4}(\boldsymbol{\xi}\boldsymbol{\xi} - c_s^2\mathbf{I}) : \mathbf{u}\mathbf{u}\right). \end{aligned} \tag{2.18}$$

Discretization of phase space means that, in addition to the domain discretization, the velocity space needs to be discretized as well. From the microscopic point of view, we restrict the possible direction of motions of the molecules to a finite set of molecular

velocity vectors $\boldsymbol{\xi}_i$ .The discrete directions are denoted by a subscript i with $i = 0, 1, ..., m$. Hence, the local equilibrium distribution becomes discrete and reads

$$\begin{aligned} f_i^{eq} &= \frac{\rho}{(2\pi c_s^2)^{3/2}} \cdot \exp\left(-\frac{\boldsymbol{\xi}_i^2}{2c_s^2}\right) \cdot \left(1 + \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}}{c_s^2} + \frac{1}{2c_s^4}(\boldsymbol{\xi}_i\boldsymbol{\xi}_i - c_s^2\mathbf{I}) : \mathbf{uu}\right) \\ &= w(\boldsymbol{\xi}_i)\rho \cdot \left(1 + \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}}{c_s^2} + \frac{1}{2c_s^4}(\boldsymbol{\xi}_i\boldsymbol{\xi}_i - c_s^2\mathbf{I}) : \mathbf{uu}\right). \end{aligned} \tag{2.19}$$

It should be pointed out that the space and time dependence of the local equilibrium appears only implicit through the space and time dependence of the conserved variables ρ and $\mathbf{u}$.

2.3.2 DnQm Lattices

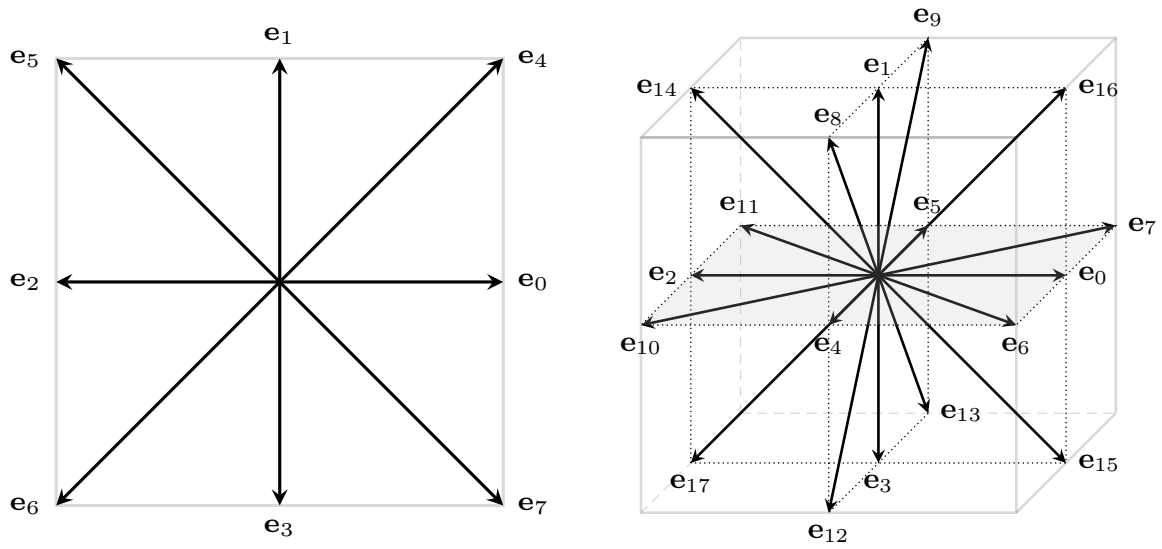
In this section, we will have a closer look at the discretization of the velocity space. The question arises, how the finite set of molecular velocity vectors should be chosen. At first glance, an infinite number of choices may be possible. The key requirement for the velocity discretization is that the moments build from the finite molecular velocity vector set should be Galilean invariant. This means that they should not change due to rotation or translation. Furthermore, the moments eqs. (2.4) to (2.7) involve tensors that are symmetric. Thus, a symmetry constraint is implied on the finite velocity set. From these requirements, a family of velocity discretizations emerged in which the different sets are named according to the number m of directions used to discretize the velocity space Q in n dimensions D [50]. The entity of the phase space discretization is commonly referred to as a lattice, which is often synonymously used for the discretization of the velocity space only.

Here, exemplary, two such lattices D2Q9 and D3Q19, on which *SAM-Lattice* is based on, are shown in Figure 2.1. Both of these lattices are nearest neighbor lattices. The D2Q9 lattice can be viewed as the projection of the D3Q19 lattice into the plane.

Given a lattice of the aforementioned structure, the central task remains to approximate the integrals from the moment definition in Section 2.2.2 in the discrete velocity space through numerical integration such that,

$$\int a(\boldsymbol{\xi})_k f^{eq}\,\mathrm{d}\boldsymbol{\xi} = \sum_i \tilde{w}_i a(\mathbf{e}_i)_k f_i^{eq}, \tag{2.20}$$

where $\mathbf{e}_i$ is the vector to the quadrature point in direction $\boldsymbol{\xi}_i$ at which f_i^{eq} is evaluated and $\tilde{w}_i$ is the corresponding weight. In the sequel, we will refer to $\mathbf{e}_i$ as lattice vectors, where it is understood as a molecular velocity vector of the lattice. Owing to the structure of the truncated equilibrium distribution, eq. (2.19) can be recognized as a truncated series



(a) D2Q9 (b) D3Q19

Figure 2.1: DnQm lattices

expansion of Hermite polynomials $\mathcal{H}(\boldsymbol{\xi}_i)$ to the second order and, thus, Gauss-Hermite quadrature is employed. With a proper rescaling of the Hermite basis, the weights can be chosen such that the quadrature points $\mathbf{e}_i$ coincide with the next neighbor lattice nodes in direction i [43]. An overall weighting factor w_i is defined, which encompasses the weight of the quadrature $\tilde{w}_i$ and the weight of the discrete distribution $w(\boldsymbol{\xi}_i)$:

$$w_i(\boldsymbol{\xi}_i) = \tilde{w}_i \cdot w(\boldsymbol{\xi}_i). \tag{2.21}$$

As one can see from eq. (2.19), eq. (2.20) produces tensor products of order $k+2$. The symmetry relations are provided through the projection of $a(\boldsymbol{\xi})_k$ onto the Hermite basis, that is, the computation of scalar products between $a(\boldsymbol{\xi})_k$ and the vectors of the basis. From the fact that the Hermite basis is orthogonal by construction, it follows that tensor products of odd order vanish

$$\sum_i w_i \mathbf{e}_i^{(2n-1)} = 0 \qquad \forall n \in \mathbb{N} \text{ with } n \leq k. \tag{2.22}$$

A direct consequence of the lattice symmetry are the following relations that are indis-

pensable lattice properties for the goal to simulate fluid dynamics [28, 40]:

$$\sum_i w_i = 1 \tag{2.23}$$

$$\sum_i w_i e_{i\alpha} e_{i\beta} = c_s^2 \delta_{\alpha\beta} \tag{2.24}$$

$$\sum_i w_i e_{i\alpha} e_{i\beta} e_{i\gamma} e_{i\delta} = c_s^4 (\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}). \tag{2.25}$$

Therefore, with this order of truncation of the equilibrium function, the lattice must be chosen so that it is able to recover symmetric 4th rank tensors because we have $k \leq 3$.

The lattices in Figure 2.1 are also known under the name of multi-speed lattices because of the different length of the $\mathbf{e}_i$ vectors. In fact, these lattices possess three different normalized magnitudes of lattice speeds: $\sqrt{2}$ for the cross links, 1 for the principle axis links, and 0 for the zero direction representing particles at rest. Table 2.3 summarizes the Gaussian quadrature points, which coincide with the neighbor nodes and their corresponding weights.

An important result of the discretization of the velocity space is the constant (and lattice specific) relation between the components ξ of the molecular velocity vector $\boldsymbol{\xi}$ and the speed of sound

$$\xi = \sqrt{3} c_s, \tag{2.26}$$

which states that $\xi \sim c_s$. With eq. (2.26), the weighting factor $w(\boldsymbol{\xi}_i)$ becomes constant and so does the overall weighting factor w_i for which the values are given in Table 2.3.

For the derivation of the integral approximations using Gauss-Hermite quadrature, the reader is referred to [57]. An extensive discussion on different lattice structures may be found in [61].

2.3.3 Moments of the Discrete Distribution

With the findings in Section 2.3.2, we can calculate the macroscopic moments from the discrete probability distribution function by applying the integration formulas. Thus, the discrete and truncated equilibrium eq. (2.19) takes its final lattice form:

$$f_i^{eq} = w_i \rho \cdot \left(1 + \frac{\mathbf{e}_i \cdot \mathbf{u}}{c_s^2} + \frac{1}{2c_s^4}(\mathbf{e}_i \mathbf{e}_i - c_s^2 \mathbf{I}) : \mathbf{u}\mathbf{u}\right). \tag{2.27}$$

Table 2.3: Table of lattices

	D2Q9		D3Q19	
Gaussian point	Lattice vector	Weight w_i	Lattice vector	Weight w_i
$\mathbf{e}_0$	$(1,0)$	$1/9$	$(1,0,0)$	$1/18$
$\mathbf{e}_1$	$(0,1)$	$1/9$	$(0,1,0)$	$1/18$
$\mathbf{e}_2$	$(-1,0)$	$1/9$	$(-1,0,0)$	$1/18$
$\mathbf{e}_3$	$(0,-1)$	$1/9$	$(0,-1,0)$	$1/18$
$\mathbf{e}_4$	$(1,1)$	$1/36$	$(0,0,1)$	$1/18$
$\mathbf{e}_5$	$(-1,1)$	$1/36$	$(0,0,-1)$	$1/18$
$\mathbf{e}_6$	$(-1,-1)$	$1/36$	$(1,0,1)$	$1/36$
$\mathbf{e}_7$	$(1,-1)$	$1/36$	$(1,0,-1)$	$1/36$
$\mathbf{e}_8$	$(0,0)$	$4/9$	$(0,1,1)$	$1/36$
$\mathbf{e}_9$	–	–	$(0,1,-1)$	$1/36$
$\mathbf{e}_{10}$	–	–	$(-1,0,1)$	$1/36$
$\mathbf{e}_{11}$	–	–	$(-1,0,-1)$	$1/36$
$\mathbf{e}_{12}$	–	–	$(0,-1,1)$	$1/36$
$\mathbf{e}_{13}$	–	–	$(0,-1,-1)$	$1/36$
$\mathbf{e}_{14}$	–	–	$(-1,1,0)$	$1/36$
$\mathbf{e}_{15}$	–	–	$(1,-1,0)$	$1/36$
$\mathbf{e}_{16}$	–	–	$(1,1,0)$	$1/36$
$\mathbf{e}_{17}$	–	–	$(-1,-1,0)$	$1/36$
$\mathbf{e}_{18}$	–	–	$(0,0,0)$	$1/3$

The equilibrium moments computed on the D3Q19 lattice are therefore given by [11, 28, 59]

$$\rho = \sum_i f_i^{eq} \tag{2.28}$$

$$\rho\mathbf{u} = \sum_i \mathbf{e}_i f_i^{eq} \tag{2.29}$$

$$\mathbf{\Pi^{eq}} = \sum_i \mathbf{e}_i\mathbf{e}_i f_i^{eq} = p\mathbf{I} + \rho\mathbf{uu} \tag{2.30}$$

$$\mathbf{Q^{eq}} = \sum_i e_{i,\alpha}e_{i,\beta}e_{i,\gamma}f_i^{eq} = p(u_\alpha\delta_{\beta\gamma} + u_\beta\delta_{\alpha\gamma} + u_\gamma\delta_{\alpha\beta}). \tag{2.31}$$

It should be noted that the only moment that differs from its continuous counterpart is the third order moment $\mathbf{Q}^{\mathbf{eq}}$. The discrete version lacks the $\rho\mathbf{uuu}$ term. This is due to the fact that next neighbor lattices do not provide enough discrete directions to compute the third order moment correctly. Strictly speaking, this breaks the Galilean invariance [14], but it is often argued that the deviation is small, due to this term being $\mathcal{O}(Ma^3)$ [51]. A corresponding evaluation for the moments of the complete distribution yields:

$$\begin{aligned}\rho &= \sum_i f_i \\ \rho\mathbf{u} &= \sum_i \mathbf{e}_i f_i \\ \mathbf{\Pi} &= \sum_i \mathbf{e}_i\mathbf{e}_i f_i = p\mathbf{I} + \rho\mathbf{uu} - \boldsymbol{\sigma}.\end{aligned} \tag{2.32}$$

With these definitions, the transition from the continuous LBE to the discrete LBE in terms of phase space and the probability distribution is complete. Before we proceed to the Lattice Boltzmann equation, we will show that a solution to the restricted (low Mach, isothermal) mesoscopic dynamics of the Boltzmann equation is in fact a solution to the macroscopic Navier-Stokes equations.

2.3.4 Derivation of the Navier-Stokes Equations

The basis of the derivation of the Navier-Stokes equations from the continuous Boltzmann equation is the Chapman-Enskog expansion. It is a pertubation method in which a small parameter $\epsilon \ll 1$ is introduced to separate physical phenomena that happen on different scales. Physically, the parameter ϵ can be interpreted as the Knudsen number Kn, which is the ratio of the mean free path and a characteristic length scale L or a time scale equivalent T. We denote this by the infinitesimal space scale δx and time scale δt, respectively,

$$Kn = \frac{\delta x}{L} = \frac{\delta t}{T} = \epsilon. \tag{2.33}$$

In the Chapman-Enskog expansion, the distributions are expanded around equilibrium in powers of ϵ. Therefore, the distributions with a superscript represent corrections to the non-equilibrium state, which get less important with increasing order of ϵ. As the Knudsen number is small, only small perturbations from equilibrium are allowed, which renders this method only valid for near continuum flows,

$$f = f^{eq} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + \mathcal{O}(\epsilon^3). \tag{2.34}$$

Additionally, the time scales and the corresponding differential operators are separated

$$\partial_t = \partial_{t_0} + \epsilon\partial_{t_1} + \mathcal{O}(\epsilon^2). \tag{2.35}$$

Short term physics happens on order t_1, whereas long term physics such as diffusion happens on t_0. Since the derivatives of the scales t_0 and t_1 may be of the same order, the parameter ϵ ensures appropriate hierarchy of scales. In the classical approach, the space scales are also expanded and the relations eq. (2.34) and eq. (2.35) are inserted into a second order Taylor expansion of eq. (2.1). Taking moments of these resulting equations for the different orders of ϵ leads to the Euler and Navier-Stokes equations, respectively.

An alternative formulation of the Chapman-Enskog procedure is highlighted by *Dellar* in [14], which we follow here in a more elaborated manner. In contrast to the classical approach, we take moments of eq. (2.1) first and expand the moments in ϵ. The resulting equations describe the evolution of the moments with contributions from different physical scales. In that sense, the macroscopic equations are solutions to the moment equations on the long term scale. We will use this fact by truncating the expansions at appropriate orders as shown below. The following moment expansion together with eq. (2.35) is introduced:

$$\begin{aligned} \mathbf{\Pi} &= \mathbf{\Pi}^{\mathbf{eq}} + \epsilon\mathbf{\Pi}^{(\mathbf{1})} + \epsilon^2\mathbf{\Pi}^{(\mathbf{2})} + \mathcal{O}(\epsilon^3) \\ \mathbf{Q} &= \mathbf{Q}^{\mathbf{eq}} + \epsilon\mathbf{Q}^{(\mathbf{1})} + \epsilon^2\mathbf{Q}^{(\mathbf{2})} + \mathcal{O}(\epsilon^3). \end{aligned} \tag{2.36}$$

Since the collision process happens locally, the space scales are left unexpanded. The conserved moments eq. (2.4) and eq. (2.5) are also unexpanded because of the conservation condition eq. (2.8).

We start the derivation by extracting the smallness parameter ϵ from the collision time through $\tau = \epsilon\tau^*$. This formal rescaling ensures that both sides of the equation are of order $\mathcal{O}(1)$. For simplicity and without loss of generality, the derivation is done using the continuous variables

$$\partial_t f + \boldsymbol{\xi} \cdot \nabla f = \frac{1}{\epsilon\tau^*}(f^{eq} - f). \tag{2.37}$$

Next, we take moments of eq. (2.37) up to second order, starting with zero order moments

$$\int \partial_t f \,\mathrm{d}\boldsymbol{\xi} + \int \boldsymbol{\xi} \cdot \nabla f \,\mathrm{d}\boldsymbol{\xi} = \frac{1}{\epsilon\tau^*}\left(\int f^{eq} \,\mathrm{d}\boldsymbol{\xi} - \int f \,\mathrm{d}\boldsymbol{\xi}\right). \tag{2.38}$$

The right hand side becomes zero by virtue of the conservation condition eq. (2.8). Applying the product rule backwards on the second term on the left hand side yields:

$$\partial_t \int f \,\mathrm{d}\boldsymbol{\xi} + \int \nabla \cdot (f\boldsymbol{\xi}) \,\mathrm{d}\boldsymbol{\xi} - \int f \nabla \cdot \boldsymbol{\xi} \,\mathrm{d}\boldsymbol{\xi} = 0. \tag{2.39}$$

By definition, the space derivative of the molecular velocity is zero. Inserting the time

expansion eq. (2.35), the equation further becomes

$$\begin{aligned}\partial_{t_0}\int f\,\mathrm{d}\boldsymbol{\xi}+\epsilon\partial_{t_1}\int f\,\mathrm{d}\boldsymbol{\xi}+\nabla\cdot\int(f\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}&=0+\mathcal{O}(\epsilon^2)\\ \Leftrightarrow\partial_{t_0}\rho+\epsilon\partial_{t_1}\rho+\nabla\cdot(\rho\mathbf{u})&=0+\mathcal{O}(\epsilon^2).\end{aligned}\tag{2.40}$$

Now, we take first order moments. Again, the right hand side vanishes because of the conservation condition.

$$\partial_{t_0}\int\boldsymbol{\xi}f\,\mathrm{d}\boldsymbol{\xi}+\epsilon\partial_{t_1}\int\boldsymbol{\xi}f\,\mathrm{d}\boldsymbol{\xi}+\nabla\cdot\int\boldsymbol{\xi}(f\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}=0+\mathcal{O}(\epsilon^2)\tag{2.41}$$

Inserting the moment expansion eq. (2.36) leads to:

$$\partial_{t_0}\rho\mathbf{u}+\epsilon\partial_{t_1}\rho\mathbf{u}+\nabla\cdot(\boldsymbol{\Pi}^{\mathbf{eq}}+\epsilon\boldsymbol{\Pi}^{(\mathbf{1})})=0+\mathcal{O}(\epsilon^2)\tag{2.42}$$

Accordingly, taking second order moments yields:

$$\partial_{t_0}\int\boldsymbol{\xi}\boldsymbol{\xi}f\,\mathrm{d}\boldsymbol{\xi}+\epsilon\partial_{t_1}\int\boldsymbol{\xi}\boldsymbol{\xi}f\,\mathrm{d}\boldsymbol{\xi}+\nabla\cdot\int\boldsymbol{\xi}\boldsymbol{\xi}(f\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}=\frac{1}{\epsilon\tau^*}\left(\int\boldsymbol{\xi}\boldsymbol{\xi}f^{eq}\,\mathrm{d}\boldsymbol{\xi}-\int\boldsymbol{\xi}\boldsymbol{\xi}f\,\mathrm{d}\boldsymbol{\xi}\right)+\mathcal{O}(\epsilon^2)\tag{2.43}$$

Inserting the expansions eq. (2.36) gives a non-zero right hand side:

$$\partial_{t_0}(\boldsymbol{\Pi}^{\mathbf{eq}}+\epsilon\boldsymbol{\Pi}^{(\mathbf{1})})+\epsilon\partial_{t_1}(\boldsymbol{\Pi}^{\mathbf{eq}}+\epsilon\boldsymbol{\Pi}^{(\mathbf{1})})+\nabla\cdot(\mathbf{Q}^{\mathbf{eq}}+\epsilon\mathbf{Q}^{(\mathbf{1})})=\frac{1}{\epsilon\tau^*}\left(\boldsymbol{\Pi}^{\mathbf{eq}}-(\boldsymbol{\Pi}^{\mathbf{eq}}+\epsilon\boldsymbol{\Pi}^{(\mathbf{1})})\right)+\mathcal{O}(\epsilon^2)\tag{2.44}$$

Zeroth Order Truncation

Taking moments up to second order resulted in the equations (2.40), (2.42), and (2.44), which is enough to recover the Navier-Stokes equations. As one can see, each moment equation contains contributions for which higher order moments need to be determined. The moment equation of the next order in turn depends on subsequent moments. Therefore, one ends up with an infinite series of moment equations. The particular higher order moments can be viewed as corrections to the evolution of non-equilibrium moments. To truncate this infinite series lies at the heart of the Chapman-Enskog expansion. Thus, truncating the derived moment equations at $\mathcal{O}(1)$ yields the compressible Euler equations:

$$\partial_{t_0}\rho+\nabla\cdot(\rho\mathbf{u})=0\tag{2.45}$$

$$\partial_{t_0}\rho\mathbf{u}+\nabla\cdot(\boldsymbol{\Pi}^{\mathbf{eq}})=0.\tag{2.46}$$

With the definition of $\boldsymbol{\Pi}^{\mathbf{eq}}$ in eq. (2.6) we get

$$\begin{aligned}\partial_{t_0}\rho\mathbf{u}+\nabla\cdot(p\mathbf{I}+\rho\mathbf{u}\mathbf{u})&=0\\ \Leftrightarrow\partial_{t_0}\rho\mathbf{u}+\mathbf{I}\cdot\nabla p+p\nabla\cdot\mathbf{I}+\nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u})&=0.\end{aligned}\tag{2.47}$$

For the evaluation of the divergence in the last term, we switch to index notation

$$\begin{aligned} &\partial_{t_0}\rho u_\alpha + \partial_\alpha p + \partial_\beta(\rho u_\alpha u_\beta) = 0 \\ \Leftrightarrow\ &\partial_{t_0}\rho u_\alpha + \partial_\alpha p + u_\alpha u_\beta \partial_\beta \rho + \rho u_\beta \partial_\beta u_\alpha + \rho u_\alpha \partial_\beta u_\beta = 0. \end{aligned} \tag{2.48}$$

Factoring out u_α and applying the product rule backwards yields

$$\begin{aligned} &\partial_{t_0}\rho u_\alpha + \partial_\alpha p + \rho u_\beta \partial_\beta u_\alpha + u_\alpha(u_\beta \partial_\beta \rho + \rho \partial_\beta u_\beta) = 0 \\ &\Leftrightarrow \partial_{t_0}\rho \mathbf{u} + \nabla p + \rho \mathbf{u}\cdot\nabla \mathbf{u} + \mathbf{u}(\nabla\cdot(\rho\mathbf{u})) = 0. \end{aligned} \tag{2.49}$$

Taking the time derivative and rearranging the resulting terms gives

$$\begin{aligned} &\mathbf{u}\partial_{t_0}\rho + \rho\partial_{t_0}\mathbf{u} + \nabla p + \rho \mathbf{u}\cdot\nabla \mathbf{u} + \mathbf{u}(\nabla\cdot(\rho\mathbf{u})) = 0 \\ &\Leftrightarrow \rho(\partial_{t_0}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}) + \mathbf{u}(\partial_{t_0}\rho + \nabla\cdot(\rho\mathbf{u})) = -\nabla p. \end{aligned} \tag{2.50}$$

The last term in parenthesis on the left hand side represents the continuity equation (2.45) and is, therefore, equal to zero. What remains is the total derivative of $\mathbf{u}$. This is the momentum equation of the Euler equations

$$\begin{aligned} &\rho\frac{D\mathbf{u}}{Dt_0} = -\nabla p \\ \Leftrightarrow\ &\partial_{t_0}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho}\nabla p. \end{aligned} \tag{2.51}$$

There is no external force and especially no friction present. Fluid motion is merely governed by pressure gradients. The incorporation of viscous forces leads to the Navier-Stokes equations.

First Order Truncation

In the context of our derivation, we truncate the moment expansion at order $\mathcal{O}(\epsilon)$ and compute the first order correction to the momentum flux $\epsilon\mathbf{\Pi}^{(1)}$ apparent in eq. (2.42). Truncating eq. (2.40), eq. (2.42), and eq. (2.44) at $\mathcal{O}(\epsilon)$ gives,

$$\partial_{t_0}\rho + \epsilon\partial_{t_1}\rho + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{2.52}$$

$$\partial_{t_0}\rho\mathbf{u} + \epsilon\partial_{t_1}\rho\mathbf{u} + \nabla\cdot(\mathbf{\Pi}^{\mathbf{eq}} + \epsilon\mathbf{\Pi}^{(1)}) = 0 \tag{2.53}$$

$$\partial_{t_0}(\mathbf{\Pi}^{\mathbf{eq}} + \epsilon\mathbf{\Pi}^{(1)}) + \epsilon\partial_{t_1}(\mathbf{\Pi}^{\mathbf{eq}} + \epsilon\mathbf{\Pi}^{(1)}) + \nabla\cdot(\mathbf{Q}^{\mathbf{eq}} + \epsilon\mathbf{Q}^{(1)}) = -\frac{1}{\tau^*}\mathbf{\Pi}^{(1)}. \tag{2.54}$$

Because mass and momentum are conserved during collisions, terms of the scale t_1 have no impact on those quantities. Therefore, eq. (2.52) and eq. (2.53) simplify to:

$$\partial_{t_0}\rho + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{2.55}$$

$$\partial_{t_0}\rho\mathbf{u} + \nabla\cdot(\mathbf{\Pi}^{\mathbf{eq}} + \epsilon\mathbf{\Pi}^{(1)}) = 0. \tag{2.56}$$

The second order moment equation (2.54) gives the evolution of the first non-equilibrium contribution of the momentum flux $\mathbf{\Pi^{(1)}}$. In order to compute the first correction $\epsilon\mathbf{\Pi^{(1)}}$ in eq. (2.56), we multiply eq. (2.54) with ϵ,

$$\partial_{t_0}(\epsilon\mathbf{\Pi^{eq}} + \epsilon^2\mathbf{\Pi^{(1)}}) + \epsilon^2\partial_{t_1}(\mathbf{\Pi^{eq}} + \epsilon\mathbf{\Pi^{(1)}}) + \nabla\cdot(\epsilon\mathbf{Q^{eq}} + \epsilon^2\mathbf{Q^{(1)}}) = -\frac{1}{\tau^*}\epsilon\mathbf{\Pi^{(1)}}. \tag{2.57}$$

On the current truncation order, terms of order $\mathcal{O}(\epsilon^2)$ are neglected. Thus the first correction term depends on the equilibrium part of the second and third order moments only,

$$\begin{aligned}\partial_{t_0}\epsilon\mathbf{\Pi^{eq}} + \nabla\cdot\epsilon\mathbf{Q^{eq}} &= -\frac{1}{\tau^*}\epsilon\mathbf{\Pi^{(1)}}\\ \Leftrightarrow -\epsilon\tau^*(\partial_{t_0}\mathbf{\Pi^{eq}} + \nabla\cdot\mathbf{Q^{eq}}) &= \epsilon\mathbf{\Pi^{(1)}}.\end{aligned} \tag{2.58}$$

Obviously, $\mathbf{\Pi^{(1)}}$ consists of the two contributions in parentheses, which we will examine separately to attain their meaning in the macroscopic equation. Inserting the definition of $\mathbf{\Pi^{eq}}$ from eq. (2.6) gives

$$\begin{aligned}\partial_{t_0}\mathbf{\Pi^{eq}} &= \partial_{t_0}(c_s^2\rho\mathbf{I} + \rho\mathbf{uu})\\ &= c_s^2\mathbf{I}\partial_{t_0}\rho + \partial_{t_0}(\rho\mathbf{u})\otimes\mathbf{u} + \rho\mathbf{u}\otimes\partial_{t_0}\mathbf{u}\\ &= c_s^2\mathbf{I}\partial_{t_0}\rho + \partial_{t_0}(\rho\mathbf{u})\mathbf{u} + \rho\mathbf{u}\partial_{t_0}\mathbf{u}.\end{aligned} \tag{2.59}$$

Because $\mathbf{\Pi^{eq}}$ is an equilibrium quantity, we can use the $\mathcal{O}(1)$ approximation of eq. (2.46) to obtain

$$\partial_{t_0}\mathbf{\Pi^{eq}} = c_s^2\mathbf{I}\partial_{t_0}\rho + (-\nabla\cdot\mathbf{\Pi^{eq}})\mathbf{u} + \rho\mathbf{u}\partial_{t_0}\mathbf{u}. \tag{2.60}$$

Applying the product rule backwards on the last term and subsequent insertion of eq. (2.46) gives

$$\begin{aligned}\partial_{t_0}\mathbf{\Pi^{eq}} &= c_s^2\mathbf{I}\partial_{t_0}\rho + (-\nabla\cdot\mathbf{\Pi^{eq}})\mathbf{u} + \mathbf{u}(\partial_{t_0}\rho\mathbf{u} - \mathbf{u}\partial_{t_0}\rho)\\ &= c_s^2\mathbf{I}\partial_{t_0}\rho + (-\nabla\cdot\mathbf{\Pi^{eq}})\mathbf{u} + \mathbf{u}(-\nabla\cdot\mathbf{\Pi^{eq}}) - \mathbf{uu}\partial_{t_0}\rho.\end{aligned} \tag{2.61}$$

Using the definition of $\mathbf{\Pi^{eq}}$ and eliminating the last time derivative through the continuity equation eq. (2.45) leads to

$$\begin{aligned}\partial_{t_0}\mathbf{\Pi^{eq}} &= c_s^2\mathbf{I}\partial_{t_0}\rho - (\nabla\cdot(c_s^2\rho\mathbf{I} + \rho\mathbf{uu}))\mathbf{u} - \mathbf{u}(\nabla\cdot(c_s^2\rho\mathbf{I} + \rho\mathbf{uu})) - \mathbf{uu}\partial_{t_0}\rho\\ &= -c_s^2\mathbf{I}\nabla\cdot(\rho\mathbf{u}) - (c_s^2\nabla\cdot(\rho\mathbf{I}) + \nabla\cdot(\rho\mathbf{uu}))\mathbf{u} - \mathbf{u}((c_s^2\nabla\cdot(\rho\mathbf{I})\\ &\quad + \nabla\cdot(\rho\mathbf{uu})) + \mathbf{uu}\nabla\cdot(\rho\mathbf{u})\\ &= -c_s^2\delta_{\alpha\beta}\partial_\gamma\rho u_\gamma - c_s^2u_\alpha\partial_\beta\rho - u_\alpha\partial_\gamma\rho u_\beta u_\gamma - c_s^2u_\beta\partial_\alpha\rho - u_\beta\partial_\gamma\rho u_\alpha u_\gamma + u_\alpha u_\beta\partial_\gamma\rho u_\gamma,\end{aligned} \tag{2.62}$$

where the last two terms partly cancel to give the expression

$$\partial_{t_0}\mathbf{\Pi^{eq}} = -c_s^2\delta_{\alpha\beta}\partial_\gamma\rho u_\gamma - c_s^2 u_\alpha\partial_\beta\rho - c_s^2 u_\beta\partial_\alpha\rho - u_\alpha\partial_\gamma\rho u_\beta u_\gamma - u_\beta\rho u_\gamma\partial_\gamma u_\alpha. \tag{2.63}$$

The dyadic product in the last term is commutative, because the resulting tensor is symmetric. Thus, the last two terms can be collapsed in a single divergence term of a third rank tensor:

$$\partial_{t_0}\mathbf{\Pi^{eq}} = -c_s^2\delta_{\alpha\beta}\partial_\gamma\rho u_\gamma - c_s^2 u_\alpha\partial_\beta\rho - c_s^2 u_\beta\partial_\alpha\rho - \partial_\gamma\rho u_\alpha u_\beta u_\gamma. \tag{2.64}$$

Equation (2.64) gives an expression for the time derivative of the equilibrium part of the momentum flux tensor $\partial_{t_0}\Pi^{eq}_{\alpha\beta}$. Next, the second part of the parentheses in eq. (2.58) is examined. The divergence for the definition of the third order moment tensor eq. (2.7) gives

$$\partial_\gamma Q^{eq}_{\alpha\beta\gamma} = \partial_\gamma\rho u_\alpha u_\beta u_\gamma + \partial_\gamma c_s^2\rho u_\alpha\delta_{\beta\gamma} + \partial_\gamma c_s^2\rho u_\beta\delta_{\alpha\gamma} + \partial_\gamma c_s^2\rho u_\gamma\delta_{\alpha\beta}. \tag{2.65}$$

Applying the product rule and using the identity $\nabla\mathbf{I} = 0$ we get

$$\begin{aligned}\partial_\gamma Q^{eq}_{\alpha\beta\gamma} &= \partial_\gamma\rho u_\alpha u_\beta u_\gamma + c_s^2\delta_{\beta\gamma}\partial_\gamma\rho u_\alpha + c_s^2\delta_{\alpha\gamma}\partial_\gamma\rho u_\beta + c_s^2\delta_{\alpha\beta}\partial_\gamma\rho u_\gamma \\ &= \partial_\gamma\rho u_\alpha u_\beta u_\gamma + c_s^2\partial_\beta\rho u_\alpha + c_s^2\partial_\alpha\rho u_\beta + c_s^2\delta_{\alpha\beta}\partial_\gamma\rho u_\gamma \\ &= \partial_\gamma\rho u_\alpha u_\beta u_\gamma + c_s^2 u_\alpha\partial_\beta\rho + c_s^2\rho\partial_\beta u_\alpha + c_s^2\rho\partial_\alpha u_\beta + c_s^2 u_\beta\partial_\alpha\rho + c_s^2\delta_{\alpha\beta}\partial_\gamma\rho u_\gamma.\end{aligned} \tag{2.66}$$

Rearranging the last equation finally gives an expression for the divergence of $\mathbf{Q^{eq}}$:

$$\partial_\gamma Q^{eq}_{\alpha\beta\gamma} = c_s^2\delta_{\alpha\beta}\partial_\gamma\rho u_\gamma + c_s^2 u_\alpha\partial_\beta\rho + c_s^2 u_\beta\partial_\alpha\rho + \partial_\gamma\rho u_\alpha u_\beta u_\gamma + c_s^2\rho(\partial_\beta u_\alpha + \partial_\alpha u_\beta). \tag{2.67}$$

At this point, eq. (2.58) can be evaluated by adding the two expressions $\partial_{t_0}\mathbf{\Pi^{eq}}$ and $\nabla\cdot\mathbf{Q^{eq}}$ that were just derived in detail. One can see that all terms except the last parenthesis in eq. (2.67) cancel on adding. Therefore, the first order correction term to the momentum flux is given by:

$$\epsilon\Pi^{(1)}_{\alpha\beta} = -\epsilon\tau^* c_s^2\rho(\partial_\beta u_\alpha + \partial_\alpha u_\beta). \tag{2.68}$$

Finally, the formal rescaling of eq. (2.37) is revoked by absorbing ϵ into the collision time τ^*. Hence, the viscous contribution $\epsilon\mathbf{\Pi^{(1)}}$ takes the form:

$$\epsilon\Pi^{(1)}_{\alpha\beta} = -\tau c_s^2\rho(\partial_\beta u_\alpha + \partial_\alpha u_\beta). \tag{2.69}$$

Obviously, the expression in parenthesis is twice the macroscopic strain rate tensor $\boldsymbol{\epsilon}$. Thus, the dynamic viscosity μ is related to the collision time according to

$$\mu = \tau c_s^2\rho \tag{2.70}$$

$$\Leftrightarrow \nu = \tau c_s^2, \tag{2.71}$$

where ν is the kinematic viscosity and $-\epsilon\Pi^{(1)}_{\alpha\beta}$ is identified with the macroscopic stress tensor $\boldsymbol{\sigma}$,

$$-\epsilon\boldsymbol{\Pi}^{(1)} = \boldsymbol{\sigma}. \tag{2.72}$$

This finding explains the minus sign in the moment definition eq. (2.9). Note that the stress tensor encompasses the parameter ϵ, which will be illuminated in the next chapter. Now we can assemble the first order truncation, which incorporates viscous forces. The zeroth order moment equation remains unchanged and the first order moment equation contains the first correction term to the momentum flux $\epsilon\boldsymbol{\Pi}^{(1)}$. Therefore, we have

$$\begin{aligned} \partial_{t_0}\rho + \nabla\cdot\rho\mathbf{u} &= 0 \\ \rho\left(\partial_{t_0}u_\alpha + u_\beta\partial_\beta u_\alpha\right) &= -\partial_\alpha p + \partial_\beta\left(\mu(\partial_\beta u_\alpha + \partial_\alpha u_\beta)\right) \\ \Leftrightarrow \partial_{t_0}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\frac{1}{\rho}\nabla p + \nabla\cdot\left(\nu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\right), \end{aligned} \tag{2.73}$$

where the same transformations were applied as for the Euler equations. Equation (2.73) is the isothermal compressible Navier-Stokes equation without external force. It should be noted, that through the isothermal assumption, the bulk viscosity is fixed to $\mu' = \mu(2/3)$ [11, 51].

2.4 The Lattice Boltzmann Equation

In the preceding sections, eq. (2.1) was discretized in phase space. Therefore, the continuous Boltzmann equation became discrete because the probability distribution function and the molecular velocity in eq. (2.10) appear in their discretized form. Moreover, it was shown that if the probability density function f is known at any time, the macroscopic solutions of the Navier-Stokes equations are known through the moments of f. The question remains, how f_i is determined from the discrete Boltzmann equation. In order to gain an equation for determining the probability distribution, eq. (2.10) has to be rendered fully discrete by additional discretization in time. Due to the second order accuracy in ϵ, the solutions to the Navier-Stokes equation are formally second order accurate in space and time because of eq. (2.33). A straightforward way to realize a second order time accuracy is the Cranck-Nicolson method where the time integration is done locally using the trapezoidal rule. The time step size will be denoted by Δt. As already stated, the left hand side of the Boltzmann equation describes the transport of the molecules, whereas the right hand side is associated with the collision of molecules. Because the collision time is much smaller than the macroscopic time step size $\tau \ll \Delta t$, the two operations can be treated separately. However, it can be shown that such a time discretization results in first order accuracy only because of the operator splitting (transport and collision) in conjunction with local time integration [13]. A remedy is the integration along a

characteristic, that is, a space-time integration, where the space step Δx is replaced with $\boldsymbol{\xi}_i \Delta t$. Hence, integration of eq. (2.10) in space-time gives

$$\int_0^{\Delta t} \frac{d\, f_i(\mathbf{x} + \boldsymbol{\xi}_i \cdot s, t + s)}{ds}\, ds = \frac{1}{\tau} \int_0^{\Delta t} (f_i^{eq}(\mathbf{x} + \boldsymbol{\xi}_i \cdot s, t + s) - f_i(\mathbf{x} + \boldsymbol{\xi}_i \cdot s, t + s))\, ds, \tag{2.74}$$

where the total derivative formulation on the left hand side integrates directly and yields the difference between end and start point. In addition, the time step size is set so that the integration interval ends with the Gaussian quadrature point in direction i. Therefore, we have

$$\mathbf{e}_i = \Delta x / \Delta t. \tag{2.75}$$

More pictorially, Δt is chosen so that the molecules with velocity $\boldsymbol{\xi}_i$ fly to the corresponding nearest neighbor node within one time step. The right hand side of the equation can be approximated using the trapezoidal rule,

$$\begin{aligned} f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = & \frac{1}{2\tau} \Delta t\, (f_i^{eq}(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) - f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) \\ & + f_i^{eq}(\mathbf{x}, t) - f(\mathbf{x}, t)) + \mathcal{O}(\Delta t^2). \end{aligned} \tag{2.76}$$

This equation is implicit because terms involving $t + \Delta t$ appear on the right hand side of the equation. *He et al.* [30] introduced a change of variables to render the equation explicit:

$$\overline{f_i}(\mathbf{x}, t) = f_i(\mathbf{x}, t) + \frac{\Delta t}{2\tau} \left(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right). \tag{2.77}$$

The resulting equation is now solved for $\overline{f_i}$ instead of f_i and reads

$$\overline{f_i}(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) - \overline{f_i}(\mathbf{x}, t) = \frac{\Delta t}{\tau + 0.5\Delta t} \left(f_i^{eq}(\mathbf{x}, t) - \overline{f_i}(\mathbf{x}, t)\right). \tag{2.78}$$

Furthermore, the fraction in eq. (2.78) is recognized as the dimensionless collision frequency and is denoted by Ω. Mathematically, it is a relaxation parameter that controls the relaxation of f_i towards the local equilibrium state. Here, the natural assumption that the relaxation parameter has to be connected with the viscosity of the gas is confirmed because the collision time appears in the denominator. With the relation eq. (2.71) the dimensionless collision time Ω can be written

$$\Omega = \frac{\Delta t}{\tau + 0.5\Delta t} = \frac{c_s^2 \Delta t}{\nu + 0.5 c_s^2 \Delta t} \qquad \Omega \in [0, 2]. \tag{2.79}$$

In contrast to the (time) continuous version of the discrete Boltzmann equation (2.10), the collision time in the denominator contains an additional contribution, which steams from

discretization. It can be viewed as the lattice viscosity, which is absorbed in the collision time to yield the correct transport coefficient. The left side of eq. (2.10) represents the transport step, which does not even contain any round off errors when implemented in a computer code. Subsequent collision results in values for $\overline{f_i}$ for the proximate time step,

$$\overline{f_i}(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) = \overline{f_i}(\mathbf{x}, t) + \Omega \cdot \left(f_i^{eq}(\mathbf{x}, t) - \overline{f_i}(\mathbf{x}, t) \right). \tag{2.80}$$

The macroscopic values needed to compute the local equilibrium prior to the collision step are computed from the distributions $\overline{f_i}(\mathbf{x}, t)$ after the transport step. Equation (2.80) is the well known Lattice Boltzmann equation (LBE), where we drop the overline of the change of variables in the sequel. Fortunately, the change of variables does not affect the conserved quantities because of the solvability condition eq. (2.8). Yet, higher order moments get affected, especially the macroscopic stress tensor $\epsilon\mathbf{\Pi}^{(\mathbf{1})}$. However, a problem is its correct computation because the direct computation of $\epsilon\mathbf{\Pi}^{(\mathbf{1})}$ from the distributions requires the knowledge of $f^{(1)}$ and ϵ. Thus, the term $\epsilon\mathbf{\Pi}^{(\mathbf{1})}$ is approximated with the non-equilibrium part of the momentum flux tensor and the moment expansion eq. (2.36):

$$\begin{aligned} \epsilon\mathbf{\Pi}^{(\mathbf{1})} \approx \mathbf{\Pi}^{\mathbf{neq}} &= \mathbf{\Pi} - \mathbf{\Pi}^{\mathbf{eq}} \\ &= \epsilon\mathbf{\Pi}^{(\mathbf{1})} + \mathcal{O}(\epsilon^2). \end{aligned} \tag{2.81}$$

The expression in eq. (2.81) reveals that the approximation of the stress tensor is formally second order accurate within the Lattice Boltzmann framework. This statement is supported with numerical results from *SamLattice* in [56] and the literature, e.g., [37].
Now, with the change of variables, the computation of the non-equilibrium part of the second order moment from the distributions $\overline{f_i}$ yields

$$\begin{aligned} \overline{\mathbf{\Pi}} - \overline{\mathbf{\Pi}}^{\,\mathbf{eq}} &= \mathbf{\Pi} + \frac{\Delta t}{2\tau} \left(\mathbf{\Pi} - \mathbf{\Pi}^{\mathbf{eq}} \right) - \mathbf{\Pi}^{\mathbf{eq}} \\ &= \left(1 + \frac{\Delta t}{2\tau} \right) \left(\mathbf{\Pi} - \mathbf{\Pi}^{\mathbf{eq}} \right) \\ \Leftrightarrow \mathbf{\Pi}^{\mathbf{neq}} &= \left(1 + \frac{\Delta t}{2\tau} \right)^{-1} \left(\overline{\mathbf{\Pi}} - \overline{\mathbf{\Pi}}^{\,\mathbf{eq}} \right) \qquad (2.82) \\ \Leftrightarrow \epsilon\mathbf{\Pi}^{(\mathbf{1})} &\approx \left(1 + \frac{\Delta t}{2\tau} \right)^{-1} \overline{\mathbf{\Pi}}^{\,\mathbf{neq}}. \qquad (2.83) \end{aligned}$$

The overlines of the momentum flux tensor will be dropped for simplicity in the sequel as well. Finally, the generalized Hook's law for isotropic materials gives the stress-strain relation [11, 35], which was already implicitly used in eq. (2.69), eq. (2.70), and eq. (2.72),

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \mu' Tr\left(\boldsymbol{\epsilon}\right) \mathbf{I}. \tag{2.84}$$

Here, $Tr\left(\boldsymbol{\epsilon}\right) = 0$ in the incompressible limit. This relates the second order moment tensor to the macroscopic stress tensor using eq. (2.72) in the following way

$$\begin{aligned}\boldsymbol{\sigma} &= -\left(1 + \frac{\Delta t}{2\tau}\right)^{-1} \boldsymbol{\Pi}^{\mathbf{neq}} \\ &= -\left(1 + \frac{c_s^2 \Delta t}{2\nu}\right)^{-1} \boldsymbol{\Pi}^{\mathbf{neq}} \\ &= -\left(\frac{2\nu + c_s^2 \Delta t}{2c_s^2 \Delta t} \cdot \frac{c_s^2 \Delta t}{\nu}\right)^{-1} \boldsymbol{\Pi}^{\mathbf{neq}} \\ &= -\frac{\Omega \nu}{c_s^2 \Delta t} \boldsymbol{\Pi}^{\mathbf{neq}}, \end{aligned} \tag{2.85}$$

where the relations eq. (2.69), eq. (2.70), and eq. (2.79) were used.

2.4.1 Accuracy

This section deals with the accuracy of the LBM for the simulation of weakly compressible and incompressible fluids. For this purpose, we investigate the continuity equation and have a look at the various error terms present in a simulation. In order to provide an order of magnitude estimate, the continuity equation is non-dimensionalized. We will denote dimensionless values by an asterisk. To this end, the non-dimensionalized quantities are introduced together with their estimation in Table 2.4. Most definitions are straightforward, except the pressure transformation. The pressure is split into two parts, where p_0 is the thermodynamic pressure and p_1 is the dynamic pressure. It is assumed that the thermodynamic pressure is spatially and temporally constant and scales with the speed of sound. By contrast, p_1 scales proportional to the dynamic reference pressure $\rho_{ref} U^2$. Because U is on the macroscopic scale and c_s on the molecular scale, the dimensionless dynamic pressure has to be rescaled with Ma^2. Accordingly, things are similar for the density transformation because of the isothermal equation of state. With these definitions, the continuity equation can be rewritten:

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \Leftrightarrow (\partial_t p_0^* + Ma^2 \partial_t p_1^*) \cdot \rho_{ref} + \nabla \cdot ((p_0^* + Ma^2 p_1^*) \cdot \rho_{ref} \cdot \mathbf{u}) &= 0 \\ \Leftrightarrow (\partial_t \rho_0^* + Ma^2 \partial_t \rho_1^*) \cdot \rho_{ref} + \frac{1}{L_{ref}} \nabla^* \cdot ((\rho_0^* + Ma^2 \rho_1^*) \cdot \rho_{ref} \cdot \frac{U}{Ma} \mathbf{u}^*) &= 0 \end{aligned} \tag{2.86}$$

Introducing the dimensionless time derivative and simplifying the equation yields

$$\begin{aligned} (\partial_t^* \rho_0^* + Ma^2 \partial_t^* \rho_1^*) \cdot \rho_{ref} \cdot \frac{U}{L_{ref} \cdot Ma} + \frac{1}{L_{ref}} \nabla^* \cdot ((\rho_0^* + Ma^2 \rho_1^*) \cdot \rho_{ref} \cdot \frac{U}{Ma} \mathbf{u}^*) &= 0 \\ \Leftrightarrow \partial_t^* \rho_0^* + Ma^2 \partial_t^* \rho_1^* + \nabla^* \cdot ((\rho_0^* + Ma^2 \rho_1^*) \cdot \mathbf{u}^*) &= 0. \end{aligned} \tag{2.87}$$

Table 2.4: Non-dimensionalization

Dimensional	Non-dimensionalization	Estimate
$\mathbf{u}$	$\mathbf{u} = \frac{U}{Ma} \cdot \mathbf{u}^*$	$U \sim \lvert\mathbf{u}\rvert$
p	$p = \left(p_0^* + Ma^2 \cdot p_1^*\right) \rho_{ref} \cdot cs^2$ $= \left(\frac{p_0}{\rho_{ref} c_s^2} + Ma^2 \cdot \frac{p_1}{\rho_{ref} U^2}\right) \rho_{ref} \cdot cs^2$	$\partial_t p_0^* = 0,\ \nabla p_0^* = 0$
ρ	$\rho = \left(p_0^* + Ma^2 \cdot p_1^*\right) \rho_{ref}$	$p = c_s^2 \rho,\ p^* = c_s^{*2} \rho^*$
c_s	$c_s = c_s^* \cdot c_s$	$c_s^* = 1$
t	$t = \frac{L_{ref} \cdot t^*}{cs} = Ma \cdot \frac{t^* \cdot L_{ref}}{U}$	$\frac{c_s}{L_{ref}} \sim t$
∂_t	$\partial_t = \frac{U}{L_{ref} \cdot Ma} \partial_{t^*}$	$\frac{dt}{dt^*} = \frac{L_{ref}}{U} \cdot Ma$
∇	$\nabla = \frac{1}{L_{ref}} \nabla^*$	$d\mathbf{x} = L_{ref} \cdot d\mathbf{x}^*$

Finally, Taking the divergence and using the assumptions for ρ_0^* gives

$$\begin{aligned} &\partial_t^* \rho_0^* + Ma^2 \partial_t^* \rho_1^* + (\rho_0^* + Ma^2 \rho_1^*) \nabla^* \cdot \mathbf{u}^* + \mathbf{u}^* \cdot \nabla^* (\rho_0^* + Ma^2 \rho_1^*) = 0 \\ &\quad \Leftrightarrow Ma^2 \partial_t^* \rho_1^* + (\rho_0^* + Ma^2 \rho_1^*) \nabla^* \cdot \mathbf{u}^* + \mathbf{u}^* \cdot Ma^2 \cdot \nabla^* \rho_1^* = 0. \end{aligned} \tag{2.88}$$

As one can see from eq. (2.88), weakly compressible fluids can undergo divergence-free flow by a suitable adjustment of the dynamic pressure. Therefore, incompressible flow is simulated with the LBM similar to artificial compressibility methods used to solve the incompressible Navier-Stokes equations. Summarizing the Mach number terms in a residual error term and remembering the truncation errors from the Chapman-Enskog expansion, it follows that

$$\rho_0^* \nabla^* \cdot \mathbf{u}^* = 0 + \mathcal{O}(Ma^2) + \mathcal{O}(\epsilon^2). \tag{2.89}$$

However, for the simulation of incompressible flow in the vanishing Mach number limit and $Ma \sim \epsilon$, we get the incompressible continuity equation:

$$\rho_0^* \nabla^* \cdot \mathbf{u}^* = 0. \tag{2.90}$$

A non-dimensionalization for the Navier-Stokes equation in the context of the Chapman-Enskog expansion can be found in [42].

Given that the LBM is a numerical scheme, different sources of errors are present in a simulation. First, there are typical discretization errors that scale like the square of the step sizes used, because of the second order accuracy in ϵ. Namely, the space error $E_x = \mathcal{O}(\Delta x^2)$ and the time error $E_t = \mathcal{O}(\Delta t^2)$. Moreover, the equations in this section show that there is a compressibility error present that scales like Ma^2 when incompressible fluids are simulated with the LBM approach. Thus, the compressibility error can be expressed as $E_{Ma} = \mathcal{O}(\Delta t^2/\Delta x^2)$. As stated in [40], the overall error is composed of these three contributions:

$$E = E_x + E_t + E_{Ma} = \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta t^2/\Delta x^2). \tag{2.91}$$

However, this statement does not account for truncation errors of the numerical scheme. As it was proposed by *Junk* [33], the LBM can be viewed as a finite difference scheme. A comprehensive study based on a truncation error analysis of the Boltzmann equation [32] indicates that the relaxation parameter Ω is used as a weight to create an effective finite difference stencil out of the nearest neighbors finite different stencil. Both stencils coincide at $\Omega = 1.0$. A result of the truncation error analysis is that the value of the relaxation time can be used to minimize numerical diffusion errors. Therefore, there are additional errors due to an improper choice of the value of the dimensionless collision time. For details of these investigations the reader is referred to [32]. The findings of [32] are numerically confirmed in, e.g., [36]. Because Ω cannot be chosen independently, eq. (2.91) may be valid for simulations with a fixed Reynolds number where the viscosity is constant. However, for the simulation of non-Newtonian fluids, the dimensionless collision time can vary as will be shown in Section 2.5.3. Thus, an additional error term E_Ω is introduced. The influence of this error source will be investigated in Chapter 7. Moreover, the finite representation of the simulation geometry introduces an error E_g to the overall error. This trite source of error is often forgotten in practical simulations. The simulation domain is approximated with a discretization of the domain boundaries, which is independent of the lattice discretization. We denote the geometry discretization step size by Δg. In Section 4.2.2 we will have a look at the geometry discretization in conjunction with wall boundary conditions. As a result, one can state that the geometry discretization has to be chosen such that $\Delta x \sim \Delta g$ and $\Delta g = \Theta(\Delta x)$. Finally, there are additional errors due to imperfect boundary conditions that we denote by E_{BC}. Sometimes, these are fairly hard to quantify. For wall boundary conditions there exists schemes with $E_{BC} = \mathcal{O}(\Delta x^2)$. Therefore, the overall error is extended to

$$E = E_x + E_t + E_{Ma} + E_\Omega + E_g + E_{BC}. \tag{2.92}$$

This definition only summarizes the main sources of error, which crucially determine the method's convergence.

2.4.2 Diffusive Scaling vs. Acoustic Scaling

In order to improve the accuracy of a LBM simulation, one has the possibility to increase the spatial resolution to capture gradients more precisely. But this cannot be done independently of the time step size since both are coupled through the simulation Mach-number $Ma \propto \Delta t/\Delta x$. Therefore, for weakly compressible flow where the physical Mach-number has to be met, the time step size has to be rescaled like $\Delta t \propto \Delta x$ in order to keep the simulation Mach-number constant. This scaling is known as the acoustic scaling. The name is inspired from the fact that the speed of sound is held constant. Also this scaling is used for grid refinement to keep the macroscopic variables continuous across grid scales. However, for incompressible fluids, the acoustic scaling creates a race condition between decreasing spatial and temporal errors and increasing compressibility errors when increasing the resolution. This actually results in divergence behavior when the compressibility errors begin to dominate [40]. This can also be seen in the definition above. One can state that the acoustic scaling results in a $\mathcal{O}(1)$ behavior of the overall error:

$$E = E_x + E_t + E_{Ma} = \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta x^2/\Delta x^2) \tag{2.93}$$

$$E = \mathcal{O}(1). \tag{2.94}$$

The divergence behavior was already found by *Reider and Sterling* [52] who state that all error terms must be reduced consistently. Thus, the diffusive scaling is introduced, which scales $\Delta t \propto \Delta x^2$ [34]. That way, if the spacing is halved, the Mach number is halved as well, which consequently reduces compressibility effects. This scaling results in second order convergence behavior:

$$E = E_x + E_t + E_{Ma} = \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta x^4/\Delta x^2) \tag{2.95}$$

$$E = \mathcal{O}(\Delta x^2). \tag{2.96}$$

An important remark is that the diffusive scaling reduces the time accuracy effectively to first order because reducing the error by a factor Δx^2 results in a $1/\Delta x^2$ factor increase in the number of time steps needed.

2.4.3 Stability

For the sake of completeness, a short discussion on the stability of the LBM should be given.

It is well known that the Lattice Boltzmann algorithm becomes unstable when the viscosity is low and the collision frequency Ω approaches 2. The equilibrium distribution is crucial in the theoretical framework of the LBM. By definition, the Maxwell distribution satisfies the Second Law of thermodynamics [28]. Unfortunately, the compliance with the

H-theorem is lost in the course of the discretization. In fact, one can prove, that it is impossible to construct polynominal equilibria with this property [63].

Incipient instability manifests itself in spurious oscillations of the solution variables. Actual divergence occurs, when one of the distributions becomes negative. Several approaches to prevent the instability were developed in the past. A group of LBM models emerged, which restore the compliance with the H-theorem. They use a different form of the equilibrium and solve an entropic estimate equation during collision and are, therefore, known under the name Entropic Lattice Boltzmann Methods (ELBM). One simple and yet robust way to ensure stability is the positivity rule or also known as "fix-up" or "hyper viscosity" model. This rule states, that if a distribution becomes negative after the collision, the collision is rerun with just enough artificial viscosity to ensure that the minimum value of the distribution is zero. A comparison of the positivity rule and the ELBM can be found in [60].

Introduction of artificial viscosity is a major obstacle to the simulation of, e.g., acoustic waves. An attempt to overcome this problem was developed by *Ricot et al.* [54], who use filtering on the LBE to damp out unwanted modes of the oscillating variables.

Details on the implementation of stability measures in *SamSolver* can be found in the PhD thesis of *Schneider* [56]. For an extensive discussion on stabilization of the Lattice Boltzmann Method, the reader is referred to [7].

2.5 Extensions to LBM

In this section, some extensions to the LBM are introduced, which are used in the course of this thesis.

2.5.1 Multiple Relaxation Time

It is well known that the Lattice Boltzmann algorithm is not unconditionally stable. It suffers from numerical instability when the viscosity gets very low or the simulation is under-resolved. Both cases have the effect that the relaxation parameter Ω approaches two, as can be seen from eq. (2.79). Because Ω is the only free parameter in eq. (2.80), this model is known as the *single relaxation time (SRT)* model. An improvement in terms of numerical stability is the *multiple relaxation time (MRT)* model. In this model, the collision operation is reformulated to take place in the moment space [15],

$$\mathbf{M}^{-1} \cdot \mathbf{S} \cdot (\mathbf{M} f_i^{eq} - \mathbf{M} f_i). \tag{2.97}$$

Here, the matrix $\mathbf{M}$ has maximum rank and defines a linear transformation from distribution space to moment space. For the D3Q19 lattice, $\mathbf{M}$ is a 19×19 matrix, which maps the 19 distributions to 19 linear independent moments. The definition of $\mathbf{M}$ is given in

appendix A. In eq. (2.80), the relaxation parameter is extended to the relaxation Matrix $\mathbf{S}$ with the non-zero entries representing independent collision rates $s_{a-e} \in [0, 2]$:

$$\mathbf{S} = diag(0, s_a, s_b, 0, s_c, 0, s_c, 0, s_c, \Omega, s_d, \Omega, s_d, \Omega, \Omega, \Omega, s_e, s_e, s_e). \tag{2.98}$$

Among the moments are the second order moments defined in Section 2.3.3 which get relaxed with Ω to yield the correct transport coefficient. Moments representing mass and momentum do not get relaxed since these are conserved moments and, thus, their entry in $\mathbf{S}$ is zero. The remaining moments are non-hydrodynamic moments, which are sometimes called "ghost variables" in the literature. They are relaxed with the independent relaxation parameters s_{a-e}, which are often set to 1.0. This means that the non-hydrodynamic moments are simply set to their equilibrium value. A listing of the 19 moments used in this model along with their corresponding definition and relaxation time is given in Table 2.5. The parameters s_{a-e} are problem specific and can be used to tune the accuracy and stability. Hints on how to choose these parameters may be found in [23].

Table 2.5: Definition of the D3Q19 MRT model [15, 58]

Moment	m^{eq}	Physical meaning	$a(\mathbf{e}_i)_k$	S_{ii}
m_0	ρ	Density	1	0
m_1	$\rho(u_x^2+u_y^2+u_z^2)$	Density independent kinetic energy	$19 \cdot e_{i,\alpha}e_{i,\alpha} - 30$	s_a
m_2	0	Density independent kinetic energy squared	$0.5(21(e_{i,\alpha}e_{i,\alpha})^2$ $- 53 \cdot e_{i,\alpha}e_{i,\alpha} + 24)$	s_b
m_3	ρu_x	X-component of momentum	$e_{i,x}$	0
m_4	0	X-component of energy flux independent of massflux	$(5 \cdot e_{i,\alpha}e_{i,\alpha} - 9)e_{i,x}$	s_c
m_5	ρu_y	Y-component of momentum	$e_{i,y}$	0
m_6	0	Y-component of energy flux independent of massflux	$(5 \cdot e_{i,\alpha}e_{i,\alpha} - 9)e_{i,y}$	s_c
m_7	ρu_z	Z-component of momentum	$e_{i,z}$	0
m_8	0	Z-component of energy flux independent of massflux	$(5 \cdot e_{i,\alpha}e_{i,\alpha} - 9)e_{i,z}$	s_c
m_9	$\rho(2u_x^2-u_y^2-u_z^2)$	Π_{xx}	$3e_{i,x}^2 - e_{i,\alpha}e_{i,\alpha}$	Ω
m_{10}	0	-	$(3e_{i,\alpha}e_{i,\alpha} - 5)$ $\cdot (e_{i,x}^2 - e_{i,\alpha}e_{i,\alpha})$	s_d
m_{11}	$\rho(u_y^2-u_z^2)$	$\Pi_{yy} - \Pi_{zz}$	$e_{i,y}^2 - e_{i,z}^2$	Ω
m_{12}	0	-	$(3e_{i,\alpha}e_{i,\alpha} - 5)$ $\cdot (e_{i,y}^2 - e_{i,z}^2)$	s_d
m_{13}	$\rho u_x u_y$	Π_{xy}	$e_{i,x}e_{i,y}$	Ω
m_{14}	$\rho u_y u_z$	Π_{yz}	$e_{i,y}e_{i,z}$	Ω
m_{15}	$\rho u_x u_z$	Π_{xz}	$e_{i,x}e_{i,z}$	Ω
m_{16}	0	-	$(e_{i,y}^2 - e_{i,z}^2)e_{i,x}$	s_e
m_{17}	0	-	$(e_{i,z}^2 - e_{i,x}^2)e_{i,y}$	s_e
m_{18}	0	-	$(e_{i,x}^2 - e_{i,y}^2)e_{i,z}$	s_e

2.5.2 Grid Refinement

In this subsection, grid refinement is discussed, which enables local increase of the spatial resolution to capture steep gradients while keeping the overall number of lattice nodes low. Originally proposed by *Filippova and Hänel* [20], the approach consists of an *a priori* refinement of the grid as well as appropriate scaling and interpolation of the distributions between the different grid levels. Although the refinement ratio may be chosen formally arbitrarily, it is common practice to choose the refinement ratio of the coarse grid spacing Δx^c to the fine grid spacing Δx^f equal to two, due to numerical stability and implementation aspects,

$$\frac{\Delta x^c}{\Delta x^f} = 2. \tag{2.99}$$

Yu et al. [64] proposed an interface between the two grids, where both patches overlap by one corase grid spacing Δx^c. This setting, which posses enhanced numerical stability properties, is illustrated in Figure 2.2.

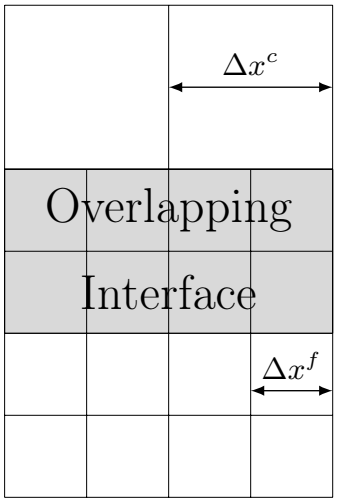


Figure 2.2: Overlapping interface

The boundary conditions for the transition between grids with different resolutions is the continuity of mass, momentum, and stress. As addressed in Section 2.4.2, the acoustic scaling $\Delta t \propto \Delta x$ is applied in the grid refinement process. This scaling renders

the equilibrium function independent of the grid level and hence it follows that

$$\sum_i f_i^{eq,c} = \sum_i f_i^{eq,f} = \rho \tag{2.100}$$

$$\sum_i \mathbf{e}_i f_i^{eq,c} = \sum_i \mathbf{e}_i f_i^{eq,f} = \rho\mathbf{u} \tag{2.101}$$

$$\sum_i \mathbf{e}_i\mathbf{e}_i f_i^{eq,c} = \sum_i \mathbf{e}_i\mathbf{e}_i f_i^{eq,f} = p\mathbf{I} + \rho\mathbf{u}\mathbf{u}, \tag{2.102}$$

for all levels. However, things are different for the non-equilibrium distribution because of the prefactor in the stress calculation eq. (2.82). Thus we have

$$\begin{aligned} \boldsymbol{\sigma}^c &\stackrel{!}{=} \boldsymbol{\sigma}^f \\ \Leftrightarrow -\left(\frac{\Omega^c\nu}{c_s^2\Delta t^c}\right)\boldsymbol{\Pi}^{\mathbf{neq,c}} &\stackrel{!}{=} -\left(\frac{\Omega^f\nu}{c_s^2\Delta t^f}\right)\boldsymbol{\Pi}^{\mathbf{neq,f}}. \end{aligned} \tag{2.103}$$

With this requirement it follows that the non-equilibrium part of the corresponding distribution must be rescaled in order to ensure continuity of the stress tensor. For the transition from fine to coarse and coarse to fine grid levels, the appropriate *scale* factors must be applied, which are due to eq. (2.103) given by

$$scale^{f\to c} = \frac{2\Omega^f}{\Omega^c} \tag{2.104}$$

$$scale^{c\to f} = \frac{\Omega^c}{2\Omega^f}. \tag{2.105}$$

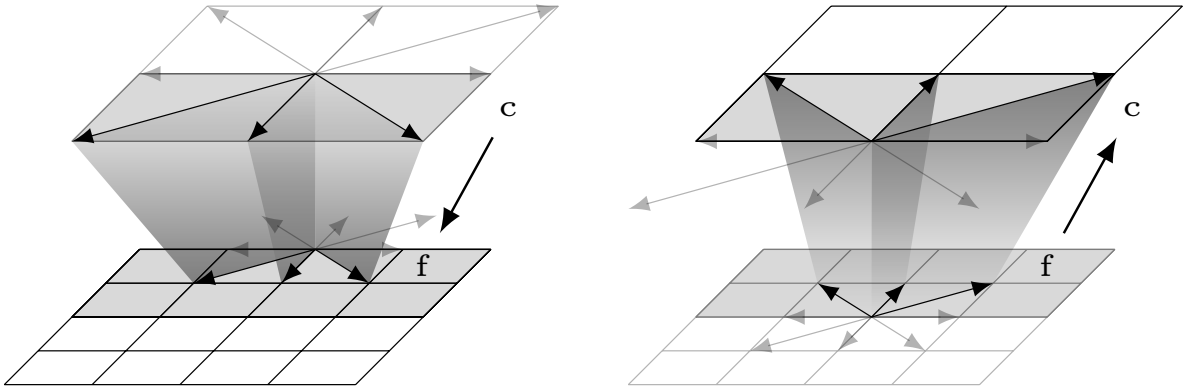
Scaling is applied to the distributions for which the lattice nodes of both levels coincide. The direction for the scaling emerges from the fact that distributions are lacking at the border of the lattice patches with different resolutions. Distributions of missing neighbors are calculated by

$$f_i^c = f_i^{eq} + scale^{f\to c}(f_i^{eq} - f_i^f) \tag{2.106}$$

$$f_i^f = f_i^{eq} + scale^{c\to f}(f_i^{eq} - f_i^c). \tag{2.107}$$

It should be noted that these formulas are valid for the pre-collision state. All incomplete stencils are restored with rescaled information from the counterpart of the corresponding grid levels. This is depicted in Figure 2.3 for both scaling directions.

Since there are interface nodes on the fine grid with no partner node on the coarse grid level, spatial interpolation is necessary in addition to the rescaling. For the approximation of these stencils, a third order polynomial is used for interpolation [64]. Hence, four interpolation partners are needed on the coarse grid level. The spatial interpolation is illustrated in Figure 2.4.



(a) Coarse to fine (b) Fine to coarse

Figure 2.3: Scaling between overlapping grid levels

Finally, since the Mach number is constant due to the acoustic scaling, it follows that $\Delta t^c = 2\Delta t^f$ from eq. (2.99). Therefore, additional interpolation in time is needed for the fine interface nodes. Figure 2.5 exemplifies the assembled interface algorithm. The distributions at $t + \Delta t^f$ are interpolated from the pre-collision distributions of the coarse level at t and $t + \Delta t^c$, respectively.

Due to the fact that the fine nodes do more transport and collide operations, the grid refinement algorithm must be implemented in a nested time stepping manner. In order to be consistent in all levels, the simulation results for post processing must, therefore, be taken from the time level of the coarsest grid used. Some additional details on the interpolation for the interface algorithm may be found in [10, 21].

Instead of the scaling and interpolation of the distributions, the interface treatment

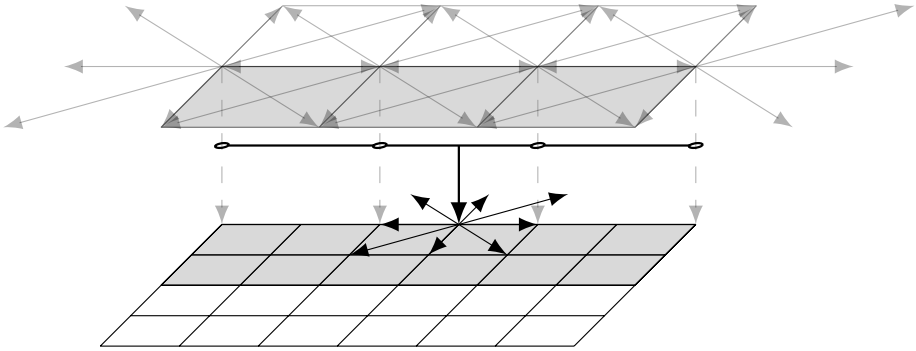


Figure 2.4: Spatial interpolation at fine interface border

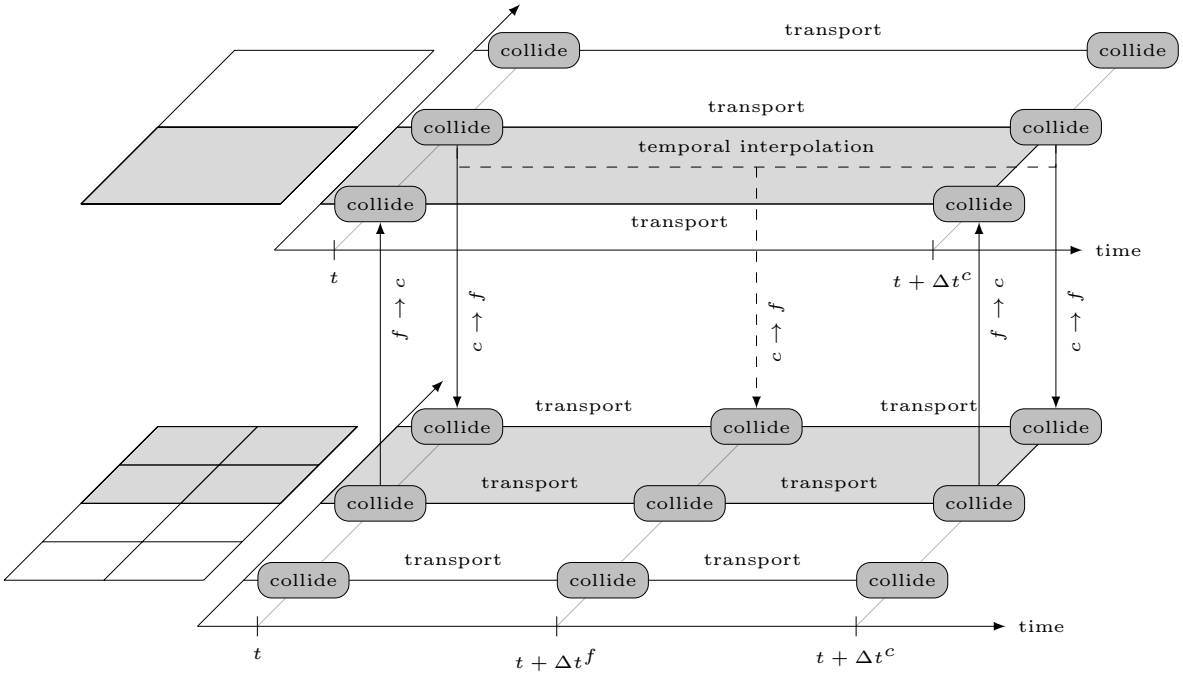


Figure 2.5: Overview of the interface algorithm

can be done with the moments of the distributions directly. This alternative approach is also implemented in *SamSolver*. Details on this treatment can be found in [56].

2.5.3 Generalized Newtonian Flow

In contrast to Newtonian fluids, where the molecular viscosity is assumed to be constant for isothermal flows, the viscosity of a non-Newtonian fluid varies with the local strain rate. A common approach is the power-law model by Ostwald de Waele [29]

$$\tau_w = k \ |\boldsymbol{\epsilon}|^n \, . \tag{2.108}$$

Combined with the linear relationship for wall shear stresses of Newtonian fluids, the effective kinematic viscosity of a generalized Newtonian fluid becomes

$$\nu = \frac{k}{\rho} \, |\boldsymbol{\epsilon}|^{n-1} \, , \tag{2.109}$$

where k is the flow consistency index $[Pas^n]$ and n is the dimensionless flow behavior index. Fluids with $n < 1$ are called *pseudo plastics (shear thinning)* and those with $n > 1$ are called *dilatants (shear thickening)*. Note that k can be identified as the dynamic

viscosity for Newtonian fluids, where $n = 1$. These parameters are fluid dependent and must be determined through curve fitting of experimental data. Incorporation of non-Newtonian behavior can be done by substitution of the kinematic viscosity in eq. (2.79) with the effective viscosity related to the magnitude of the strain rate tensor according to the constitutive law used. One of the major advantages of the lattice Boltzmann method is the ability to evaluate the stress and strain tensors locally with second order accuracy [37]. The magnitude of the strain rate tensor is calculated through the second invariant,

$$|\boldsymbol{\epsilon}| = \frac{\Omega}{\Delta t \cdot \rho \cdot c_s^2} \sum_{\alpha\beta} \sqrt{\frac{1}{2} \cdot \Pi_{\alpha\beta}^{neq} \Pi_{\alpha\beta}^{neq}} \qquad \forall \alpha \neq \beta. \tag{2.110}$$

The kinematic viscosity is then computed from eq. (2.109). Technically speaking, this expression cannot be evaluated directly because the kinematic viscosity itself appears in the collision frequency eq. (2.79). For an exact solution, the root of this implicit equation has to be computed. However, to keep the algorithm simple, the viscosity from the previous time step is used to compute the collision frequency in eq. (2.110), which has proved to be a good approximation.

As already stated in the beginning of this chapter, the LBM suffers from instabilities when the viscosity gets low. Hence, for the case of shear thinning fluids, the viscosity obtained through eq. (2.109) must be limited with a lower bound in regions with vanishing magnitude of the strain rate tensor. Similarly, it is necessary to introduce an upper bound on ν [22]. The upper bound was introduced in [22] with the very general rationale to ensure hydrodynamic behavior. As will be discussed in Chapter 7, the upper bound is needed to limit the error E_Ω introduced in Section 2.4.1. In the case of the power-law model, the effective viscosity becomes

$$\nu = \begin{cases} \nu_{max} & \epsilon > \epsilon_{max} \\ \frac{k}{\rho} |\epsilon|^{n-1} & \epsilon_{min} \leq \epsilon \leq \epsilon_{max} \\ \nu_{min} & \epsilon < \epsilon_{min} \end{cases}. \tag{2.111}$$

This effectively results in a lower bound Ω_{min} and an upper bound Ω_{max} of the dimensionless collision frequency. Therefore, the stability criteria are given by:

$$\Omega = \begin{cases} \Omega_{max} = \dfrac{c_s^2 \Delta t}{\nu_{min} + 0.5 c_s^2 \Delta t} \\ \Omega \\ \Omega_{min} = \dfrac{c_s^2 \Delta t}{\nu_{max} + 0.5 c_s^2 \Delta t} \end{cases} \tag{2.112}$$

In our simulations, we use an upper bound of $\Omega_{max} = 1.965$ for the SRT model and $\Omega_{max} = 1.985$ for the MRT model. The choice of Ω_{min} is addressed in Chapter 7. When not stated otherwise, the value of Ω_{min} is set to 1.0.

2.5.4 External Force

There exists an extensive overview on the different force implementations by *Guo et al.* [25], who analyzed the proper way to implement spatially and temporally varying forces that include discretization effects by *a posteriori* matching of the forcing term to the macroscopic equations. The incorporation of external forces $\mathbf{f}^F = \rho\mathbf{a}$ is achieved by adding an additional forcing term to the LBE,

$$f_i(\mathbf{x} + \mathbf{e}_i\Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega \cdot (f_i^{eq}(\mathbf{x}, t) - f_i(\mathbf{x}, t)) + \Delta t f_i^F. \tag{2.113}$$

The discretized forcing term f_i^F is expressed similar to the series expansion of the equilibrium distribution eq. (2.27) and reads

$$f_i^F = w_i \left(\frac{\Omega\nu}{c_s^2\Delta t}\right) \cdot \left(\frac{\mathbf{e}_i \cdot \mathbf{f}^F}{c_s^2} + \frac{1}{2c_s^4}(\mathbf{e}_i\mathbf{e}_i - c_s^2\mathbf{I}) : (\mathbf{u}\mathbf{f}^F + \mathbf{f}^F\mathbf{u})\right). \tag{2.114}$$

The same result can be obtained *a priori* through hermite series expansion of the forcing term. Since forces must not increase mass, the zeroth moment of f_i^F is zero. It should be noted that due to the presence of external forces, the first moment is altered. As can be seen from eq. (2.78), eq. (2.76), and eq. (2.22), the change of variables causes [43]:

$$w_i \sum_i \mathbf{e}_i \overline{f_i} = \rho\mathbf{u} - \frac{\Delta t \mathbf{f}^F}{2}. \tag{2.115}$$

Since the equilibrium function is given in terms of the physical velocity $\mathbf{u}$, the computation of f^{eq} must be done with

$$\mathbf{u} = \frac{1}{\rho} w_i \sum_i \mathbf{e}_i \overline{f_i} + \frac{\Delta t \mathbf{f}^F}{2}. \tag{2.116}$$

With this scheme, additional non-physical terms involving space and time derivatives of the forcing term in the macroscopic equations get canceled. Hence, in the case of a spatially constant body forces for which the time derivative is small or zero, the forcing term is often simplified to

$$f_i^F = w_i \cdot \left(\frac{\mathbf{e}_i \cdot \mathbf{f}^F}{c_s^2}\right). \tag{2.117}$$

This applies to the SRT model. The same simplified forcing term is easily incorporated in the MRT model by addition of the corresponding momentum due to the body force on the macroscopic moments for the momentum in Table 2.5 after the collision step.

$$\begin{aligned} m_3^{post} &= m_3^{pre} + \Delta t f_x^F \\ m_5^{post} &= m_5^{pre} + \Delta t f_y^F \\ m_7^{post} &= m_7^{pre} + \Delta t f_z^F \end{aligned} \tag{2.118}$$

For the implementation details of the fully discrete forcing scheme in the case of the MRT model, the reader is referred to [26].

CHAPTER 3

Implementation

This chapter is concerned with some theoretical aspects of the implementation of *SAM-Lattice*. Details on the actual programming may be found in the programmers documentation to *SamGenerator* and *SamSolver*, respectively.

3.1 Lattice Generation

One part of any form of computational fluid dynamics is the spatial discretization of the geometry that will be subject to the simulation. This is the task of the program *SamGenerator*. The concept of Lattice Boltzmann calls for an equidistant lattice that can be represented by a joint set of hexahedra cells. Such an representation is known in computer graphics as a voxel (neologism of volumetric and pixel) representation. Starting point of the discretization, i.e, the input file for *SamGenerator* is a triangulated representation of the boundaries of the geometry. Each triangle in the triangulation is defined by three points and a normal vector. By definition, the normals point out of the fluid domain. The requirements for a valid *SamGenerator* input file is met by either a modified *stl* (stereolithography) format or a boundary mesh, e.g, in *msh* format, consisting of triangle elements only. Note, that the standard *stl* format does not offer possibilities to assign triangles to surfaces of the geometry.

SamGenerator is octree based, which allows for local grid refinement, where a coarse cell is decomposed in eight child cells. In that sense, each octree level is assigned a number, for which the coarsest discretization is on level 0. Figure 3.1 shows a flowchart of *SamGenerator* with the most significant program routines. They are grouped by their functional affiliation.

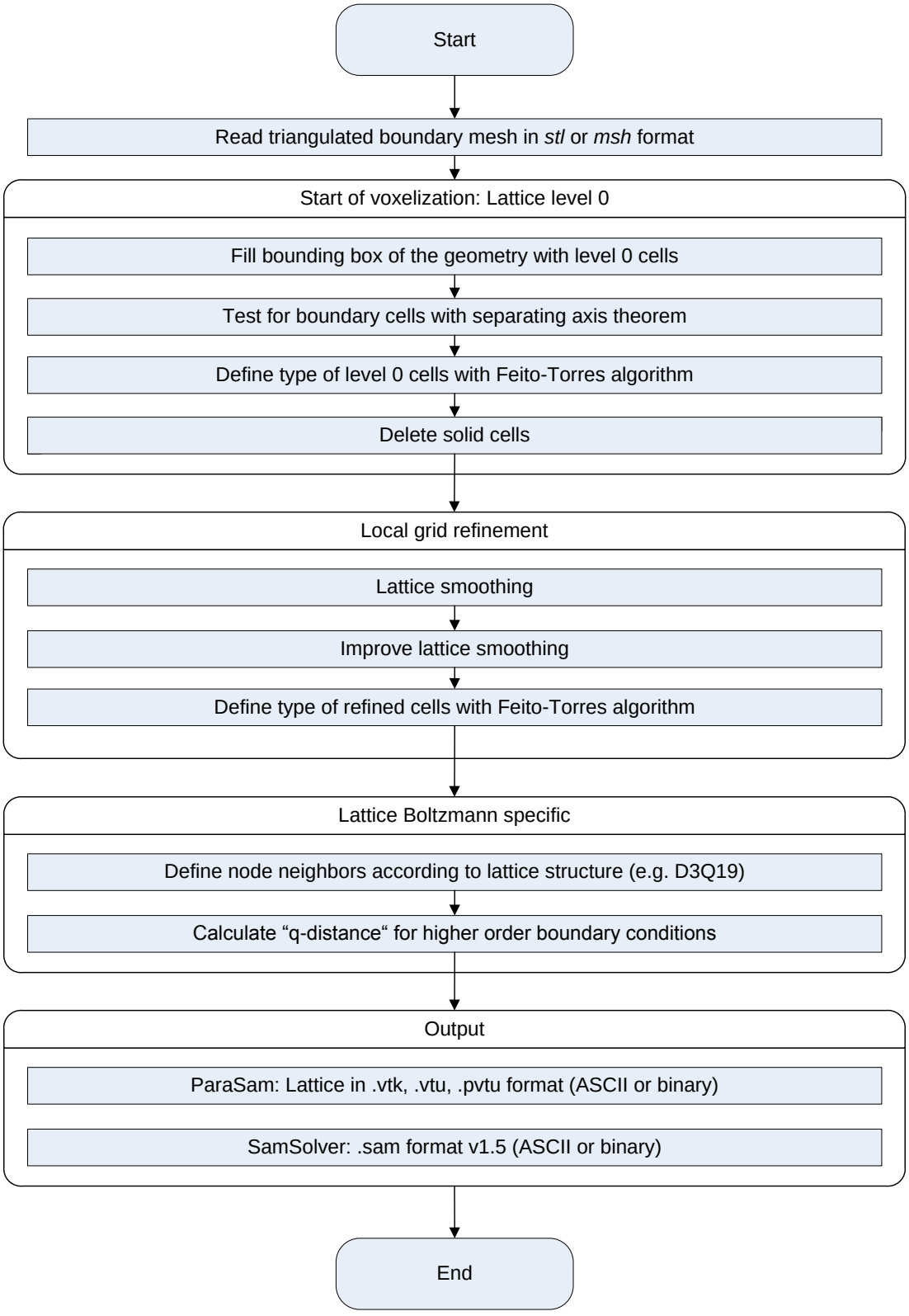


Figure 3.1: Flowchart of *SamGenerator*.

Upon the start of *SamGenerator*, the input file is read and stored internally in containers provided by the C++ standard template library. Discretization begins by filling the bounding box of the fluid domain with hexaedra with edge length according to the desired lattice spacing in level 0. In the next step, we have to decide, which cells are inside the fluid domain. To this end, we use a combination of the separating axis theorem and a point inclusion test algorithm by *Feito* and *Torres*, which will be discussed in Section 3.1.1. The separating axis theorem is used to filter those cells intersecting the boundary of the geometry, that is, we test if the boundary triangle and the lattice cell overlap. Since the LBM is node based, we then need to test if the nodes, which the boundary cells are composed of, lie inside or outside of the fluid domain, i.e., if a node is a fluid node or a wall/boundary node. This is done with the Feito-Torres algorithm. Based on the node types in the tested boundary cells, we use a flood fill procedure to separate cells that completely lie in the fluid domain from those cells located completely inside an obstacle or outside the fluid domain (solid cells). The latter can be deleted to save valuable memory.

If desired, the level 0 lattice can be locally refined. Objectives of the refinement can be either refinement of the boundary discretization or local refinement of an area inside of the fluid domain. Local domain refinement is implemented in *SamGenerator* in two ways. On one side, a box, defined by an origin and its extension in three dimensions, can be used and on the other hand, areas of refinement can be arbitrarily shaped through usage of an auxiliary geometry in *.stl* or *.msh* format. Both approaches need to be supplied with the desired refinement depth, i.e., the number of the octree level for the local grid refinement. Due to stability issues at the interface, neighboring lattice levels must not differ by more than one lattice level [10]. Therefore, the lattice is smoothed in regions, where this condition is violated. In some cases, the result of the smoothing can be improved. A more detailed discussion on lattice smoothing is done in the sequel. Especially in the case of boundary refinement, the resulting higher level cells and corresponding new nodes need to be tested for their type. This is again done with the Feito-Torres algorithm.

So far, the previously described discretization gives a three dimensional lattice with an staggered boundary representation. Therefore, an additional tool was written, which makes it possible to export the lattice in the *CGNS* format in order to use it in a classical Navier-Stokes solver. An additional step is needed to make the lattice boundary conform. For use with the Lattice Boltzmann method, the appropriate neighbor relationships must be set according to the velocity discretization model used, e.g, D3Q19. In order to incorporate the actual geometry into the simulation, as opposed to its staggered representation, the intersections of the lattice links and the geometry is calculated. A dimensionless distance from the fluid nodes to the geometry, known as "q-distance", is introduced, which will be discussed in Section 3.1.3.

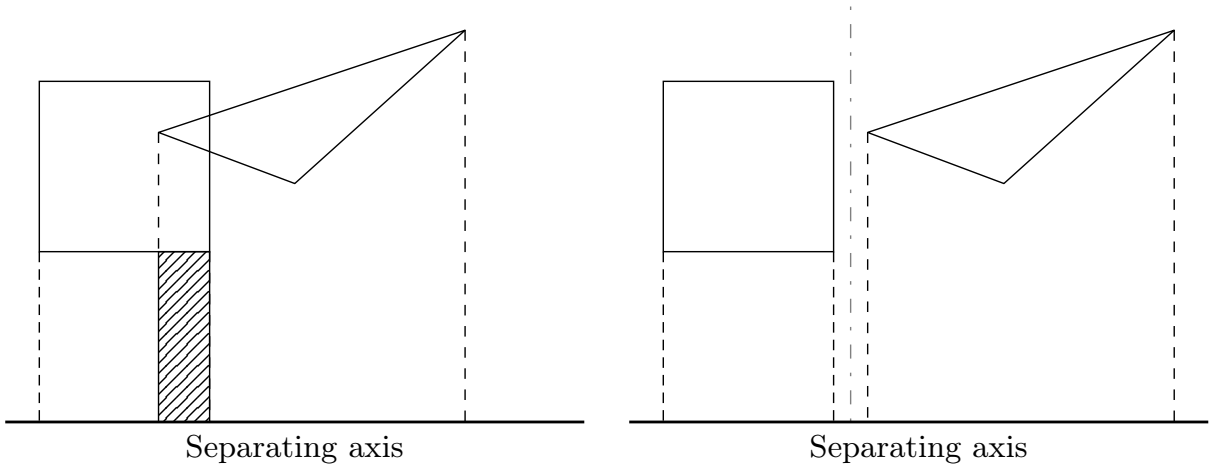
Result of the whole discretization process is the voxelized geometry in one of the *VTK* (Virtual Tool Kit) formats, readable by *ParaSam*, and/or the .sam file, which serves as

the input file for *SamSolver*. Both resulting files can be in ASCII or (raw) binary format, where the latter is preferred, due to higher performance and small storage requirements.

3.1.1 Generation Algorithms

Separating Axis Theorem

We borrow some ideas used in computer graphics and apply these during lattice generation. The task of deciding if two bodies intersect, is common in a variety of applications, e.g., collision detection in the logic of a computer game. A way to do that is the hyperplane separation theorem or also known as the separating axis theorem (SAT). This theorem states that two bodies are disjoint if an axis can be found, for which the projections of the bodies on that axis are two disjoint sets. This is illustrated in Figure 3.2.



(a) Collision (b) No collision

Figure 3.2: Example of collision detection using the separating axis theorem.

The projection of the triangle and the cuboid is done along a line perpendicular to the separating axis. In the first case, there is a collision of the bodies, because the projections on the separating axis do overlap, which is depicted by a dashed area. This is also the case for all other choices for a separating axis. In Figure 3.2(b) the projections do not overlap and, therefore, the bodies are disjoint. Since the theorem does only make statements about the existence of a separating axis, a suitable set of projection directions must be chosen. Note that in three dimensions, projection of the bodies into the plane must be conducted

prior to the projection on a separating axis. In its basic form, the separating axis theorem is only valid for convex bodies. Fortunately, this is enough in our case, since we exclusively deal with hexahedra and triangles. In our work, we rely on a fast box-triangle algorithm derived in [1]. The algorithm consists of 3 overlapping tests for the bounding boxes, one overlapping test for the triangle normal, and 9 overlapping tests for the triangle edges.

For reasons of simplicity, the hexahedron is transformed into the origin, such that it is axis aligned with the global coordinate system and the triangle is transformed accordingly. On the first 3 tests, a axis aligned bounding box around the triangle is tested against the cuboid, while projecting on the principal axes. Next, a plane-bounding box overlap test from [27] is used with the normal vector of the triangle. Finally, each of the three edges of the triangles are used to project the triangle and the box on each of the principal axes, which gives 9 tests in total. The algorithm is terminated, if a separating axis could be found in one of the projections. This is the reason why the SAT algorithm is very fast, since it is possible to find a separating axis early within the few first tests.

The algorithm is used in the course of the discretization to check if a cell and the triangulated geometry intersect. In that sense, we want to determine if a cell is a boundary cell, e.g., inlet or wall boundary, for which boundary conditions must be provided.

In order to save computational work, we loop over STL triangles and only test those hexahedra that lie in the bounding box of the triangle. That way, a large part of the cells can be left out of the SAT test for the particular triangle. Those cells can easily be calculated based on their index in the lattice.

Feito-Torres Algorithm

Feito and *Torres* presented a planar point inclusion algorithm in [16], which does not resort to solving equation systems or use of trigonometric functions. Hence, the test does not suffer from stability problems. The idea of the inclusion test, is to decompose the test polygon into sub-polygons involving the test point I. The orientation of the sub-polygons is used to decide, whether the test point lies inside or outside of the polygon. In the case of a triangle with the nodes ABC, which are arranged to give a positive orientation, the test point I is checked for inclusion by decomposing the triangle into three sub-triangles IAB, IBC, and ICA. The orientation of the triangles can be computed via the corresponding determinants. The point I lies inside the triangle if, and only if, all three sub triangles have the same orientation as the geometry triangle ABC, i.e., if all determinants are positive.

The same authors presented a generalization of their algorithm to 3 dimensions in [17], where we have to compute determinants of tetrahedra.

3.1.2 Grid Refinement and Lattice Smoothing

As already mentioned, *SamGenerator* is octree based, which allows for local grid refinement, where a coarse cell is decomposed in eight child cells. Each octree level is assigned a

number, for which the coarsest discretization is on level 0. An example is shown in Figure 3.3, where a level 0 cell is refined with eight level 1 cells and one level 1 cell is refined with eight level 2 cells. Therefore, the spacing in level 2 is four times smaller than in level 0. Although the configuration shown in Figure 3.3 may be realized, it is generally not recommended to have such a sharp transition of grid levels. This is often a source for instabilities emerging from the interpolation at the interface [10, 54].

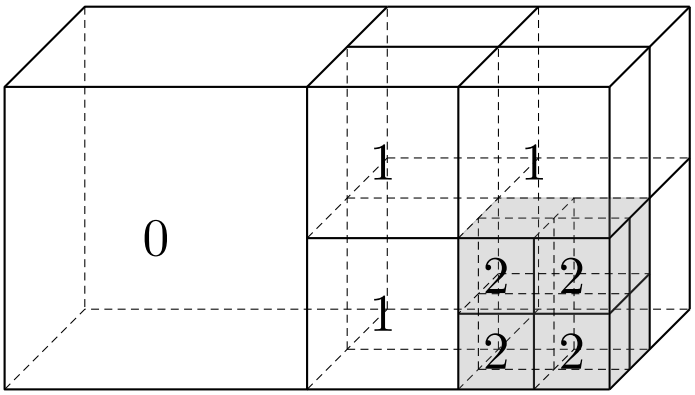
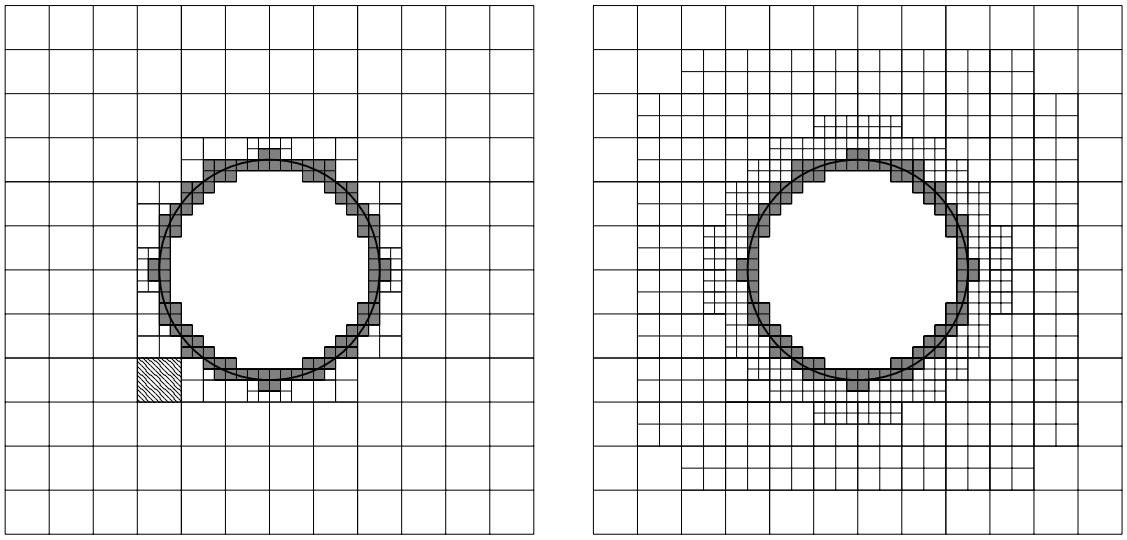


Figure 3.3: Octree refinement

Therefore, neighboring lattice levels must not differ by more than one lattice level. In fact, in *SamGenerator*, we restrict the minimum number of cells necessary before changing lattice levels to three at boundaries and to two for intermediate lattice levels. Figure 3.4(a) shows a slice of the intermediate result of a level 2 discretization for a flow around a sphere, where the lattice is refined towards the surface of the sphere. The cells intersecting the geometry are colored in gray. This intermediate result is an example, where the requirement at the interface is violated. There is a shaded level 0 cell, which is next to level 1 and level 2 cells. The discretization is enhanced by the smoothing algorithm implemented in *SamGenerator*. On the picture to the right, Figure 3.4(b), the smoothed lattice is shown. One can see, that the number of level 2 cells at the boundary is at least three and the number of cells in level 1 is at least two.

Starting from the level 2 boundary cells, the smoothing algorithm traces each possible lattice direction and refines the cells up to the corresponding lattice level, until the minimum number of cells is reached. Depending on the geometry, the smoothing procedure can produce valid results, for which the local amount of interpolations can be reduced by further refinement. This may be the case for overlapping refinement patches at neighboring boundaries. Some more exclusive examples are given in [10]. Since further refinement can



(a) Intermediate refinement (b) Smoothed refinement

Figure 3.4: Effect of lattice smoothing implemented in *SamGenerator*.

in turn produce cases, where additional refinement can be advantageous, the improvement of the lattice quality by smoothing can be considered as an iterative process. Therefore, a maximum number of improvement cycles is introduced.

The smoothing algorithm can be tuned by the number of improvement cycles as well as the number of smoothing layers, which are free parameters implemented in *SamGenerator*.

3.1.3 Higher Order Schemes

Based on the discretization shown in Figure 3.4, the surface of the sphere is represented by a staggered lattice. This ultimately degrades the LBM to a first order accurate scheme. This will be discussed in Section 3.2.2. Since the LBM is second order accurate, we need boundary conditions that are at least second order accurate as well. Therefore, a more precise approximation of the geometry is needed. In order to incorporate the curvature of the sphere, i.e., the real position of the geometry, into the simulation, the intersections of the lattice links with the surface of the geometry must be computed for the boundary cells. This is shown in Figure 3.5 for a two dimensional view of a boundary cell.

In this situation, the wall node is depicted as a filled black dot. Because of the lattice structure, the length of the lattice links to the surrounding fluid nodes differ by a factor of $\sqrt{2}$. Therefore, the distance to the wall, i.e., that part of the lattice link that lies inside the fluid domain, is measured by the axis aligned distance $q^{'}$. That way, a dimensionless

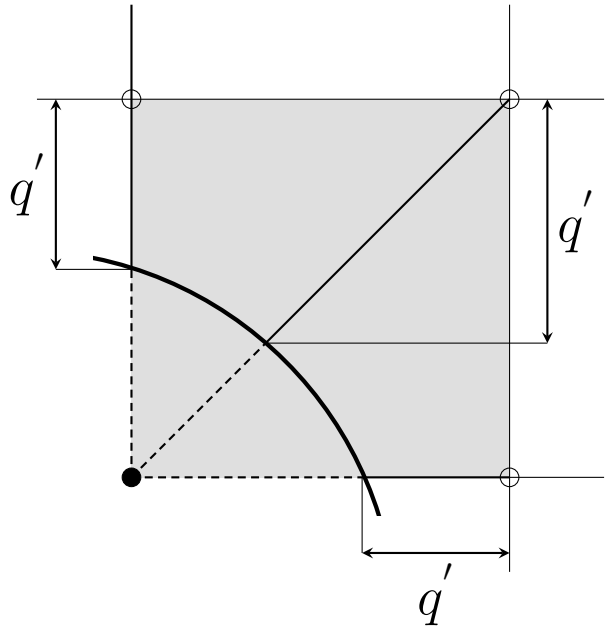


Figure 3.5: Illustration of the dimensionless wall distance q.

wall distance q can be introduced,

$$q = \frac{q^{'}}{\Delta x}, \tag{3.1}$$

which is independent of the length of the lattice links through normalization with the lattice spacing. This is known as the "q-distance" throughout the literature.

Although these distances may be calculated analytically in this exemplary case of a flow around a sphere, the general way is to compute the intersections of the lattice links with the triangulation of the geometry surface. Figure 3.6 illustrates such a situation, where only three lattice links are shown for the sake of simplicity.

The problem of computing the distance $q^{'}$ can be solved with methods of analytic geometry. All intersecting triangles per boundary cell are already known from the separating axis overlapping tests. Hence, we loop over all boundary cells and compute the intersection of the lattice links of wall nodes with each triangle of the geometry. To this end, the lattice links are cast in a straight line equation and inserted into a plane equation, which is given by the triangle normal and one of the triangle nodes. With this approach, the computed intersection point may not necessarily be the point that actually intersects the geometry. This is shown in Figure 3.6 for the rightmost lattice link. The triangle under consideration is the triangle to the left and the intersection with the corresponding plane is computed. It is obvious, that an additional test is needed, since the actual intersection of the lattice

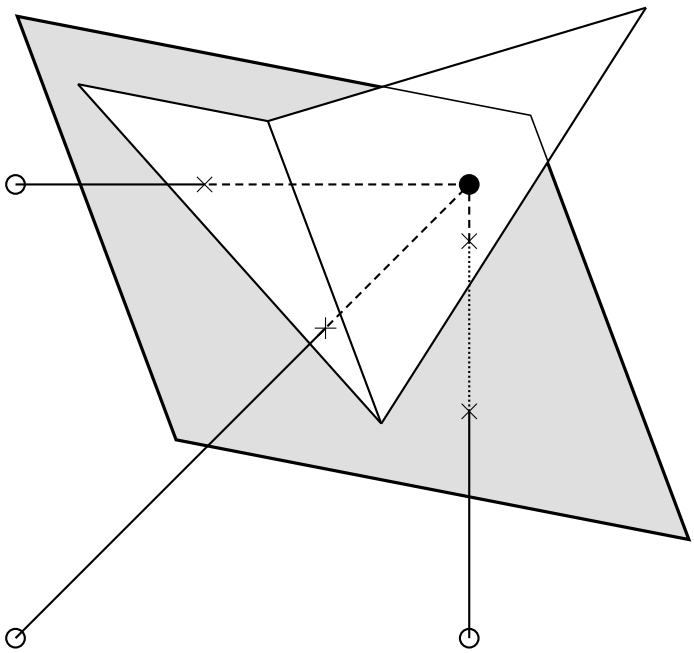


Figure 3.6: Calculation of the dimensionless wall distance q.

link with the geometry happens for the triangle on the right.

The following test is used to determine if the intersection point lies inside the triangle: We interpret the three edges of the triangle as planes perpendicular to the triangle plane, that is, planes in Hesse normal form that are defined by a normal vector perpendicular to the triangle edge and one of the edge nodes. The three nodes of the geometry triangle are arranged in a mathematically positive sense. Thus, we can compute a vector perpendicular to the triangle edge that points outside of the triangle. These vectors serve as the normal vector in the Hesse normal form of the three edge-planes. Now, it is easy to compute the distance of the intersection point to one of the three planes.

The point lies inside the triangle if, and only if, all three distances of the intersection point to the edge-planes are negative, since the normal vectors point outside of the triangle. This test is fast, because similar to the box-triangle algorithm, the test can be terminated as soon as one of the distances is positive.

3.2 Solver Implementation

The modular structure of the program package *SAM-Lattice* manifests itself in the various output files generated by the individual modules. The advantage of this concept is a great flexibility, because different modules can act on the input/output file of an other. *SamGenerator* generates the *.sam* file, which can either be used directly as the input file for *SamSolver* or as the input file for *SamDecomposer*. A slight disadvantage of this approach is the I/O overhead and the associated space requirements. Although this will not be discussed in detail, the application *SamDecomposer* can be used to decompose the lattice on the basis of the graph partition library *MeTiS*. Therefore, the *.sam* file serves as the input for *SamDecomposer*, which results in a series of *.samd* files, containing the decomposed parts of the lattice and numbered according to their processor affiliation. Moreover, *SamSolver* is parallelized in shared memory (SMP) architectures through usage of the *OpenMP* library [8, 31].

Figure 3.7 shows a flowchart for *SamSolver* with the most important program routines. In addition to the lattice information in the *.sam* file, there is a solver control file. This is a text file, which contains information of the various boundary conditions associated with the corresponding boundary in the lattice. Most boundary conditions implemented in *SamSolver* will be discussed in Section 3.2.2. Moreover, general settings, e.g., the duration of the simulation, write interval, etc., are stored in the control file. Upon the start of *SamSolver*, the *.sam* file and the control file are read and the data is stored internally in the various variables and C++ containers. The *.sam* file contains information on the interface connectivity for lattices with local grid refinement. A setup routine is implemented, which sets the corresponding relationships at the interface for the spatial and temporal interpolation. In the preprocessing for MPI runs, the corresponding send/receiver relationships for the inter-processor communication is set up. Before the start of a simulation, the distributions must be initialized. This is done by calculating the equilibrium distributions according to a prescribed global value for the velocity and the density, respectively. Additionally, an advanced initialization procedure, proposed in [44], is implemented in *SamSolver* to generate a pressure field that is consistent with the velocity field.

After the preprocessing, the solver is started with different configurations depending on whether the lattice is refined or not. Moreover, there is a different treatment in the collide routine. The simulation can be carried out with the SRT or the MRT collision model as well as it can be laminar or turbulent and non-Newtonian. We use a Large Eddy Simulation (LES) approach for modeling of turbulent flows. An intensive investigation and application of this feature of *SamSolver* can be found in the PhD thesis of *Schneider* [56].

This flexibility in terms of the different solvers and models is internally realized through usage of C++ functors. In the single level case, the solver enters a time loop which controls the duration of the simulation. According to the Lattice Boltzmann algorithm, the collide and transport step are conducted for each individual time step.

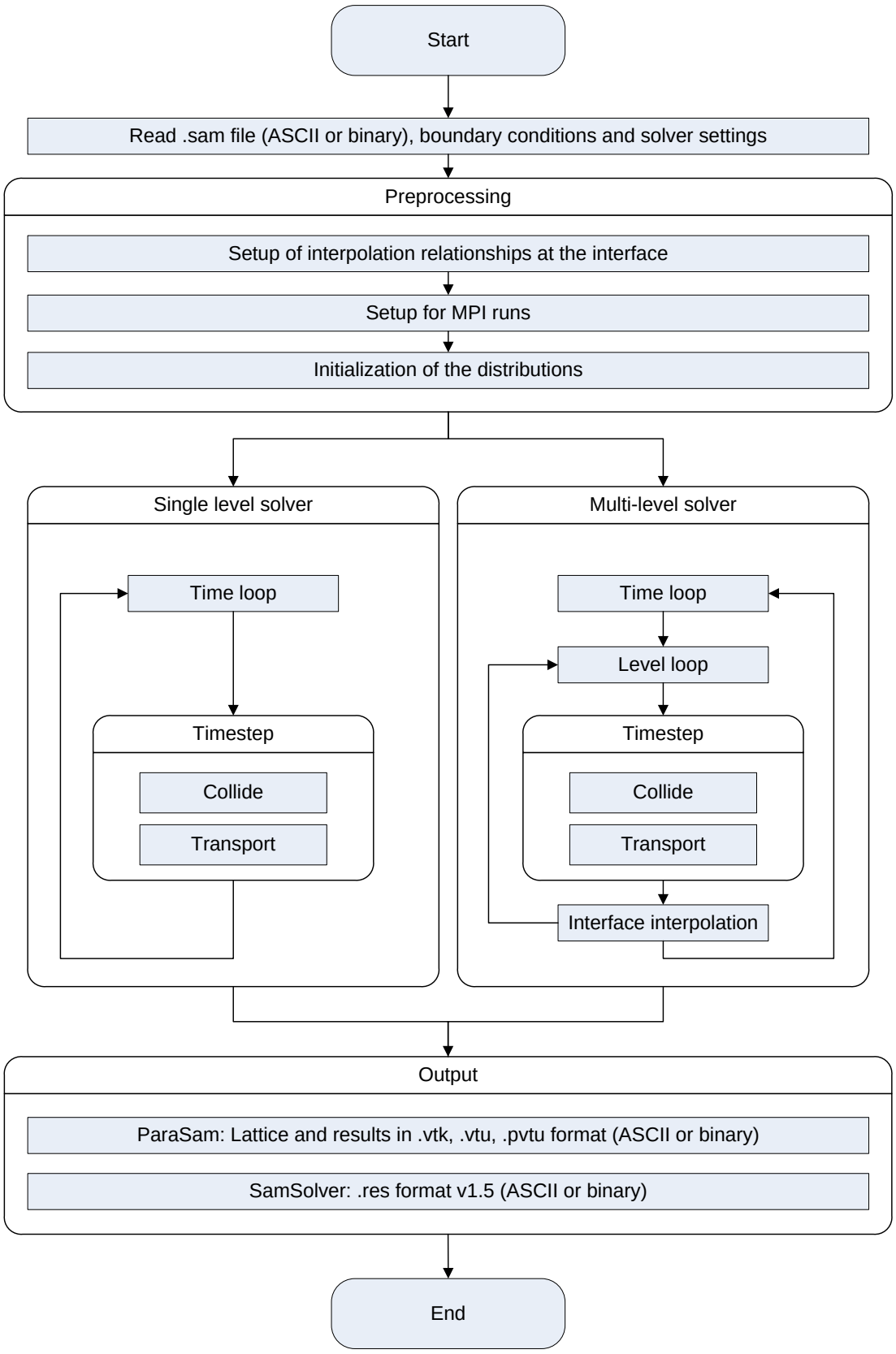


Figure 3.7: Flowchart of *SamSolver*.

Additionally, there is a command interpreter implemented in the class *SamCommandInterpreter*, which is nested in the outer time loop. This makes it possible to control the solver through editing of the *.sam_ cmd* file, while the simulation is running. Actions range from a simple change of the simulation duration over modification of boundary conditions, to change of the number of processor threads in shared memory parallelized runs. The various commands can be found in the programmers documentation.

For the multi-level treatment, there is an additional loop for the different octree levels, which will be discussed in Section 3.2.1. This level loop is nested inside the time loop.

At prescribed write intervals or at the end of the simulation, *SamSolver* produces various output files depending on the settings. Simulation results such as velocity, pressure, viscosity, etc., are written in the partitioned vtu format *.pvtu*, which is the master file for assembling of different part files in the *.vtu* format. That way, it is possible to realize parallel output in SMP runs or each MPI process outputs its part file. In both cases, the results can be in ASCII or (raw) binary. There is also the possibility to directly output the distributions of each lattice node in the *.psam* file and calculate the macroscopic moments with our extension to *ParaView*: *ParaSAM*.

Finally, there is a *.res* file, which freezes the current distributions and settings to resume the simulation at a later point in time.

3.2.1 Multi-Level Treatment

In this subsection, the implementation of the multi-level treatment is discussed. Basis of the implementation is the need for a nested time stepping. Our grid refinement algorithm relies on the acoustic scaling $\Delta t \propto \Delta x$. Owing to the octree structure, the spacing Δx is always halved on the transition to a finer resolution. Hence, the time step on an octree level n is twice the time step on a level $n + 1$.

Listing 3.1 shows the C++ style pseudo code for the multi-level solver. In lines 1-4 some control variables are defined, where *maxLevel* gives the maximum number of octree level as shown in Figure 3.3.

Depending on the number of octree levels, a series determining the sequence in which the nodes on the corresponding levels perform a collide and transport step can be derived. Therefore, the array *levels* of length *maxLevel* stores the series of the form $2^{maxLevel-octreeLevel}$. It is computed in line 6. There are two member functions in the listing. The member function *timeStep*, implements the collide and transport step for the corresponding level passed to it in parenthesis. Furthermore, the member function *interpolateInterface* implements the interpolation in space and time for the interface nodes of the corresponding levels. Besides the rescaling to the fine level, the distributions have to be interpolated in space for nodes, where there is no colocated node on the corresponding coarse grid. For odd times, the values have to be interpolated in time as well. For even

times, at the end of the time step, the values have to be scaled back to the coarse grid nodes at the interface. This is summarized graphically in Figure 2.5.

A multi level time step starts with the collision and transport for nodes in the coarsest level 0. For subsequent levels, the nested time stepping is realized with a loop over the levels for which the execution condition has to be checked. This condition is satisfied, if the time variable is divisible without remainder by the interval given by the series formula. It is clear, that the finest level (maxLevel) must collide and transport in every time step. This can also be seen from the series formula above. Hence, there are always two time steps in the maximum octree level, where interpolation in time is done in the first of the two steps as already explained above.

Listing 3.1: C++ style pseudo code for multi-level solver

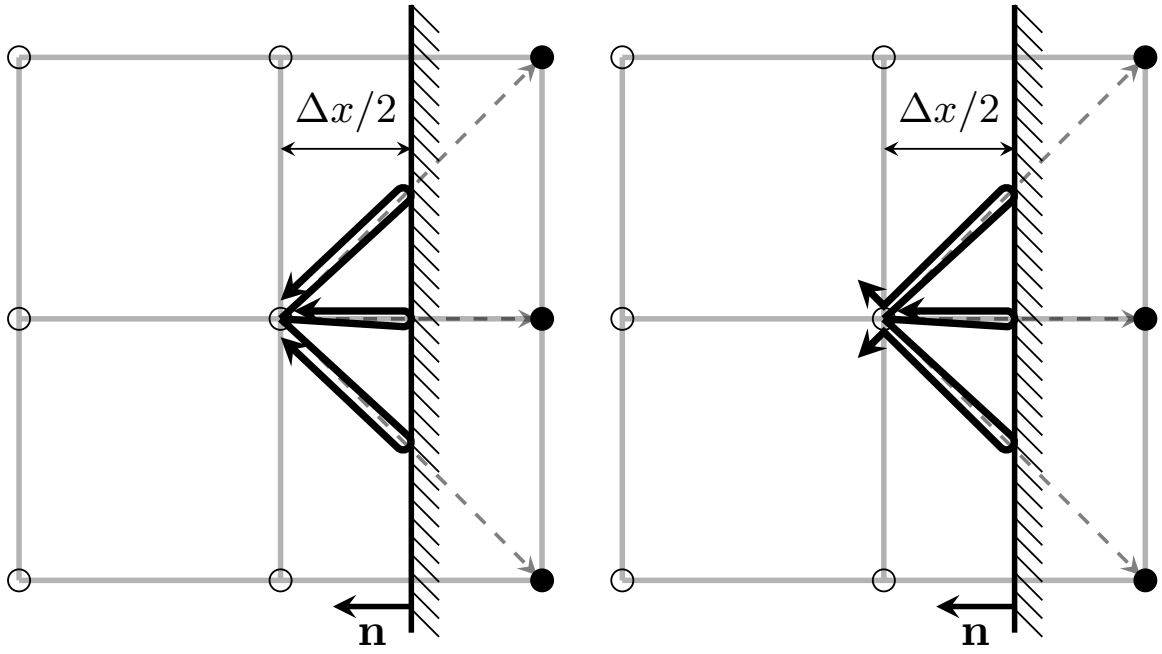
```
time=0;
level=0;
levels[maxLevel];
end=pow(2.,maxLevel);

for (i=0;i<maxLevel;i++) levels[i]=pow(2.,maxLevel-i);

timeStep(0); // Timestep: collide and transport for level=0

while (time!=end)
{
 for(level=1;level<maxLevel;level++)
 {
  if ((time % levels[level])==0)
  {
   timeStep(level);
   interpolateInterface(level); //interpolate interface nodes
  }//End if mod
 }//End for level

 //Do maxLevel
 for(subcircles=0; subcircles<2; subcircles++)
 {
  timeStep(maxLevel);
  interpolateInterface(maxLevel); //interpolate interface nodes
 }//End for subcircles

 time+=2; // Increase time variable
} // End while
runTime++; // Advance run time on level 0
```

The whole level loop is repeated until all levels have reached the time level of level 0, before the run-time is advanced by an outer time loop as it is depicted for the multi-level solver in Figure 3.7.

3.2.2 Boundary Conditions

Simple Bounce Back Schemes

One of the features of the LBM is the possibility, that boundary conditions can be modeled on a mesoscopic basis. In that sense, the simplest boundary condition is the molecular reflection on solid walls. This no-slip wall boundary condition is known in the Lattice Boltzmann community as the bounce back scheme. A sketch of this scheme is shown in Figure 3.8(a), where the wall normal points inside the fluid domain and the wall nodes are denoted by filled black circles. At time t, the dashed distribution are supposed to be



(a) Bounce back (b) Frictionless

Figure 3.8: Transport step for modeling a no slip wall (a) and a frictionless wall (b). Dashed: Time t, solid: Time $t + \Delta t$.

transported to the wall nodes in the corresponding directions at time $t + \Delta t$. Because of the solid wall, the distributions are reflected ("bounced back") to the starting node and

point to the corresponding counter direction of the wall nodes at time $t + \Delta t$. This is illustrated with black arrows. Formally this scheme is given by

$$f_{i^-}(\mathbf{x}, t + \Delta t) = f_i(\mathbf{x}, t) \quad \text{with} \quad \mathbf{e}_{i^-} = -\mathbf{e}_i. \tag{3.2}$$

A direct consequence of this treatment is the positioning of the wall at $\Delta x/2$, since the space-time integration is bound to the values for the step sizes Δt and Δx. More pictorially, the molecules with their corresponding molecular velocities must travel the distance Δx in one time step with size Δt. It should be noted, that this form of the bounce back scheme is known as the "half way bounce back". In this case, the velocity vanishes exactly midway between the fluid and wall node at $\Delta x/2$. Therefore, an error of $\mathcal{O}(\Delta x)$ is introduced for walls positioned off-center, which reduces the accuracy of the LBM to first order in these cases [28]. It is also possible to model moving walls with this approach. An additional term is introduced by *Ladd and Verberg* [39], where the velocity of the wall $\mathbf{u}_w$ is used to compute the first order term of the equilibrium distribution. The form of this term emerges from the balance of momentum, which will be discussed in more detail in Section 3.2.3. The bounce back scheme for a wall moving with velocity $\mathbf{u}_w$ reads

$$f_{i^-}(\mathbf{x}, t + \Delta t) = f_i(\mathbf{x}, t) - 2 \cdot w_i \rho \cdot \frac{\mathbf{e}_i \cdot \mathbf{u}_w}{c_s^2} \quad \text{with} \quad \mathbf{e}_{i^-} = -\mathbf{e}_i. \tag{3.3}$$

In a similar manner, frictionless walls can be modeled with the simple bounce back scheme. To this end, the normal component of the velocity is reflected and the tangential component of the velocity is conserved. This is shown in Figure 3.8(b) with solid black lines and mathematically expressed in the following equation:

$$f_{i^-}(\mathbf{x}, t + \Delta t) = f_i(\mathbf{x}, t) \quad \text{with} \quad \mathbf{e}_{i^-,n} = -\mathbf{e}_{i,n} \quad \text{and} \quad \mathbf{e}_{i^-,t} = \mathbf{e}_{i,t}. \tag{3.4}$$

As in the no-slip case, the modeled wall is placed at $\Delta x/2$, because the normal velocity vanishes at this point.

Higher Order Bounce Back Schemes

Since the Lattice Boltzmann Method is second order accurate, several authors extended the wall treatment to incorporate the exact position of the wall into the simulation. Three of these approaches are discussed in the sequel. Common to all, is the usage of the dimensionless wall distance, as it was discussed in Section 3.1.3 with the difference that we do not restrict ourselves to flat walls anymore. Hence, the dimensionless wall distance becomes discrete in each lattice direction i as shown in Figure 3.5.

Bounce Back Filippova

Historically, the first wall boundary treatment for curved boundaries was presented by *Filippova and Hänel* in [19]. The idea of this approach is to compute a fictitious equilibrium at the surface of the wall with values for $\mathbf{u}$ from the fluid node $\mathbf{x}_f$ or extrapolated values for $\mathbf{u}$ on the wall node $\mathbf{x}_w$ depending on the dimensionless wall distance q_i. Values for ρ are always taken from the fluid nodes. The resulting equilibrium is then combined through linear interpolation with the distributions from the simple bounce back rule. According to the Chapman-Enskog analysis in [20], this scheme provides results with second order accuracy. As a result of a stability analysis, a case distinction in terms of the dimensionless wall distance q_i is made. The computation of the "bounced back" distributions can be summarized as follows [19, 28]:

Case: $q_i < 0.5$

- Compute $f_i^{eq}(\mathbf{x}, t)$ with $\mathbf{u} = \mathbf{u}(\mathbf{x}_f, t)$ and $\rho = \rho(\mathbf{x}_f, t)$.
- Compute weighting factor $\chi = \dfrac{\Omega}{1-\Omega} \cdot (2q_i - 1)$.
- Combine bounce back ($\mathbf{e}_{i^-} = -\mathbf{e}_i$) with f_i^{eq} according to:
  $f_{i^-}(\mathbf{x}_f, t + \Delta t) = (1-\chi) \cdot f_i(\mathbf{x}_f, t) + \chi \cdot f_i^{eq} - 2 \cdot w_i \rho \cdot \dfrac{\mathbf{e}_i \cdot \mathbf{u}_w}{c_s^2}$

Case: $q_i \geq 0.5$

- Compute $f_i^{eq}(\mathbf{x}, t)$ with $\mathbf{u} = \mathbf{u}(\mathbf{x}_f, t) \cdot \left(1 - \dfrac{1}{q_i}\right) + \dfrac{1}{q_i} \cdot \mathbf{u}_w$.
  and $\rho = \rho(\mathbf{x}_f, t)$.
- Compute weighting factor $\chi = \Omega \cdot (2q_i - 1)$.
- Combine bounce back ($\mathbf{e}_{i^-} = -\mathbf{e}_i$) with f_i^{eq} according to:
  $f_{i^-}(\mathbf{x}_f, t + \Delta t) = (1-\chi) \cdot f_i(\mathbf{x}_f, t) + \chi \cdot f_i^{eq} - 2 \cdot w_i \rho \cdot \dfrac{\mathbf{e}_i \cdot \mathbf{u}_w}{c_s^2}$

Bounce Back Mei

This bounce back descendant is a further improvement of the scheme proposed by *Filippova*. According to the authors, it is more stable, that is, it allows for higher values of Ω. This scheme was developed in 2D by *Mei et al.* [45]. An extension to 3D with further improvements was presented by the same authors in [46]. Based on the fictitious equilibrium approach, the linear interpolation is the same as for the *Filippova* scheme,

$$f_{i^-}(\mathbf{x}_f, t + \Delta t) = (1-\chi) \cdot f_i(\mathbf{x}_f, t) + \chi \cdot f_i^{eq} - 2 \cdot w_i \rho \cdot \frac{\mathbf{e}_i \cdot \mathbf{u}_w}{c_s^2} \quad \text{with} \quad \mathbf{e}_{i^-} = -\mathbf{e}_i. \tag{3.5}$$

A no-slip wall at rest is modeled by setting the wall velocity $\mathbf{u}_w$ equal to zero. The equilibrium function in eq. (3.5) is defined with separate velocities for the first and second order terms of the equilibrium [46]:

$$f_i^{eq} = w_i \cdot \rho(\mathbf{x}_f, t) \cdot \left(1 + \frac{\mathbf{e}_i \cdot \mathbf{u}_b}{c_s^2} + \frac{1}{2c_s^4}(\mathbf{e}_i\mathbf{e}_i - c_s^2\mathbf{I}) : \mathbf{u}(\mathbf{x}_f, t)\mathbf{u}(\mathbf{x}_f, t)\right). \tag{3.6}$$

Again, as in the previous scheme, the velocity at the boundary surface $\mathbf{u}_b$ and the interpolation parameter χ are a function of the dimensionless wall distance q. Moreover, a case distinction is used for the computation of these values, due to stability issues. Thus, the values for $\mathbf{u}_b$ and χ are chosen according to

$$\mathbf{u}_b = \begin{cases} \mathbf{u}(\mathbf{x}_f + \mathbf{e}_{i^-}\Delta t, t) & q_i < 0.5 \\ \left(1 - \frac{3}{2q_i}\right) \cdot \mathbf{u}(\mathbf{x}_f, t) + \frac{3}{2q_i} \cdot \mathbf{u}_w & q_i \geq 0.5 \end{cases} \tag{3.7}$$

$$\chi = \begin{cases} \frac{2q_i - 1}{\Omega - 2} & q_i < 0.5 \\ \frac{2q_i - 1}{\Omega + 0.5} & q_i \geq 0.5 \end{cases} \tag{3.8}$$

in the *Mei* scheme. It can be shown through a Chapman-Enskog expansion, that this scheme also provides results that are second order accurate [45, 46]. Furthermore, it should be noted, that the schemes of *Filippova* and *Mei* only use information from the nearest neighbors to model curved walls and can therefore be considered as local schemes.

Bounce Back Bouzidi

In contrast to the higher order wall treatments presented above, the wall boundary condition developed by *Bouzidi et al.* [5] is an interpolation supplemented scheme. The idea of this boundary condition is to exploit the stability and simplicity of the simple bounce back rule, while remaining time consistency, which results in the need for spatial extrapolation or interpolation of distributions, respectively. It is well known, that extrapolation of distributions renders the LBM vulnerable for instability. For that reason, a case distinction is needed in order to be able to interpolate, independent of the wall position. Two cases must be distinguished: The wall is closer to the fluid node ($q < 0.5$) or closer to the wall node ($q \geq 0.5$). These are the same cases as in the fictitious equilibrium schemes, but derived from a different point of view, as will be discussed now. In both cases, the result of the boundary treatment must be the values for the unknown distributions in wall opposite direction at the wall adjacent fluid nodes. Figure 3.9(a) shows the situation for the distribution pointing in direction normal to the wall, which is $q\Delta x < 0.5\Delta x$ away from the fluid node at $\mathbf{x}_{f1}$ and at time t.

An interpolation or extrapolation in time can be avoided by keeping the space-time integration consistent with all other nodes, that is, integrating from t to $t + \Delta t$. The integration can be split in two parts: Integrating along direction i over the interval

$s = \mathbf{e}_i \Delta t(1-q)$ and integration along the counter direction i^- over $\mathbf{e}_{i^-}\Delta t \cdot q$. Since the target end point of the integration is $\mathbf{x}_{f1}$ in this case, the start point is located at $\mathbf{x}_s$ as shown in Figure 3.9(a). Linear interpolation involving $\mathbf{x}_{f1}$ and $\mathbf{x}_{f2}$ is used to compute the distributions at $\mathbf{x}_s$ and at time t.

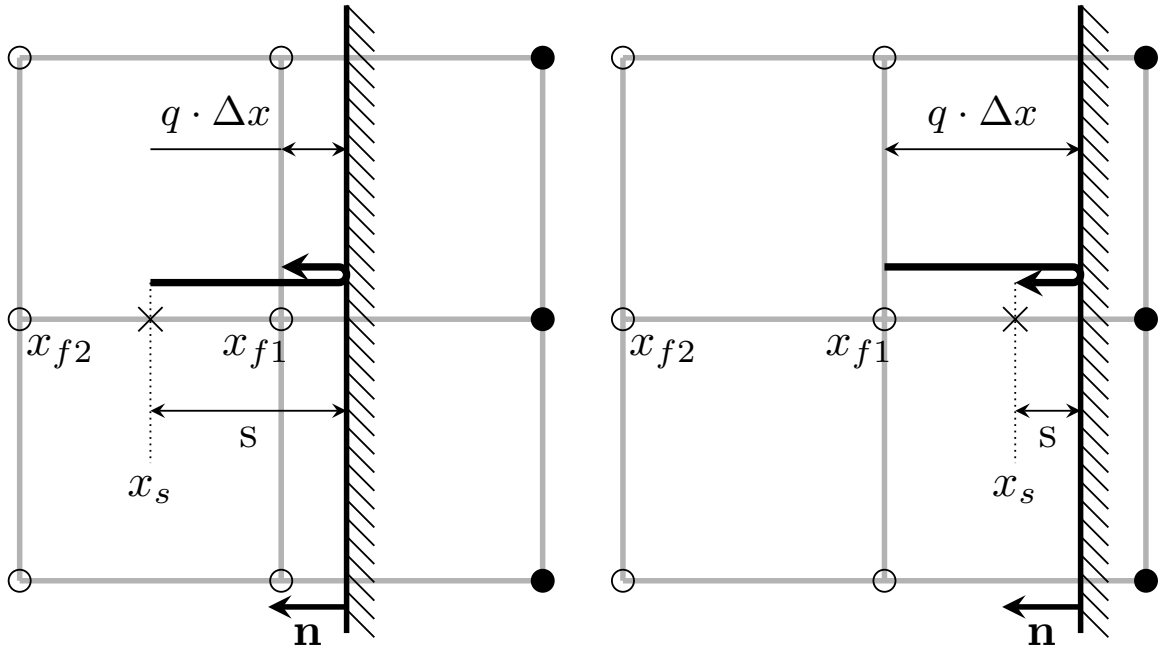
In the second case for $q \geq 0.5$, the spatial interpolation is done at time $t + \Delta t$. Start point of the space-time integration is $\mathbf{x}_{f1}$ at time t as it is depicted in Figure 3.9(b). The integration is done over $\mathbf{e}_i \Delta t \cdot q$ along direction i and over $s = \mathbf{e}_{i^-}\Delta t(1-q)$ along direction i^-, which results in values for the distributions in direction i^- at $\mathbf{x}_s$ and at time $t + \Delta t$. Missing distributions for the node at $\mathbf{x}_{f1}$ and for time $t + \Delta t$ can be computed through linear interpolation between the corresponding distributions of $\mathbf{x}_{f2}$ and $\mathbf{x}_s$ in direction i^-.

The *Bouzidi* scheme including the terms for moving walls is given by [5]

$$f_{i^-}(\mathbf{x}_{f1}, t+\Delta t) = \begin{cases} 2q_i f_i(\mathbf{x}_{f1}, t) + (1-2q_i) f_i(\mathbf{x}_{f2}, t) - 2w_i \rho(\mathbf{x}_{f1}, t) \dfrac{\mathbf{e}_i \mathbf{u}_w}{c_s^2} & q_i < 0.5 \\ \dfrac{1}{2q_i} f_i(\mathbf{x}_{f1}, t) + \dfrac{2q_i - 1}{2q_i} f_{i^-}(\mathbf{x}_{f2}, t+\Delta t) - \dfrac{1}{q_i} w_i \rho(\mathbf{x}_{f1}, t) \dfrac{\mathbf{e}_i \mathbf{u}_w}{c_s^2} & q_i \geq 0.5, \end{cases} \tag{3.9}$$

where it can easily be seen, that the wall boundary condition according to *Bouzidi* reduces to the simple bounce back rule for $q = 0.5$. Owing to the interpolation, this scheme must be considered non local, since next-to-neighbor information is needed.

Strictly speaking, since this scheme acts on the distributions through interpolation in space, it breaks the conservation of mass. The departure from mass conservation directly depends on the accuracy of the interpolation scheme employed. In practical simulations, this error source can be neglected, especially because the density variation scales like the square of the Mach number.



(a) $q < 0.5$ (b) $q \geq 0.5$

Figure 3.9: Case distinction for the Bouzidi bounce back scheme.

Inflow and Outflow Boundary Conditions

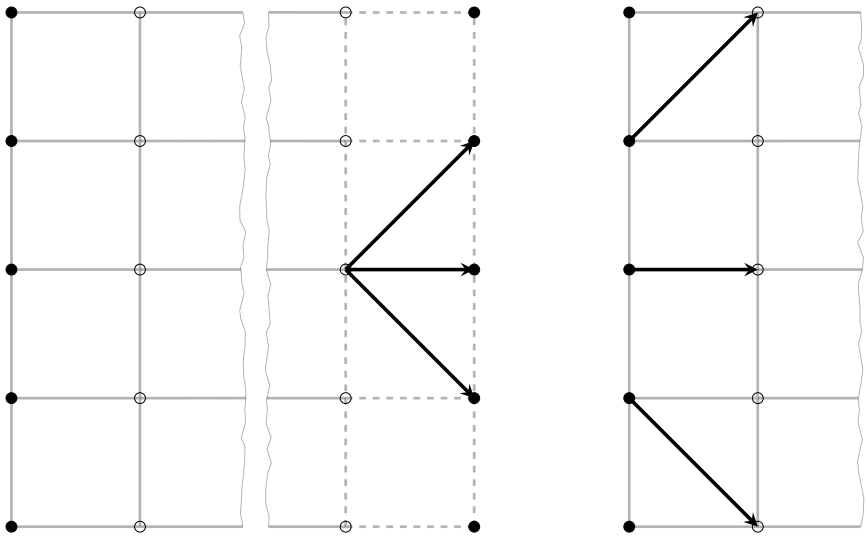
A straightforward way to model flows with open boundaries is the prescription of the equilibrium distributions calculated with the known macroscopic quantities: Velocity and density. In practice, velocity and density are rarely known simultaneously for a given boundary or it is desired to prescribe only one of the two quantities due to stability issues. In these cases, the unknown quantity is extrapolated from the fluid domain. Additionally, it is also possible to extrapolate the non-equilibrium part of the distribution from the fluid domain in order to enhance the quality of the boundary condition. This introduces a stress state on the boundary nodes as opposed to the plain equilibrium. In that sense, there is also the possibility to extrapolate the whole distribution from the fluid domain with the aim to model a zero gradient outflow boundary condition.

An other way of modeling an inflow boundary condition is to make use of the wall boundary conditions presented above. The idea is to drive the flow with a moving wall, where the direction and amplitude of movement is given by the prescribed velocity vector. This is a stable approach and has the advantage that the stress state, i.e., the non-equilibrium part of the distributions evolves naturally.

There are some more schemes that are suitable especially for transient outflow conditions, such as the convective boundary condition or the do-nothing boundary condition [62].

Periodic Boundary Condition

Often the computational effort of a simulation can be drastically reduced by exploiting the periodicity or symmetry of the computational domain. The flow leaving the domain at a periodic boundary re-enters it at its periodic counterpart. Therefore, this boundary condition assumes the domain to be infinitely extended and the fluxes are equal on each periodic boundary pair. This boundary condition can be realized in the LBM by an appropriate modification of the neighbor relationships for the transport step. Figure 3.10 shows the start at time t and the end at time $t + \Delta t$ of a transport step for an periodic boundary pair.



(a) Time t (b) Time $t + \Delta t$

Figure 3.10: Transport step for modeling a periodic boundary condition.

The distributions in Figure 3.10(a) at time t are supposed to leave the computational domain at the outlet. Since there are no more neighbors in the real domain, the domain is fictitiously extended. This is denoted by dashed lines. The neighbors in outlet direction are set to be the inlet nodes of the domain, denoted by black circles. That way, the distributions are recycled at the domain inlet as it is shown in Figure 3.10(b) at the end of the timestep at time $t + \Delta t$.

It should be noted, that periodic nodes have a full and valid set of distributions and, therefore, may need to conduct a collision step, depending on the implementation.

3.2.3 Force and Reference Frames

The question arises, how forces acting on surfaces can be computed within the Lattice Boltzmann Method. Since the force on a surface is an integral quantity, it can be computed, by integration of the surface stress. The stress may in turn be computed from the strain rate tensor using the generalized Hook's law as well as the pressure acting on the surface.

An elegant alternative to the integration of the surface stress was proposed by *Ladd and Verberg* [38], which is known as the momentum exchange method in the LBM community. The fact, that force equals the derivative of momentum with respect to time is used to compute surface forces. This method is used throughout the thesis for the evaluation of forces. It is implemented in *SamSolver* as the force acting on the wall evaluated at the wall nodes $\mathbf{x}_w$

$$\mathbf{f}^F = \frac{\Delta x^3}{\Delta t} \sum_{\mathbf{x}_w} \sum_i \mathbf{e}_i \cdot (f_i(\mathbf{x}_f, t) + f_i(\mathbf{x}_f, t + \Delta t) - 2w_i\rho), \tag{3.10}$$

where direction i denotes the incoming distributions from the fluid nods $\mathbf{x}_f$ along the available lattice links. In contrast to the original momentum exchange method, we extended the expression with an ambient pressure term $2w_i\rho$ built from the zero velocity equilibrium distribution. This is necessary for the correct evaluation of forces on bodies that are not completely immersed in the fluid, e.g., a wall mounted cube. Since there is no surface on the opposite of the lid surface, the momentum exchange method gives wrong results in such cases. For bodies completely immersed in the fluid, the ambient pressure term cancels, when summing over all wall nodes and therefore our scheme reduces to the original one. It should be noted, that the space and time steps have to be taken from the corresponding octree levels for simulations with grid refinement.

In this scheme, the time interval is simply chosen to be Δt, evaluated at the wall node and the adjacent fluid node. Thus, the momentum exchange method is a first order method, since the same argumentation for the simple bounce back scheme holds. However, the momentum exchange method gives reliable results even for curved boundaries [47].

The same approach can be used to compute moment forces. In addition to the force vector $\mathbf{f}^F$, one has to compute the radius $\mathbf{r}_w$, that is, the shortest distance between the wall node and the moment axis, e.g., the axis of rotation of an impeller. Given is a moment axis defined by the direction vector $\mathbf{d}$ and the origin point $\mathbf{a}$. The radius of a wall node to the axis is computed, by projection of

$$s = (\mathbf{x}_w - \mathbf{a}) \cdot \mathbf{d}, \tag{3.11}$$

onto the moment axis. It is then given by

$$\mathbf{r}_w = \mathbf{x}_w - (\mathbf{a} + s \cdot \mathbf{d}) \tag{3.12}$$

and the torque vector $\mathbf{m}^T$ is computed with the momentum exchange method according to

$$\mathbf{m}^T = \frac{\Delta x^3}{\Delta t} \sum_{\mathbf{x}_w} \mathbf{r}_w \times \sum_i \mathbf{e}_i \cdot (f_i(\mathbf{x}_f, t) + f_i(\mathbf{x}_f, t + \Delta t) - 2w_i\rho). \tag{3.13}$$

This approach is used throughout the thesis to compute moment forces and in particular the torque on the impeller in Section 6.2, where the spinning impeller is modeled with a wall moving with circumferential velocity. The velocity is computed with the same radius derived above. For the sake of completeness it should be mentioned, that a moving wall approach is equivalent to the frozen rotor approach in a system with relative velocity formulation. This is also implemented in *SamSolver*. If a moving reference frame approach is taken, one has to consider additional forces, such as the Coriolis force. For a discussion on moving reference frames and the corresponding forces, the reader is referred to [65].

CHAPTER 4

Solver Verification

This chapter is dedicated to the verification of *SAM-Lattice*. Before we start, the concept of verification and validation should be addressed in accordance to [2]. The concern of verification is to make sure, that the code is free of mistakes and accurately solves the implemented mathematical models. Therefore, the solutions of *SAM-Lattice* are compared to solutions where the mathematical description can be solved analytically. By contrast, the task of validation is to check how good a model matches real world processes for which the model was designed. In that sense, there is no validation without experimental data. Thus, the solutions of the computer code are compared with data from real world experiments and the accuracy is investigated with the auxiliary condition of experimental uncertainty. A sophisticated code development process relies on verification prior to validation. Hence, verification is done in this chapter, whereas validation will be conducted in Chapter 6 with experimental data of a propeller viscosimeter.

Verification is split into two parts. The Newtonian cases ensure the correctness of the main LBM implementation, whereas the generalized Newtonian case focuses more on the verification of the non-Newtonian model. The results of the Newtonian cases are presented here only qualitatively. The accuracy and convergence behavior is investigated in more detail in the generalized Newtonian case for both Newtonian and non-Newtonian fluids.

4.1 Newtonian

4.1.1 Stokes's First Problem: The Suddenly Accelerated Plate

Stokes's First Problem describes the flow induced by the acceleration of a flat plate from standstill. Non-dimensionalizing the problem results in a differential equation that can be solved analytically for the given boundary conditions. The solution gives the normalized velocity as a function of time and distance y perpendicular to the plate for a fluid with kinematic viscosity ν. It is self-similar and reads [55]

$$\frac{u}{U_0} = 1 - \mathrm{erf}(\Psi) = 1 - \mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right), \tag{4.1}$$

with the plate velocity U_0 at $t > 0$, the similarity parameter Ψ and the Gauß error function erf.

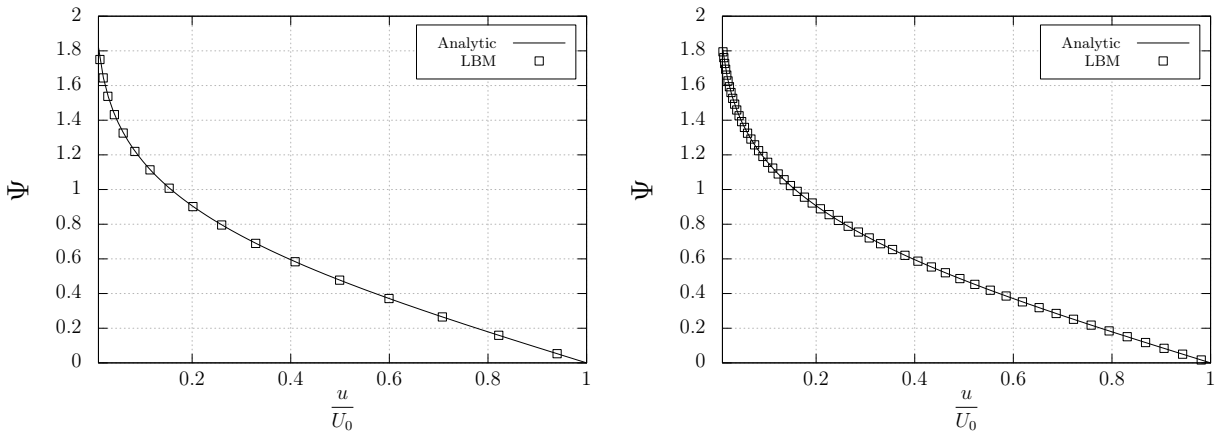
The simulation domain is modeled as a 3D rectangular duct with periodic boundaries in streamwise direction. All walls, except the moving plate, are modeled as frictionless walls. On the plate itself, the simple bounce back boundary condition is imposed. At $t = 0$, the plate and the whole fluid domain have zero velocity. The distributions are initialized with their equilibrium values. Table 4.1 summarizes the simulation setup.

Table 4.1: Simulation set up Stokes's First Problem

Parameter	Value	Unit
Δx	0.025	m
Δt	$6.944e-5$	s
Ω	1.2	–
ρ	1000	kg/m^3
ν	1	m^2/s
Char. L.	1	m
Ma	$4.811e-3$	–
Re	1	–
U_0	$0 \quad t = 0$ $1 \quad t > 0$	m/s

Collision model: SRT

Figure 4.1 shows the simulation results for two distinct points in time. With progressing time, more fluid layers with increasing distance to the moving plate get entrained. This



(a) 200 time steps

(b) 2000 time steps

Figure 4.1: LBM solution of Stokes's First Problem at two distinct moments in time

can be seen on the image to the right, where more grid points have a non-zero velocity. Yet, since the solution is self-similar, the solutions of any time are rescaled to the analytic solution through non-dimensional representation with Ψ. One can see from Figure 4.1 that the LBM simulation correctly reproduces the solution to Stokes's First Problem.

4.1.2 Stokes's Second Problem: The Oscillating Plate

Stokes's Second Problem has the exact same geometrical configuration as in Stokes's First Problem, but this time, the plate oscillates harmonically in plane instead of a spontaneous acceleration. Once more, the problem is solved through non-dimensionalization. The solution is given by [55]

$$\frac{u}{U_0} = exp\left(-\Psi\right) \cdot cos\left(\omega t - \Psi\right) = \exp\left(-y\sqrt{\frac{\omega}{2\nu}}\right) \cdot cos\left(\omega t - y\sqrt{\frac{\omega}{2\nu}}\right), \tag{4.2}$$

where ω is the oscillation frequency, U_0 the oscillation amplitude,the similarity parameter Ψ and y the distance perpendicular to the plate. The solution to Stokes's Second Problem describes the flow induced by the oscillating plate movement, which gets exponentially damped with increasing distance to the plate. The strength of the damping also depends on the viscosity of the fluid. Table 4.2 provides an overview of the simulation setup for this problem.

Figure 4.2 shows the dimensionless representation of the simulation results for four prominent plate positions. The dimensionless plate velocity, due to the harmonic oscillation, is denoted on the abscissa and the results for π and $\pi/2$ are highlighted. LBM results are taken after the first oscillation was complete. It is obvious, that the simulation results fit

Table 4.2: Simulation setup for Stokes's Second Problem

Parameter	Value	Unit
Δx	0.025	m
Δt	$1.388e-3$	s
Ω	1.2	–
ρ	1000	kg/m^3
ν	0.05	m^2/s
Char. L.	1	m
Ma	0.096	–
Re	20	–
ω	$\frac{\pi}{5000 \cdot \Delta t}$	1/s
U_0	$0 \quad t = 0$ $U_0 \cdot cos(\omega t) \quad t > 0$	m/s

Collision model: SRT

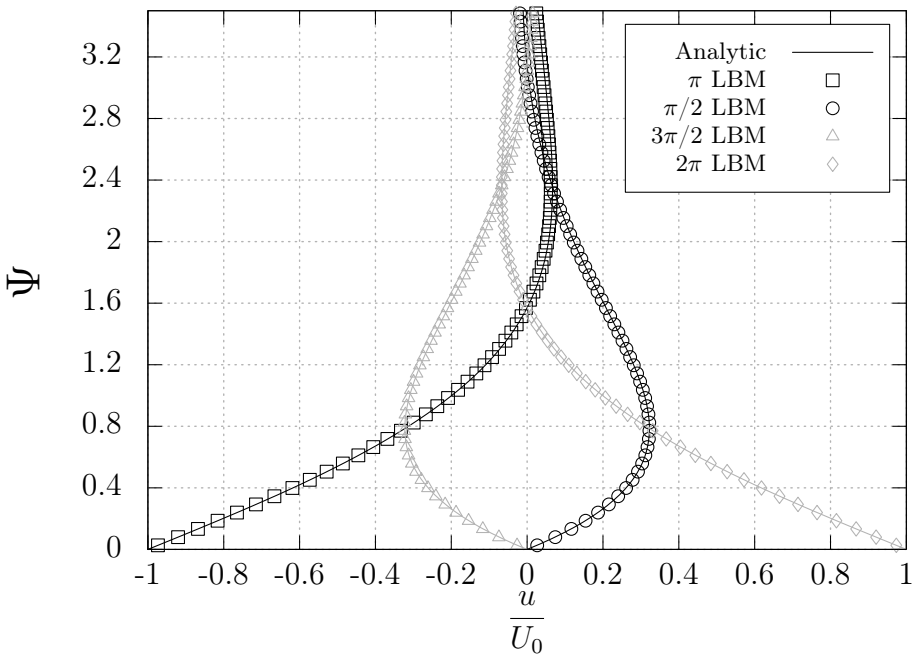


Figure 4.2: Solution to Stokes's Second Problem for four distinct plate positions

the analytic solution very well. An other interesting quantity for this problem is the wall shear stress on the plate. For a Newtonian fluid, the wall shear stress possesses an angular phase shift to the plate frequency [49],

$$\tau_w(t) = cos\left(\omega t + \frac{5\pi}{4}\right). \tag{4.3}$$

This angular phase shift can be verified from results of the first fluid lattice node above the wall. The wall shear stress τ_w is evaluated locally through computation of the corresponding entry in the stress tensor according to eq. (2.82). A time history of normalized $\tau_w(t)$ is illustrated in Figure 4.3 together with the analytic solution from eq. (4.3) as well as the comparative frequency of the plate.

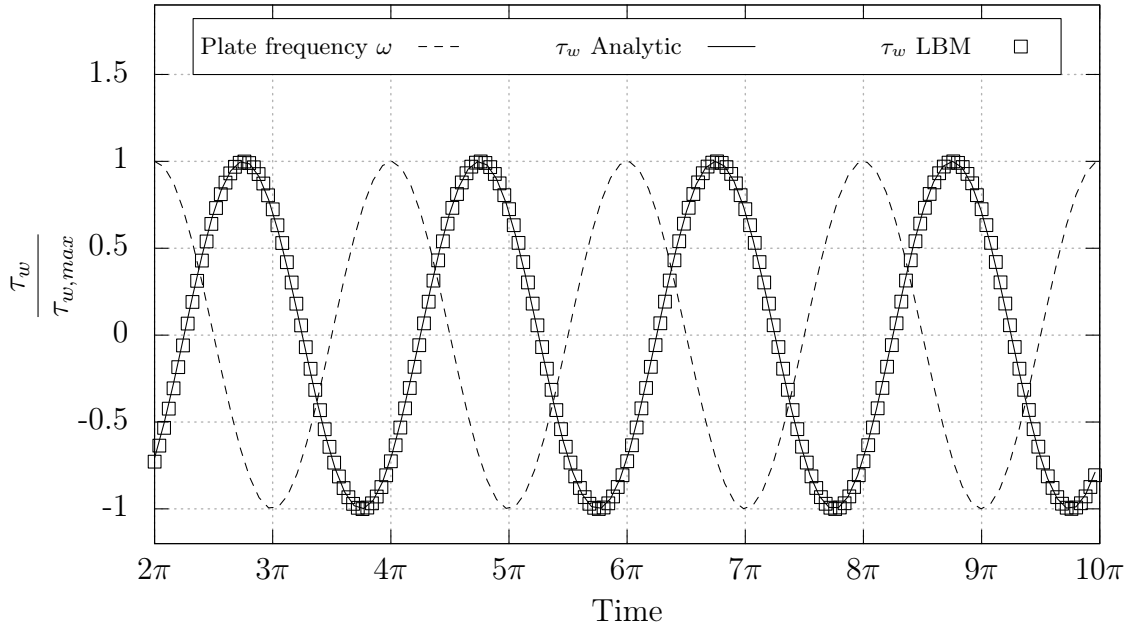


Figure 4.3: Angular phase shift of the wall shear stress to the plate frequency

It should be noted that only every hundredth result in time is depicted due to better readability. The results presented in Figure 4.2 and Figure 4.3 demonstrate that our implementation of the LBM correctly reproduces the solution to Stokes's Second Problem.

4.2 Generalized Newtonian

This section deals with the verification of the generalized Newtonian power-law model as it was discussed in Section 2.5.3. The simulation domains are modeled with periodic boundaries in streamwise direction and the pressure gradient is substituted for a spatially constant body force according to eq. (2.117). To quantify the quality of the simulation, we measure its accuracy with respect to the analytic solution over all lattice sites N,

$$E = \sqrt{\frac{1}{N} \cdot \sum_{\mathbf{x}} (u(\mathbf{x}) - u_a(\mathbf{x}))^2}. \tag{4.4}$$

This is the general definition of the global absolute error for the velocity distribution used throughout this thesis. Since the investigation in this chapter deals with steady state problems, the simulations are run until the following convergence criterion is reached:

$$|E(t) - E(t - \Delta t)| < 10^{-12}. \tag{4.5}$$

4.2.1 Hagen-Poiseuille Channel Flow

Since the correctness of the core implementation was already demonstrated in the previous Section 4.1, we will have a closer look at the spatial convergence behavior of the method. In order to investigate this with minimal influence of the wall boundary conditions, the flow through two parallel plates is considered. The simulation domain is modeled as a square channel, where one pair of opposite walls are treated as frictionless walls and on the other pair, the simple bounce back wall boundary condition is imposed. Therefore, the plane Hagen-Poiseuille equation can be used. Usually, the equation is given for a Newtonian fluid, but can be extended to generalized Newtonian power-law fluids. The velocity distribution for this case is given by [4]

$$u_a(y) = \frac{n}{n+1} \cdot (\frac{dp}{dx}\frac{1}{k})^{\frac{1}{n}} \cdot (h^{\frac{n+1}{n}} - y^{\frac{n+1}{n}}), \tag{4.6}$$

where $y \in [-h, h]$ is perpendicular to the flow direction. Again, the pressure gradient is substituted for the body force $dp/dx = \rho \cdot a$. It should be noted, that the lattice is generated as to exactly retain the channel width and height. The height of the channel H is set to one and is measured from the center line of the channel. Moreover, the channel is axis aligned. Therefore, the simple bounce back scheme becomes a second order wall boundary condition. The simulation setup is summarized in Table 4.3 for the coarsest lattice investigated. All simulations are initialized with zero velocity equilibrium values.

Table 4.3: Simulation setup for the generalized Hagen-Poiseuille Channel flow

Parameter	Value	Unit
$H/\Delta x$	10	–
ρ	1000	$\mathrm{kg/m^3}$
a	1	$\mathrm{m/s^2}$
k	250	$\mathrm{Pa\,s^n}$
n	$0.6 - 1.4$	–
Char. L.	1	m
U	1	m/s
Ma	0.08	–

Collision model: SRT

Newtonian Case ($n = 1.0$)

First, the Newtonian case with $n = 1.0$ is considered. With the goal of verification, the velocity profile is examined. It is taken from the lattice values in the vicinity of the channel center, along a line with coordinate y perpendicular to the channel walls. Evaluated is the streamwise component of the velocity result, although the remaining components are negligibly small. Figure 4.4 shows the velocity profile of the Newtonian channel flow with resolution $H/\Delta x = 20$. Additionally, an illustration of the lattice is attached. The LBM solution matches the parabolic form of the analytic solution of eq. (4.6). Vertical lines are added to help with the assignment of the results. Again, one can see the lattice is constructed, such that the channel walls, illustrated with a thick line, lie midway between the first fluid nodes and the wall nodes.

At this point, the grid refinement algorithm, introduced in Section 2.5.2, should be verified. To this end, a level 2 lattice is used, which is two times refined towards the channel walls. The result is depicted in Figure 4.5, with an appropriate illustration of the level 2 lattice. Although it is not easily visible, the channel walls, drawn with a thick line, lies midway between the fluid and wall nodes of the finest grid level. Due to the setting for the lattice smoothing, the intermediate level 1 grid layers occupy a relatively large portion of the domain. Once more, the LBM solution matches the analytic solution very well. Besides that, the velocity field is smooth throughout grid transitions.

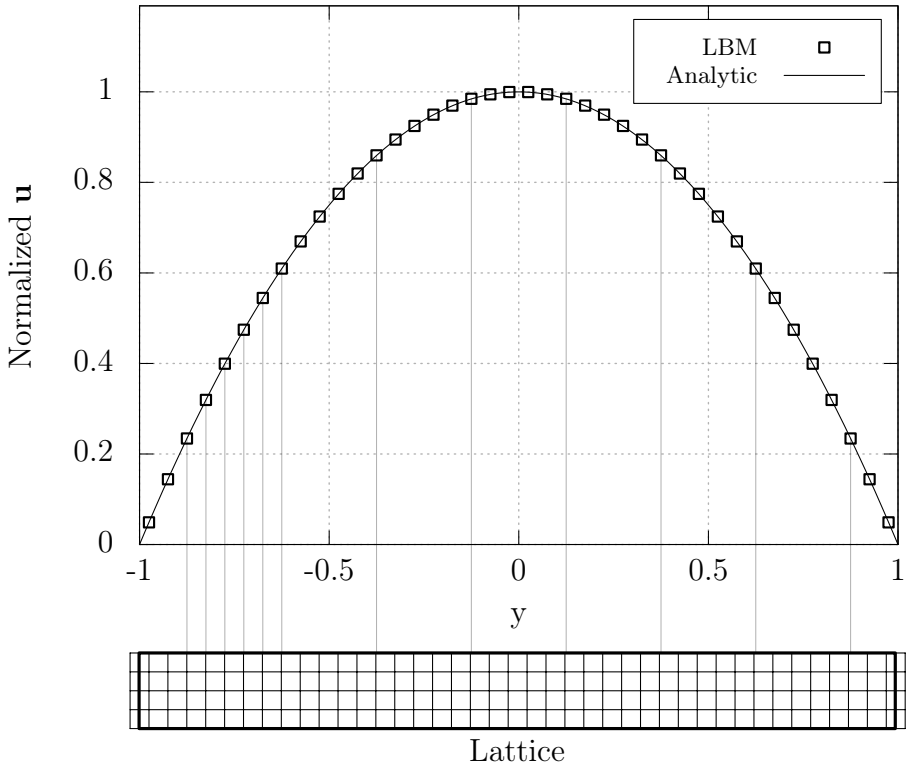


Figure 4.4: Velocity profile for Newtonian channel flow.

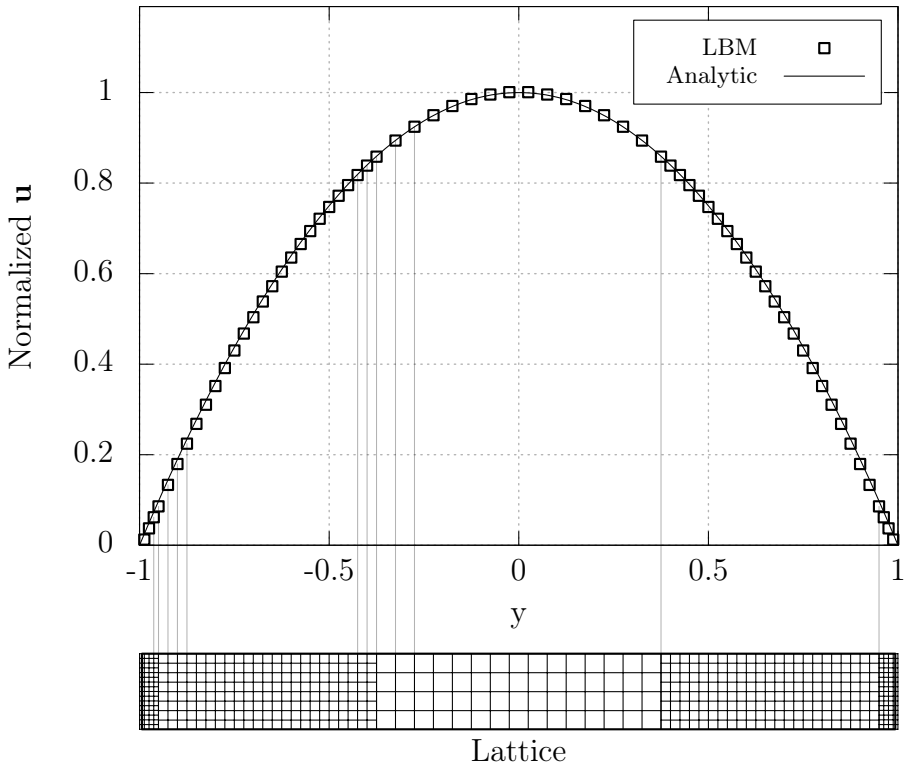


Figure 4.5: Velocity profile for Newtonian channel flow on level 2 lattice.

In order to verify the grid transition constraints eq. (2.103), that is, continuity up to second order moments, the strain rate in streamwise direction is inspected. The analytic solution for the strain rate is obtained by taking the derivative of eq. (4.6) with respect to y. Figure 4.6 illustrates the results. The strain rate profile fits excellently to the analytic solution. Furthermore, the strain rate is also smooth at the lattice transitions.

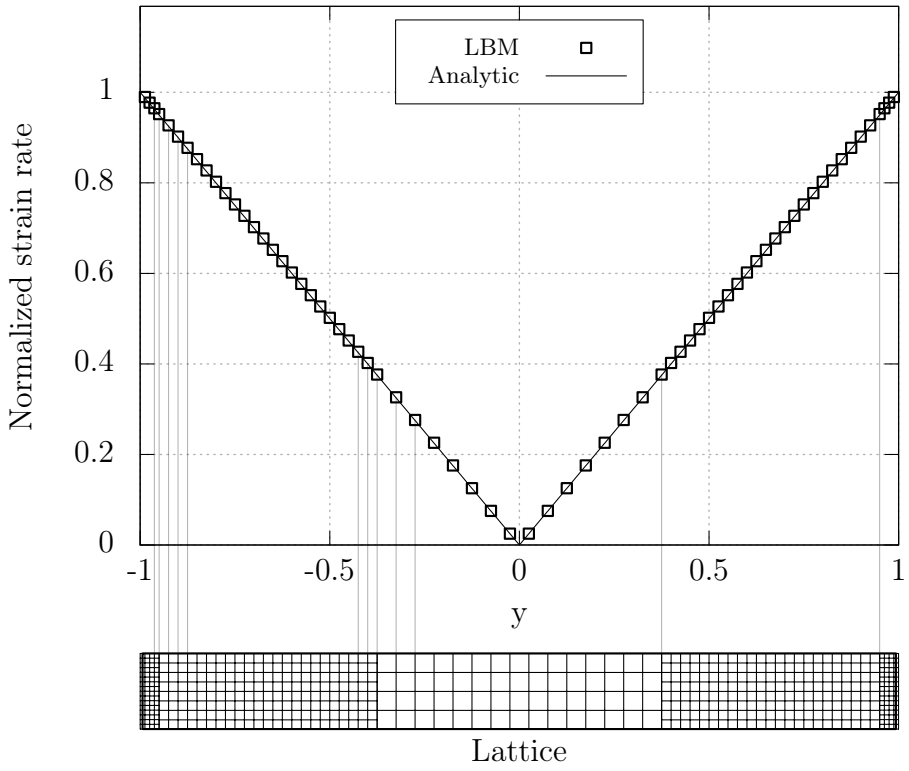


Figure 4.6: Strain rate profile for Newtonian channel flow on level 2 lattice.

Non-Newtonian Case ($n \neq 1.0$)

Next, the non-Newtonian model is verified for different shear thickening and shear thinning fluids. Like in the Newtonian case, we examine the velocity profiles. For the sake of clearness, the results are split into two parts, namely for fluids with $n = 0.75$ and $n = 1.25$, and $n = 0.6$ and $n = 1.4$. Figure 4.7 presents simulation results for the first group of fluids.

As already mentioned, the pressure gradient is constant. Hence, the various non-Newtonian fluids have different maximal values of velocity in the apex of the profile. For that reason, the velocity profiles are normalized. One can see, that the profile of the shear

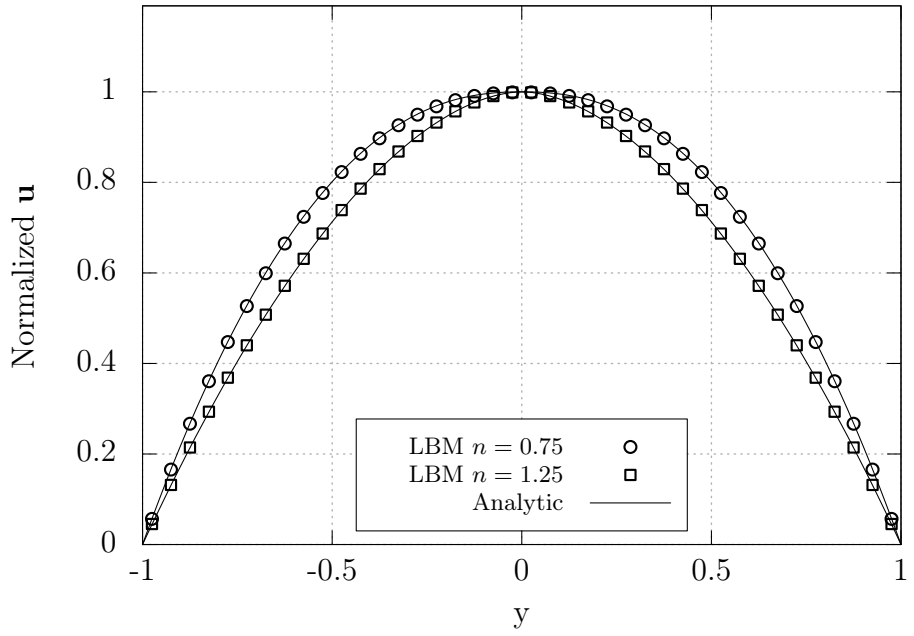


Figure 4.7: Velocity profile for shear thinning ($n = 0.75$) and shear thickening ($n = 1.25$) fluids.

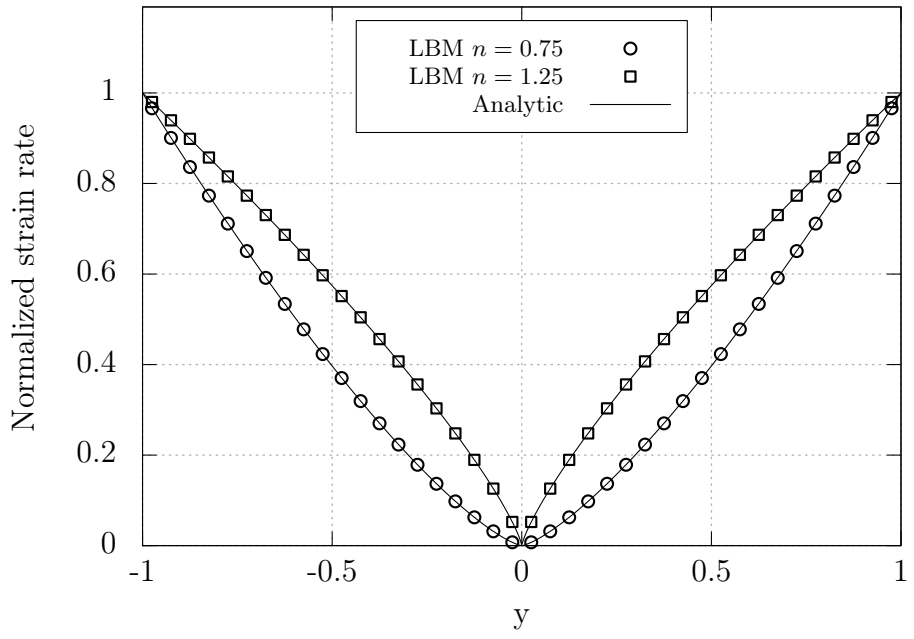


Figure 4.8: Strain rate profile for shear thinning ($n = 0.75$) and shear thickening ($n = 1.25$) fluids.

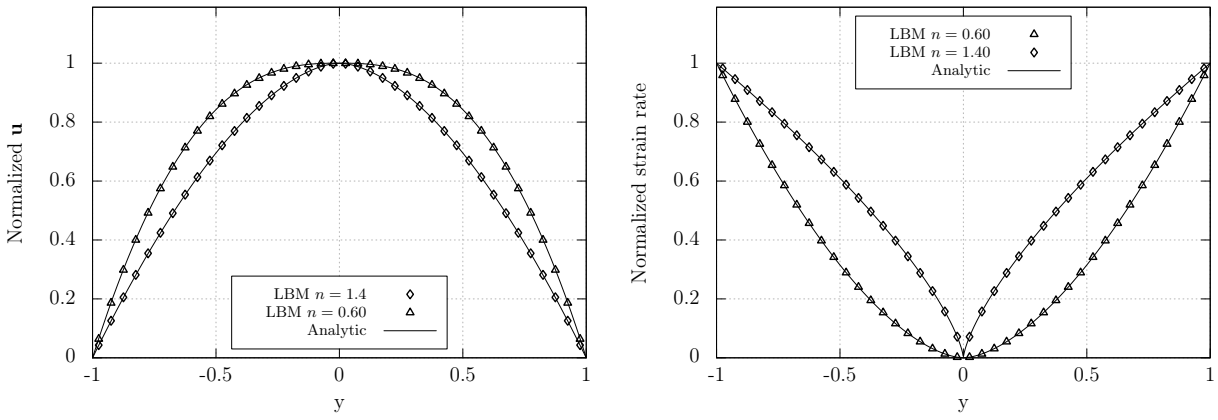
thinning fluid is more flat around the center line, albeit the fluid exhibits higher values of maximum velocity compared to the shear thickening fluid, when the same pressure gradient is present.

Furthermore, since non-Newtonian fluid modeling is based on the strain rate tensor norm, the strain rate in streamwise direction is also examined. This is depicted for the first group of fluids in Figure 4.8.

The LBM simulation reproduces the analytic profiles very well. The strain rates are lower for shear thinning fluids under the influence of the same pressure gradient. Also, the profile gets more parabolic and flat towards the channel center. By contrast, the profile of the shear thickening fluid gets more steep on the channel center.

It should be noted, that due to the way of lattice construction and the ratio $H/\Delta x$, there is no lattice node at the exact center line of the channel. Hence, there is no vanishing strain rate tensor norm present. Therefore, the present simulations could actually be conducted without the need to resort to stability criteria, since the minimal exhibited strain rate still exceeds the stability threshold.

The results for the second group of fluids is summarized in Figure 4.9(a) and Figure 4.9(b).



(a) Velocity profile

(b) Strain rate profile

Figure 4.9: Simulation results for $n = 0.6$ and $n = 1.4$.

One can state, that the already mentioned preferences and differences between the shear thickening and shear thinning fluids are much more pronounced with increasing and decreasing flow behavior index, respectively. Once more, the LBM simulation results are qualitatively verified.

Accuracy

Finally, the accuracy and convergence behavior is considered quantitatively. Four successively refined lattices will be investigated to judge the spatial convergence of the method. The simulations are initialized with zero velocity equilibrium values and run until the convergence criterion eq. (4.5) is fulfilled. Calculation of the simulation Mach number is done with a characteristic flow speed of $U = 1$ m/s and the pressure gradient is calculated with an acceleration of $a = 1$ m/s^2 for all fluids. Therefore, simulations are done for a constant Reynolds number with respect to $n = 1.0$. Diffusive scaling, as introduced in Section 2.4.2, is employed for the convergence study. The result is depicted in Figure 4.10. A comparative line with slope -2 indicates second order convergence of the method. As one can see, the absolute values of the error is higher for fluids with smaller flow behavior index.

Despite different error definitions, as well as Reynolds numbers, the results obtained here are in accordance with the findings in [6]. A more detailed analysis of the accuracy of non-Newtonian fluids is conducted in Section 7.4.

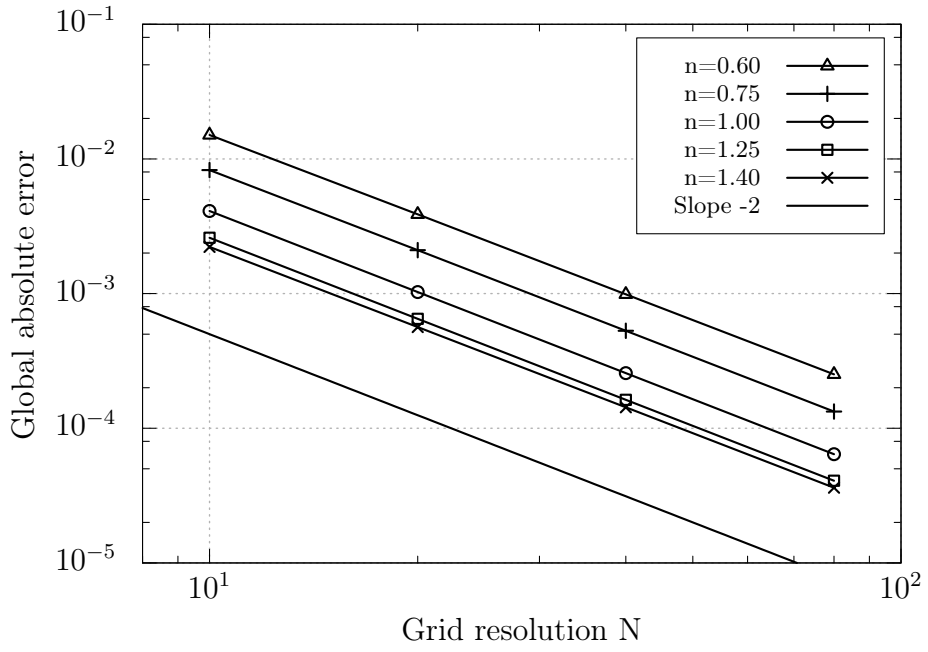


Figure 4.10: Spatial convergence for different shear thickening, shear thinning and Newtonian fluids on four successively refined lattices.

4.2.2 Hagen-Poiseuille Pipe Flow

The Hagen-Poiseuille equation gives the pressure drop in a fluid, flowing with a given mass flow rate through a cylindrical pipe with radius R. Because of the no-slip condition at the pipe walls, the velocity profile of a pipe flow takes paraboloidal form. In this case, the generalized analytic velocity distribution is given in radial coordinates [4]

$$u_a(r) = \frac{n}{n+1} \cdot \left(\frac{dp}{dx} \frac{1}{2k} \right)^{\frac{1}{n}} \cdot \left(R^{\frac{n+1}{n}} - r^{\frac{n+1}{n}} \right), \tag{4.7}$$

where $r \in [-R, R]$, the flow consistency index k and the flow behavior index n, respectively.

Newtonian Case ($n = 1.0$)

For the purpose of verification, we will first investigate a Newtonian fluid for $n = 1.0$. The spatial convergence was already demonstrated in the previous subsection. Hence, we will have a closer look at the implementation of the wall boundary conditions. Since the walls possess curvature, it is necessary to make use of higher order bounce back schemes for which the proposed $\mathcal{O}(\Delta x^2)$ behavior of E_{BC} can be inspected. In this context, the influence of the error E_g due to the geometry discretization, as introduced in Equation (2.92), is highlighted.

In order to ensure spatial convergence, the diffusive scaling is employed. This effectively means that the relaxation parameter Ω is constant throughout the different resolved lattices. Table 4.4 shows the simulation set up for the Newtonian pipe flow with the coarsest lattice resolution used. The pipe geometry is discretized with a medium resolution for the additional investigation of E_g. All second order wall boundary conditions introduced in Section 3.2.2 are tested.

Table 4.4: Simulation set up for the Newtonian pipe flow case $n = 1.0$

Parameter	Value	Unit
Δx	0.2	m
Δg	0.05	m
Δt	$1.777e-2$	s
Ω	1.2	–
ρ	1000	kg/m^3
k	250	Pa s
R	1	m
Ma	0.15396	–

Collision model: SRT

The simulation results for four successively refined lattices are shown in Figure 4.11. One can see that the improved bounce back schemes of *Bouzidi* and *Mei* perform slightly better than the original bounce back scheme for curved geometries by *Filippova*. But independent of the scheme used, the deficient geometry discretization results in a $\mathcal{O}(1)$ behavior of the spatial convergence. Therefore, the same simulation set up is rerun with a

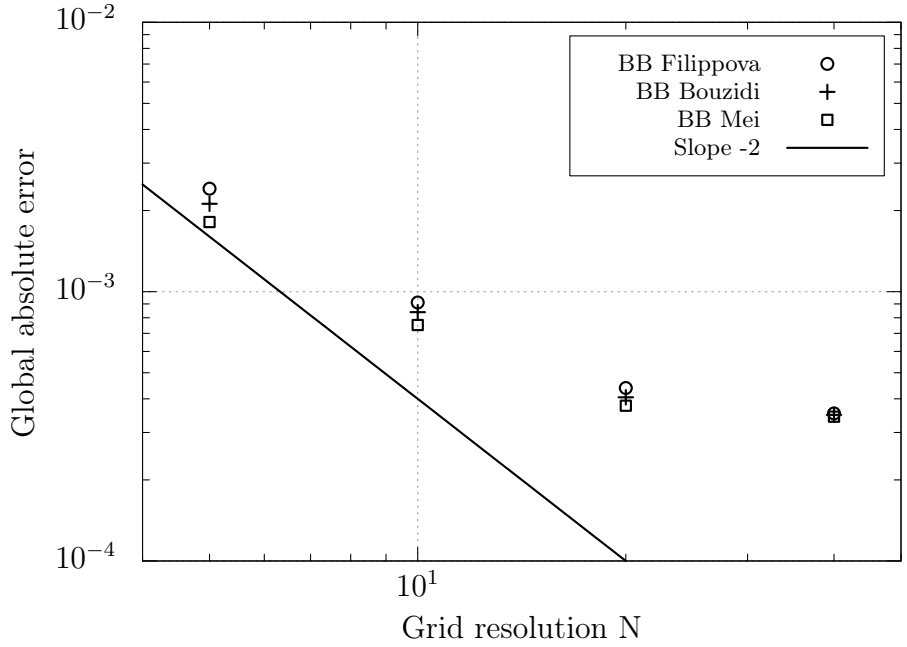


Figure 4.11: Spatial convergence with medium Δg

sufficiently fine geometry discretization. In order to consistently reduce the error E_g, the geometry discretization is refined, such that $\Delta g \sim \Delta x$ and $\Delta g = \Theta(\Delta x)$. This premise is confirmed from the simulation results in Figure 4.12. One can state that the $\mathcal{O}(\Delta x^2)$ spatial convergence, as denoted by a solid line, is confirmed for all wall boundary conditions investigated. Thus, the results indicate that the error due to the geometry discretization can be approximated as $E_g = \mathcal{O}(\Delta x^2)$ in eq. (2.92) with an appropriate choice of Δg.

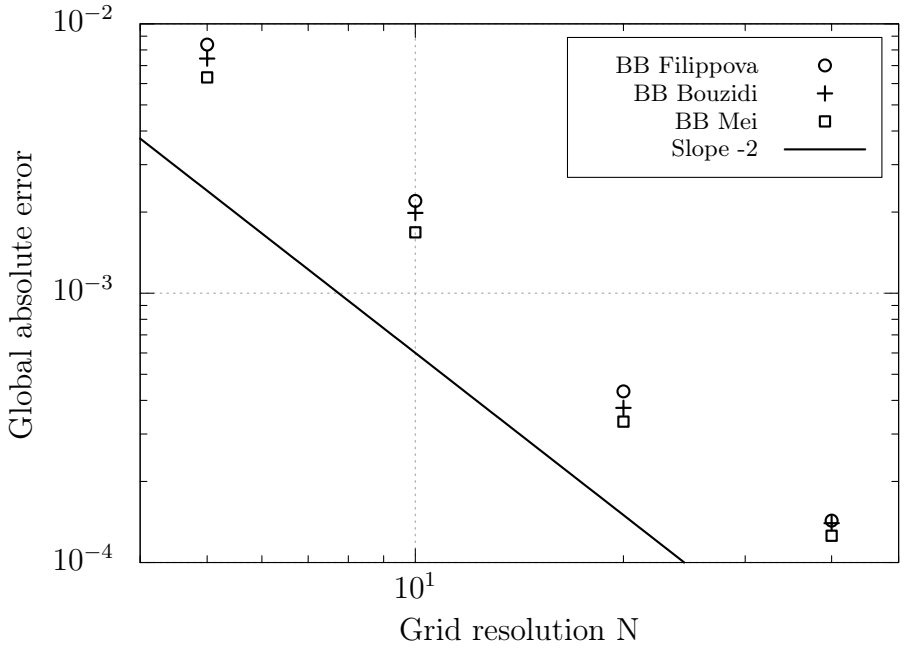
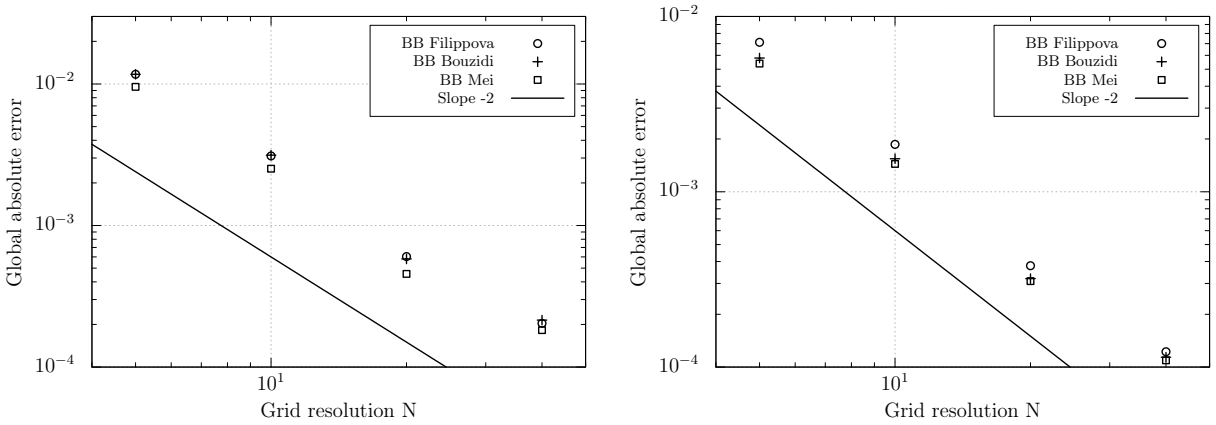


Figure 4.12: Spatial convergence of Newtonian pipe flow with suitably fine Δg

Non-Newtonian Case ($n \neq 1.0$)

Since the dimensionless collision frequency Ω appears in the bounce back formulas for curved geometries it is of particular interest how they perform in the non-Newtonian case. Therefore, the spatial convergence study is rerun for two non-Newtonian fluids with $n = 0.75$ and $n = 1.25$, respectively.

Figure 4.13 shows the developing of the absolute global error when the lattice resolution is increased and the time step is set according to the diffusive scaling. All results seem to exhibit second order spatial convergence behavior as indicated with a solid line. Like in the Newtonian case one can state that the bounce back scheme of *Mei* performs slightly better than that of *Bouzidi* which in turn shows slightly better results than the original scheme of *Filippova* for both non-Newtonian fluids in terms of absolute global error. But the results indicate that this ranking is inverted in terms of the convergence rate. Table 4.5 summarizes the actual convergence rates computed from the log-log values of the results of the convergence study for both Newtonian and non-Newtonian cases.



(a) $n = 0.75$ (b) $n = 1.25$

Figure 4.13: Spatial convergence of non-Newtonian fluids for different second order wall boundary conditions

Table 4.5: Computed convergence rates of the wall boundary conditions for curved geometries

Fluid	Convergence rate		
	BB *Filippova*	BB *Bouzidi*	BB *Mei*
$n = 0.75$	1.95903	1.94807	1.96997
$n = 1.00$	1.96866	1.95081	1.94994
$n = 1.25$	1.96926	1.94264	1.93273

CHAPTER 5

Viscosity Adaption Method

The Lattice Boltzmann Method can be extended to non-Newtonian fluid flows as was shown in Section 2.5.3. Several authors have demonstrated its applicability, e.g., in [6]. The regular grid based scheme makes the LBM ideally suitable for simulations involving complex solids. Such geometries are common in the food processing industry, where fluids are mixed by static mixers or agitators. Those fluid flows are often laminar and non-Newtonian. Special care has to be taken for non-Newtonian Lattice Boltzmann simulations in order to keep them stable. A straightforward way is to truncate the modeled viscosity range by numerical stability criteria eq. (2.112). This is an effective approach, but from the physical point of view the viscosity bounds are chosen arbitrarily. Moreover, these bounds depend on and vary with the grid and time step size and, therefore, with the simulation Mach number. However, the simulation Mach number is problem dependent and must be set by the user *a priori*. Thus, the modeled viscosity range may not fit to the actual physical problem, which yields corrupted simulation results. A way around is to determine a suitable simulation Mach number by longsome precursor simulations [9, 48]. This renders the standard LBM unattractive for practical use, especially for complex cases. Moreover, in the standard LBM the simulation Mach number is constant throughout the simulation and, thus, the LBM cannot account for changing transient viscosity ranges.

In this chapter, we derive the viscosity adaption method (VAM) to consistently and constantly adapt the modeled viscosity range to the actual physical problem. This is the basis of the viscosity adaptive Lattice Boltzmann Method (VALBM). We verify the VALBM in Chapter 6 for steady state channel flow and transient pulsatile pipe flow of non-Newtonian fluids. Additionally, validation is done with experimental data from a propeller viscosimeter.

5.1 Theoretical Considerations

The basis of the following discussion is the modeling of a shear thinning fluid, flowing between two parallel plates, with the Lattice Boltzmann Method. In Figure 5.1, the developing of the kinematic viscosity of the shear thinning fluid is plotted against the shear rate. The dashed area denotes the continuous physically exhibited shear rate range and viscosity range, respectively. A suitable Mach number for the simulation of the problem is unknown *a priori*, since the shear rate range is a solution-dependent quantity. This lies at the heart of the VALBM. The modeled range for a standard LBM simulation with an arbitrarily chosen simulation Mach number is also depicted in Figure 5.1. The range indicates what can be modeled within the stability bounds for this choice of the Mach number. It is obvious, that a vast portion of the modeling range is wasted and the Ω_{min} limit truncates the viscosity range with a relatively low value for ν_{max}.

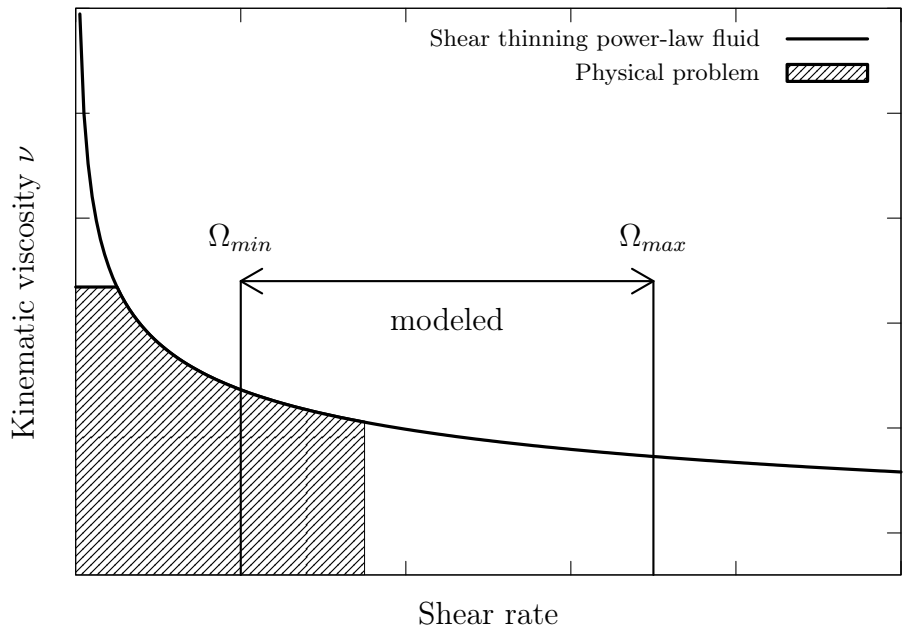


Figure 5.1: Physical problem and numerical approximation

The objective of the VAM could be to shift one of the boundaries of Ω to coincide with one of the boundary values of the physical problem. In the example above, the modeling is significantly improved by the shift of the Ω_{max} limit towards the maximum shear rate present in the system. For this purpose a rescaled time step size Δt^* is introduced, which leads to rescaled lattice values:

$$e^* = \frac{\Delta x}{\Delta t^*} \qquad c_s^* = \frac{e^*}{\sqrt{3}} \qquad Ma^* = \frac{U}{c_s^*} \qquad \Omega^* = \frac{{c_s^*}^2 \Delta t^*}{\nu^* + 0.5 \cdot {c_s^*}^2 \Delta t^*}, \tag{5.1}$$

where Ω^* must be rescaled at each lattice site. In order to yield the same macroscopic moments in the new system, the distributions f_i must be rescaled. This is done with the linear transformation matrices M in the old system and M^* in the new system according to

$$f_i^* = M^{*-1} M f_i \,. \tag{5.2}$$

By definition, the moments and especially the second order moment tensor $\Pi_{\alpha\beta}$ is unchanged by this operation. However, the macroscopic shear rate eq. (2.110) changes because of the pre-factor. Consequently, the kinematic viscosity changes through the constitutive law used in the simulation. It is desired that the rescale operation (5.2) leaves the shear rate and, therefore, the viscosity unaffected. We achieve this through rescaling of ${f_i^*}^{neq}$ in eq. (2.110) to give the same shear rate, hence $\nu^* = \nu$,

$$f_i^{**} = {f_i^*}^{eq} + \frac{\nu + \frac{\Delta x^2}{6 \cdot \Delta t^*}}{\nu + \frac{\Delta x^2}{6 \cdot \Delta t}} \cdot (f_i^* - {f_i^*}^{eq}), \tag{5.3}$$

where ${f_i^*}^{eq}$ is computed from eq. (2.27) in the new system.

5.2 Choice of Time Step Size

In order to adapt to the physical problem, the time step size Δt^* is determined from the minimal or maximal shear rate in the modeled system, which is a solution-dependent quantity. Following the example situation in Figure 5.1, the numerical approximation of the physical problem is improved if we shift the maximal collision frequency towards the maximal shear rate of the system. Therefore, the target collision frequency Ω_t should take the value of Ω_{max}, built with the kinematic viscosity ν_m obtained from the maximum shear rate. Thus, we choose the new time step size Δt^* according to

$$\Delta t^* = \frac{(\frac{1}{\Omega_t} - \frac{1}{2}) \cdot \Delta x^2}{3 \cdot \nu_m}. \tag{5.4}$$

Since the value of ν_m is solution-dependent, it can be expected to converge to the physical value of the system if the time step size is recursively adapted. From eq. (5.1) it follows that an increase in Ω causes a decreased time step size and, therefore, a decrease in Mach because $Ma \propto \frac{\Delta t}{\Delta x}$. Hence, we can expect a prolonged simulation time with an enhanced time resolution. Another possibility is to set the target relaxation parameter by Ω_{min}. This is possible in cases where the lowest shear rate still exceeds a valid threshold like in the non-Newtonian Hagen-Poiseuille channel flow (see Figure 4.8). In this discrete physical problem, we can opt for the Ω_{min} bound without triggering the upper Mach number limitation. If Δt^* is determined by eq. (5.4) with $\Omega_t = \Omega_{min}$ and ν_m from the minimal shear rate, the time step size as well as the Mach number is increased. Therefore,

the simulation is accelerated. Note, that in a general flow situation it may then be possible to cut off viscosities from high shear rates due to the Ω_{max} limit with this approach. The optimal choice for Ω_{min} will be part of the investigations in Chapter 7.

5.3 Mach Number Limitation

As described in Section 5.2, the viscosity adaptive LBM makes use of a variable time step size on a fixed grid. Given that the Mach number is coupled with the time step size, we have to make sure the lattice values remain valid throughout the simulation. To this end, we limit the possible choice of the time step size through physically meaningful Mach number limits.

$$Ma^{**} = \begin{cases} Ma^{max} & Ma^{*} > Ma^{max} \\ Ma + \lambda \cdot (Ma^{*} - Ma) & \lambda \in [0,1] \\ Ma^{min} & Ma^{*} < Ma^{min} \end{cases} \tag{5.5}$$

It is common practice to consider flows with Mach numbers Ma^{max} below 0.3 to be weakly compressible [18]. Additionally, we set $Ma^{min} = 10^{-4}$ to stay in the weakly compressible regime because the hydrodynamic limit does not exist for uncorrected BGK dynamics [12, 40]. In practice, this also prevents the time step size to get uneconomically small. For enhanced stability properties of the viscosity adaptive LBM, we compute the simulation Mach number for the next time step Ma^{**}, through under-relaxation of the Mach number Ma^{*} obtained from eq. (5.1) and eq. (5.4). The final lattice values including the time step size have to be recomputed from Ma^{**}. For stationary flows, the simulation Mach number can be expected to converge as the target shear rate converges to the physical problem. Hence, we are able to introduce a convergence criterion for the simulation Mach number,

$$\frac{|Ma^{**} - Ma|}{Ma} < \delta. \tag{5.6}$$

If the expected Mach number change is small, the adaption process can be abandoned. In order to keep the Mach number threshold δ independent of the actual value of the Mach numbers and, therefore, meaningful for all simulations, the relative Mach number change is considered.

5.4 Multi-Level Approach

So far, the presented considerations are valid for single level simulations. The question arises, how the VAM can be extended to simulations with local grid refinement. In order to ensure continuity of the Re number, a redefinition of the stability criteria eq. (2.112) and eq. (2.111) is necessary, since the relationship $\Omega^{c} > \Omega^{f}$ holds at the interface. As

the maximal value for Ω_{max} must not be exceeded in the whole simulation domain, the minimal value of the viscosity used for the dimensionless collision frequency in each level is determined by $\nu_{min} = \nu^c_{min}$ computed from Ω_{max} on the coarsest level. On the other hand, Ω_{min} must not be exceeded globally. Hence, the global maximal value for the viscosity $\nu_{max} = \nu^f_{max}$ is computed from Ω_{min} on the finest level. The stability criteria for the coarse and fine level can, thus, be expressed as:

$$\Omega^c = \begin{cases} \Omega_{max} = \dfrac{c_s^2 \Delta t^c}{\nu^c_{min} + 0.5 c_s^2 \Delta t^c} \\ \Omega \\ \Omega^c_{min} = \dfrac{c_s^2 \Delta t^c}{\nu^f_{max} + 0.5 c_s^2 \Delta t^c} \end{cases} \tag{5.7}$$

$$\Omega^f = \begin{cases} \Omega^f_{max} = \dfrac{c_s^2 \Delta t^f}{\nu^c_{min} + 0.5 c_s^2 \Delta t^f} \\ \Omega \\ \Omega_{min} = \dfrac{c_s^2 \Delta t^f}{\nu^f_{max} + 0.5 c_s^2 \Delta t^f} \end{cases} . \tag{5.8}$$

It should be noted, that this multi-level approach narrows the overall modeled viscosity range as opposed to what would be possible with the increased resolution, since it is the intersecting set of the viscosity ranges of the coarsest and finest level. This is the price to pay for stability of simulations with grid refinement. Thus, another possibility, which presumes some knowledge about the flow of interest, is to refine the grid such that the values of Ω^c and Ω^f of the interface region are far enough from the stability borders. That way, the single level approach can be used for each grid level separately. An investigation and discussion on the different approaches in a simulation with local grid refinement is done in Section 6.2.

5.5 Algorithm Overview

Finally, we will summarize the proposed algorithm. To precis the procedure, we consider a stationary flow of a shear thinning fluid (Figure 5.1) with the aim to accelerate the simulation. Starting the algorithm for the viscosity adaption method at an arbitrary time step t_0 with an arbitrary simulation Mach number it can be summarized as follows:

- Save the minimal shear rate $|\boldsymbol{\epsilon}_m|$ present in the system.
- Compute ν_m with $|\boldsymbol{\epsilon}_m|$ according to the constitutive law, e.g., eq. (2.109).
- Compute new time step size Δt^* with the target collision frequency Ω_t and ν_m from eq. (5.4).

- Compute new lattice values from eq. (5.1).
- Relax Ma with under-relaxation factor λ or truncate it eq. (5.5).
- Recompute final adapted lattice values eq. (5.1) with the time step size obtained from Mach number Ma^{**} and the new viscosity bounds for eq. (2.112).
- Finally rescale distributions through eq. (5.2) and eq. (5.3).

The algorithm must be repeated in the subsequent time steps until the Mach number Ma^{**} converges and the desired target collision frequency is reached. This can also be realized in adaption intervals, allowing the solution to develop in between VAM steps.

In order to maximize the modeled viscosity range in the aforementioned example of Figure 5.1, an other possibility is to shift Ω_{max} to the maximal physical shear rate. Hence, ν_m should be computed with the maximal shear rate $|\boldsymbol{\epsilon}_m|$ in the system and Ω_t is set to Ω_{max}. The remaining steps in the algorithm are identical to the ones in the list above.

Actual physical boundaries, if known, can easily be implemented in the VALBM by setting the appropriate values in eq. (2.112). For example, the real upper bound on the viscosity for a shear thinning material corresponds to the horizontal line of the shaded area (physical problem) in Figure 5.1. In practical simulations, the collision frequency built with the real upper viscosity bound of the material (if known) may lead to values $\Omega_{min} \ll 1$, which in turn leads to instability of the LBM. Thus, a smaller viscosity bound must be chosen, such that, e.g., $\Omega_{min} = 1.0$. If the real upper bound results in a collision frequency of $\Omega_{min} \geq 1.0$, that value must be chosen for the limits in eq. (2.112). Hence, the implementation of the VALBM is straightforward for other non-Newtonian models, e.g., the Carreau-Yasuda or Cross model.

CHAPTER 6

Verification and Validation of the Viscosity Adaptive LBM

In this chapter, we consider the verification and validation of the viscosity adaptive LBM approach. First, the method is verified for the Hagen-Poiseuille flow for both the steady state and the transient case. Afterwards, the method is validated with experimental data from a propeller viscosimeter that was developed at the chair SAM.

6.1 Simple Case: Hagen-Poiseuille

6.1.1 Steady-State

For the verification of the proposed algorithm we investigate the flow through a 3D duct. We impose periodic boundary conditions in streamwise direction and the flow is driven through a body force similar to the simulations in Chapter 4. The lattice is constructed as to exactly retain the channel width and height, while using the simple bounce back boundary condition. For the sake of simplicity, we model the side walls of the duct to be frictionless. Therefore we can use the plane Hagen-Poiseuille solution for power-law fluids eq. (4.6). Again, the quality of the simulation is measured through the absolute global error eq. (4.4) and the simulations are run until the convergence criterion eq. (4.5) is fulfilled.

Mach Number Investigation

In a first approach, we want to investigate the convergence properties of the viscosity adaption method and its robustness to the starting Mach number, which is unknown user input. A series of simulations with starting Mach numbers from 0.1 to 10^{-4} are conducted

for a shear thickening fluid with $k = 250$, $n = 1.25$, $\rho = 1000$ and an acceleration of unity. The target collision frequency for the VAM is set to $\Omega_{min} = 1.0$. We set the relaxation parameter $\lambda = 0.05$ to gain an appropriate resolution of the Mach number development in this case. The VAM is started after 50 time steps and repeated every 10 time steps thereafter until the Mach number M^{**} converges according to eq. (5.6) with a residuum of $\delta = 10^{-8}$. Simulation continues until eq. (4.5) is satisfied. Figure 6.1 illustrates the Mach number development for the VALBM.

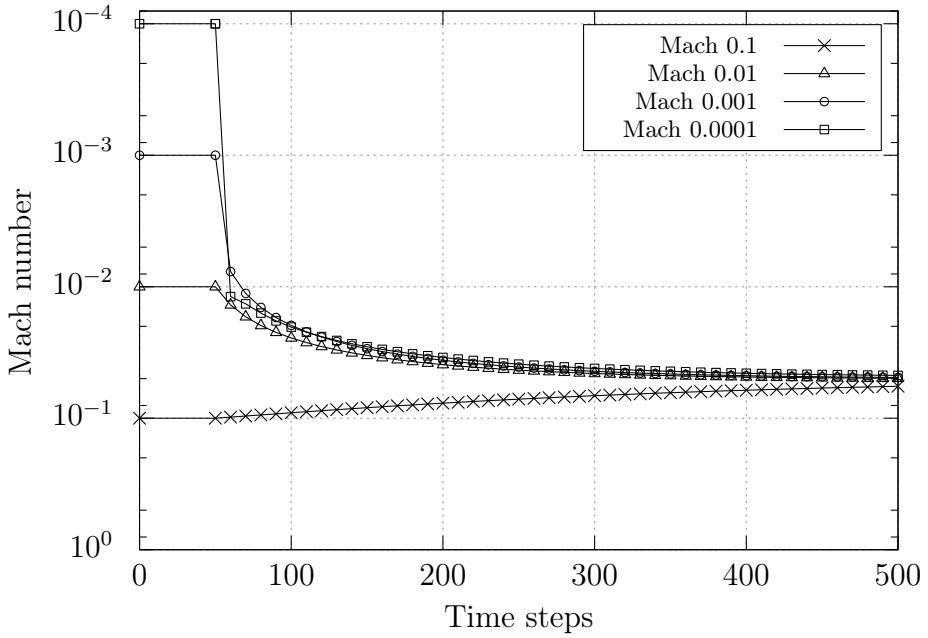


Figure 6.1: Mach number development over time for different staring Mach numbers.

The Mach number is plotted against the number of time steps. It is visible that the Mach number is adapted to the actual physical shear rate as the solution develops. All simulations yield a converged simulation Mach number of $Ma^{**} = Ma_C = 0.0439772$, independent of the Mach number at the beginning of the simulation. To verify the theoretical considerations in Section 5.2, we rerun the same series of simulations without viscosity adaption. Additionally, a standard LBM simulation is conducted for the converged Mach number Ma_C. Figure 6.2 shows the error in velocity for the different starting Mach numbers.

It can be seen that the simulation Mach number has a major impact on the error of the standard LBM approach. In fact, the simulation is performed more or less outside of the discrete physical problem (fig. 5.1). The quality of the simulation is improved as the simulation Mach number of the standard LBM approaches the regime of the converged

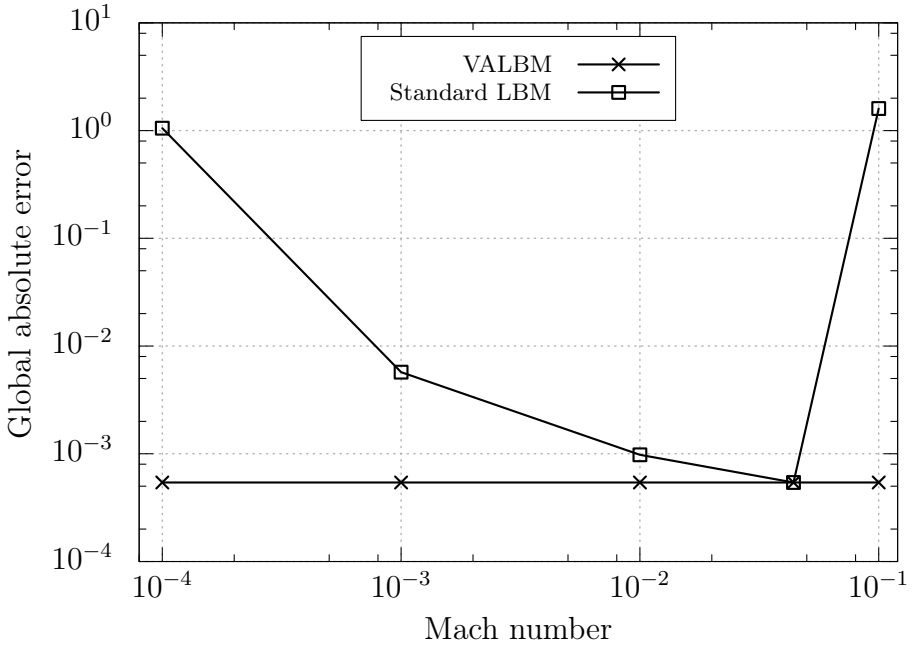


Figure 6.2: Error for Mach numbers corresponding to 6.1 for Standard LBM and the viscosity adaptive LBM.

Mach number Ma^{**} which is unknown *a priori*. All viscosity adaptive LBM simulations yield the same error independent of the starting Mach number, since the Mach number is changed to the converged Mach number by VAM (fig. 6.1). For the converged Mach number the standard LBM and the VALBM yield the same error. This confirms numerically that the viscosity is not influenced by the rescale operation as proposed in Chapter 5.

Accuracy

Next, the accuracy of the viscosity adaptive LBM is investigated. We perform simulations for a shear thinning $n = 0.75$, shear thickening $n = 1.25$, and a Newtonian fluid $n = 1.0$ with the boundary conditions and convergence creteria corresponding to the setup in Section 6.1.1. The lattice is successively refined to test the spatial convergence. Figure 6.3 shows a log-log plot of the global absolute error over the grid resolution.

Again, the respective result on each lattice is independent of the starting Mach number. Therefore only already Mach-converged results are shown. It is visible that the viscosity adaptive LBM ensures diffusive scaling of the lattice values [34]. This results in a global second order convergence as indicated by a line corresponing to a -2 slope.

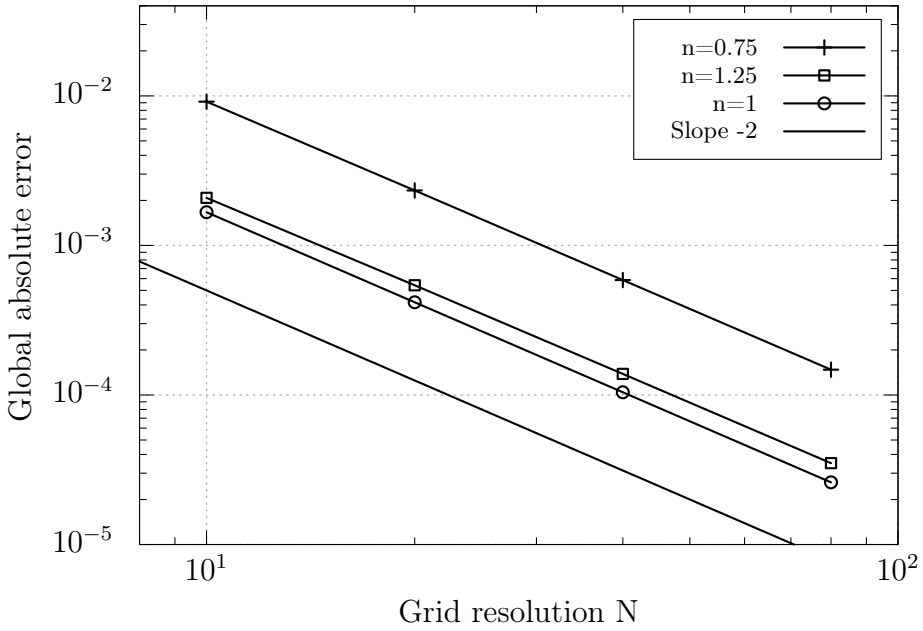


Figure 6.3: Accuracy of the viscosity adaptive LBM with $\Omega_t = \Omega_{min} = 1.0$ for shear thickening, shear thinning and Newtonian fluids on a successively refined lattice.

6.1.2 Transient

For the investigation of the viscosity adaptive LBM in transient cases, we consider the pulsatile flow through a 3D pipe with radius R. Again, the fluid is driven through a spatially constant body force and the simulation domain is periodic in streamwise direction. We impose the second order accurate no-slip *Mei* boundary condition to model the pipe wall [46]. In this case the Hagen-Poiseuille solution for the velocity of power-law fluids is given in radial coordinates [4]:

$$u_a(r) = \frac{n}{n+1} \cdot (\frac{dp}{dx}\frac{1}{2k})^{\frac{1}{n}} \cdot (R^{\frac{n+1}{n}} - r^{\frac{n+1}{n}}) \tag{6.1}$$

The pressure gradient is substituted for $\rho \cdot a$ at which the acceleration is made pulsatile with frequency ω through:

$$a(t) = cos(\omega \cdot t) \tag{6.2}$$

An important number in this context is the dimensionless dynamic similarity parameter α. It is known as the Womersley number and describes the relation of a characteristic time periodic flow frequency to viscous effects. It can be used to judge transient flows.

$$\alpha = R \cdot \sqrt{\frac{\omega}{\nu_{min}}} \tag{6.3}$$

In particular, a pulsatile flow can be considered quasi-steady if $\alpha < 1$. The instantaneous velocity distribution is, then, computed from eq. (6.1) using the instantaneous pressure gradient [41]. Because the viscosity is not constant in non-Newtonian simulations, we compute the maximal Womersley number through the minimal kinematic viscosity in the system to characterize the transient flow. The quality of the transient simulations is measured over one period T of the oscillation and computed in accordance with eq. (4.4):

$$E_t = \frac{1}{T} \sum_t \sqrt{\frac{1}{N} \cdot \sum_{\mathbf{x}} (u(\mathbf{x}, t) - u_a(\mathbf{x}, t))^2}. \tag{6.4}$$

We are interested in the properties of the VALBM in transient flow situations. For this purpose we choose a shear thinning material with $n = 0.75$, $k = 250$ and $\rho = 1000$ to simulate the Womersley flow. The pulsation frequency is set to $\omega = 3 \cdot 10^{-3}$. In contrast to the standard LBM the simulation Mach number is expected to vary because the VAM adjusts to the varying shear rates of the pulsatile flow. The target collision frequency is set to $\Omega_t = 1.0$. The adaption process is repeated every 10 time steps with an under-relaxation factor of $\lambda = 0.8$. Like in the steady state case, we conduct standard LBM simulations for constant Mach numbers 0.1, 0.05, 0.01, and 0.005. Figure 6.4 shows the results for the viscosity adaptive LBM simulation over one pressure oscillation, depicted in the upper plot. In the middle plot, the development of the Mach and Womersley number is shown. The maximal Womersley number as defined in eq. (6.3) is $\alpha = 0.123625$. With this value, we can justify the quasi-steady flow approximation. The maximal Mach number yields a value of 0.023859. In order to check if the choice of the adaption interval and under-relaxation factor is reasonable, a viscosity adaptive steady state simulation with the full pressure gradient is attached. It yields a converged Mach number of $Ma_C = 0.0238591$. An additional standard LBM simulation for Ma_C is carried out for the transient case. The plot at the bottom of Figure 6.4 shows the development of the global absolute error as defined in eq. (4.4). The error tends to zero as the pressure gradient vanishes and rises again with increasing values of the pressure gradient. However, the error decreases again when Δp approaches the turning points at $1/2\,T$ and T. This can be attributed to the quasi-steady approximation, because the derivative of the pressure gradient becomes zero. Hence, the VAM is able to converge comparable to the steady state case, which results in a more accurate result.

Simulation Quality

To judge the performance of the viscosity adaptive LBM, we compute the error E_t from eq. (6.4) and compare the results with standard LBM solutions. Figure 6.5 shows the time accumulated global absolute error plotted against the number of time steps needed to complete one pressure oscillation. The simulation with the Mach number of 0.1 shows the highest error. This error is composed of a high viscosity error. Also there could

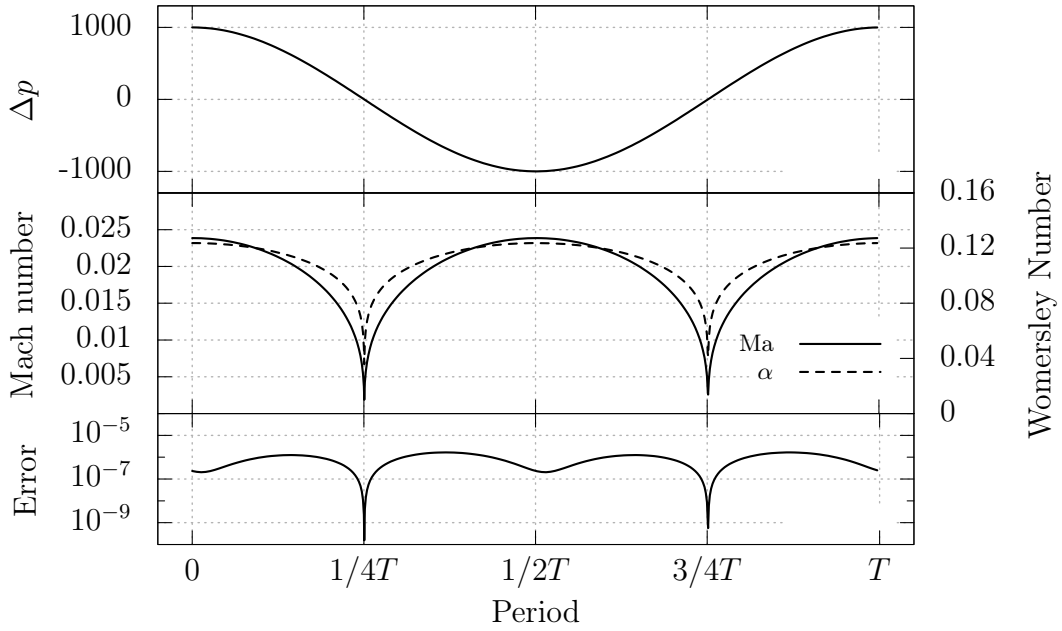


Figure 6.4: Top: Pressure oscillation, Middle: Development of the Mach and Womersley number, Bottom: Development of the global absolute error over one period of oscillation.

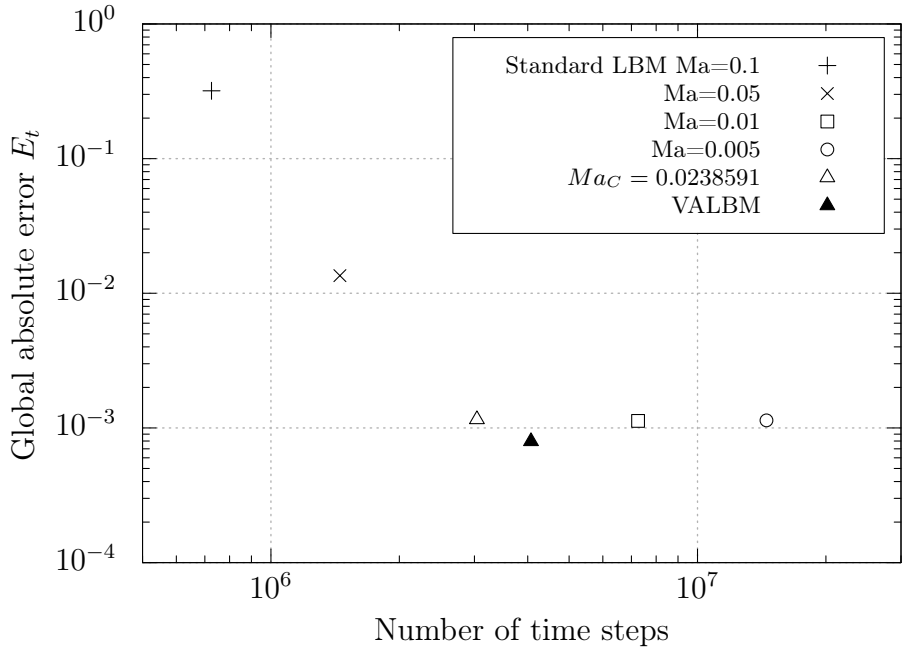


Figure 6.5: Plot of the error measured over one period against the number of time steps for the standard LBM with different simulation Mach numbers and VALBM.

be a heightened time discretization error present [40]. The simulations with the Mach numbers 0.01, 0.005 and Ma_C show comparable results, whereas the latter needs the lowest computational effort. The viscosity adaptive LBM yields the most accurate result with a slightly increased number of time steps as opposed to the standard LBM simulation with the converged Mach number. In this case, the additional computational effort in terms of the number of time steps amounts to 33.8%. Therefore, the accuracy is enhanced by 44.5%. Again, it should be pointed out that the Mach number Ma_C is unknown *a priori*. If the viscosity adaptive simulation is compared to the standard LBM simulations with $Ma = 0.01$ and $Ma = 0.005$ the VALBM simulation is more accurate as shown above and accelerated in terms of time steps by 178.2% and 336.4%, respectively.

6.2 Complex Case: Propeller Viscosimeter

In the previous section, we verified the VALBM for steady state and transient flows. This section is concerned with the validation of the VALBM. For this purpose, simulation results are compared with experimental data from a propeller viscosimeter, which was developed at the chair SAM. Propeller viscosimeters are an alternative to classical rotational viscosimeters. Fields of application are mainly the measurement of the viscosity of inhomogeneous Newtonian or non-Newtonian fluids. The reason for this, is that every obstacle in the small gap between the two cylinders of an rotational viscosimeter would strongly adulterate the result. Obviously a propeller viscosimeter does not have such a drawback.

6.2.1 Experimental and Virtual Setup

Figure 6.6 shows an assembly drawing of the propeller viscosimeter. The stirrer is an anodized two wing impeller (A) with a diameter of $D_i = 172.3$ mm that is connected with a torque measurement device (C) through a concentrically supported shaft. Also, the agitator speed can be recorded by the measurement unit. The engine (C) of the propeller viscosimeter is connected with a gear box in order to realize small revolutions per minute. Additionally, a temperature sensor is used to monitor the temperature of the fluid. The whole device can be mounted on a cylindrical container filled with the experimental fluid. Through holes in the mounting plate (B) the experimental rig can operate in baffled or unbaffled mode. Figure 6.7 shows a picture of the real test rig where the propeller device is operating in a cylindrical container filled with water. The mapping of the setting of the real test rig to the simulation domain is visualized in Figure 6.8. Here, the free surface, corresponding to the filling level of the experimental fluid, is modeled as a frictionless wall. This is justified, because of the small revolutions per minute of the impeller, i.e., a vanishing Froude-number $Fr \approx 0$. On the remaining walls of the cylindrical container, a second order accurate no-slip wall boundary condition is imposed. Since the agitator speed is low, the height of the real container can be safely shortened, in order to reduce the computational effort. All simulations and measurements are related to the unbaffeled propeller viscosimeter. Thus, the shaft and the propeller are modeled as a frozen rotor. To this end, a second order accurate no slip wall boundary condition is used. The walls are then treated as moving walls, where the circumferential speed, corresponding to the agitator speed, is imposed.

For the purpose of validation, three different non-Newtonian fluids will be simulated. These are mixtures of water and xanthan. Such a solution results in a shear thinning non-Newtonian fluid that is well described with the power-law model. Table 6.1 shows the flow consistency index k and the flow behavior index n used in eq. (2.109) to model the fluids. These correspond to the three materials in [53] for which simulation results are shown in addition to experimental data.

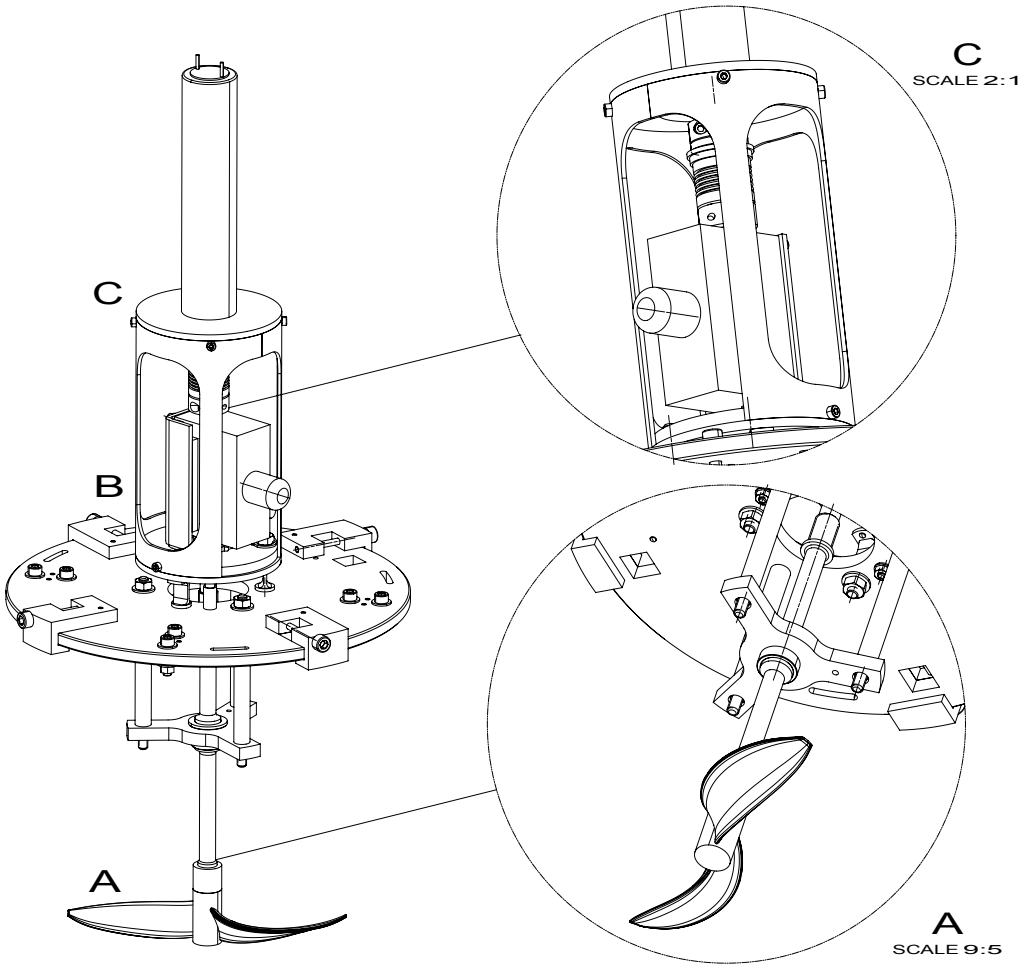


Figure 6.6: Assembly drawing of the propeller viscosimeter

Table 6.1: Non-Newtonian validation fluids from [53]

Material	Consistency index k	Behavior index n
$r1$	1.916	0.293
$r4$	6.344	0.203
$r7$	22.130	0.132

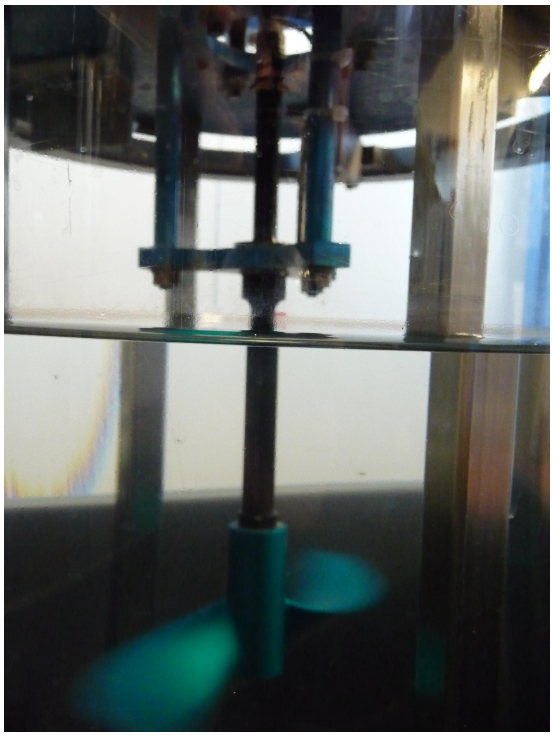


Figure 6.7: Real test rig

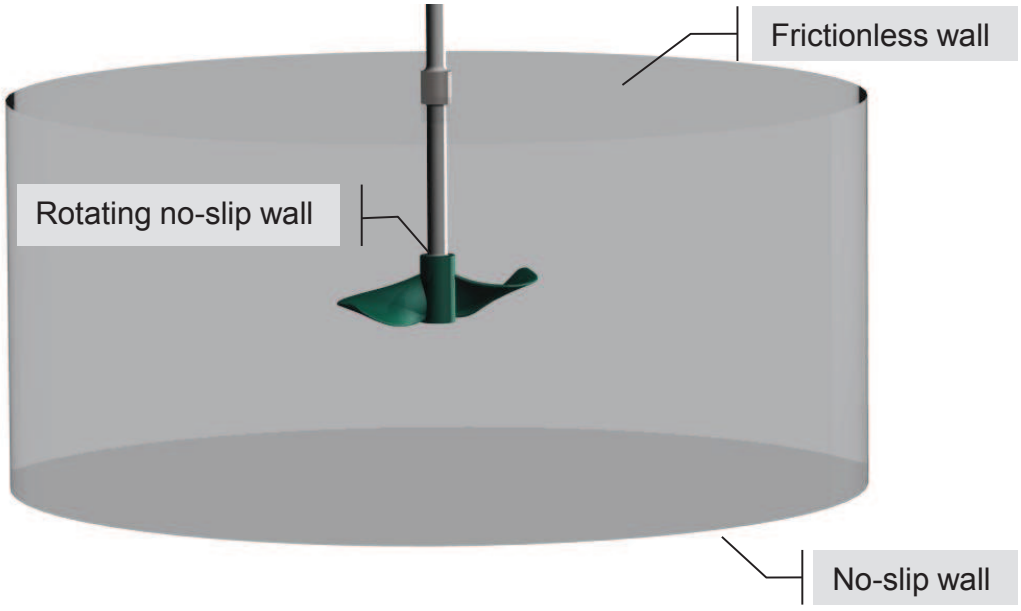


Figure 6.8: Simulation setup

6.2.2 Discretization

In this geometrically complex case, the full capabilities of *SamGenerator*, which has been developed in the course of this thesis, can be exploited. The lattice is generated automatically with minimal user input. *SamGenerator* expects a triangulated representation of the geometry. A straightforward way is to create the input geometry in the *stereolithography (stl)* format. Unfortunately, the common CAD programs do not provide enough freedom to create a high quality triangulation. Those CAD *stl* representations may be unhesitatingly used for visualization purposes, but do not necessarily fit for the purpose of simulation. Hence, a boundary mesh in the *msh* format consisting of triangle elements is used to represent the geometry of Figure 6.8 in order to minimize E_g of eq. (2.92). Additionally, *SamGenerator* offers the possibility to refine parts of the computational domain with the help of boxes or arbitrarily shaped auxiliary geometries in the *stl* format. This capability is used to refine the region around the propeller in order to locally enhance the spatial resolution. As it was shown in Section 2.5.2, a nested time stepping algorithm is used, when working with local grid refinement. Therefore, the number of stream and collide operations needed to complete one time step, depends on the refinement level chosen. A level two lattice is used for the simulations, as a trade-off between resolution and computational effort. This means, the computational domain is discretized with a base resolution and the region around the propeller is discretized with a resolution that is four times finer than the base resolution. The transition between the regions of different resolution is smoothed. Figure 6.9 shows a slice of the lattice, viewed from the side of the propeller viscosimeter. One can see the domain refinement around the impeller and the shaft. It is chosen sufficiently large, such that the multi-level approach from section 5.4 of the VAM can be used or each refinement level can be treated individually. The frictionless wall on the top is also refined to ensure a consistent height of the cylinder. This is necessary since the frictionless approximation is realized through the simple bounce back scheme.

Figure 6.10 shows a top view of the lattice. Illustrated is a slice through the lattice at the height level of the impeller. In particular, the differently resolved patches are shown by their 3D cell representation to highlight the refinement ratio. As one can see, a cylindrical auxiliary geometry was used for the domain refinement, for which the sides of the cylinder are cropped to avoid an unnecessary amount of level two nodes. The axis aligned regular lattice leads to an apparently staggered approximation of the impeller. Second order accurate wall boundary conditions incorporate the real position of the impeller geometry into the simulation. A comparison of the two differently resolved lattices in Figure 6.11 shows the effect of the resolution on the approximation of the impeller geometry. The resolution is expressed through the ratio $D_i/\Delta x$, which gives the number of lattice nodes in terms of the base resolution along the impeller diameter.

For the sake of completeness, the computational performance of *SamGenerator* for this validation case should be mentioned. With a resolution of $D_i/\Delta x = 60$, a total amount of 17.812.591 cells were created during construction. For the simulation, the lattice contains a

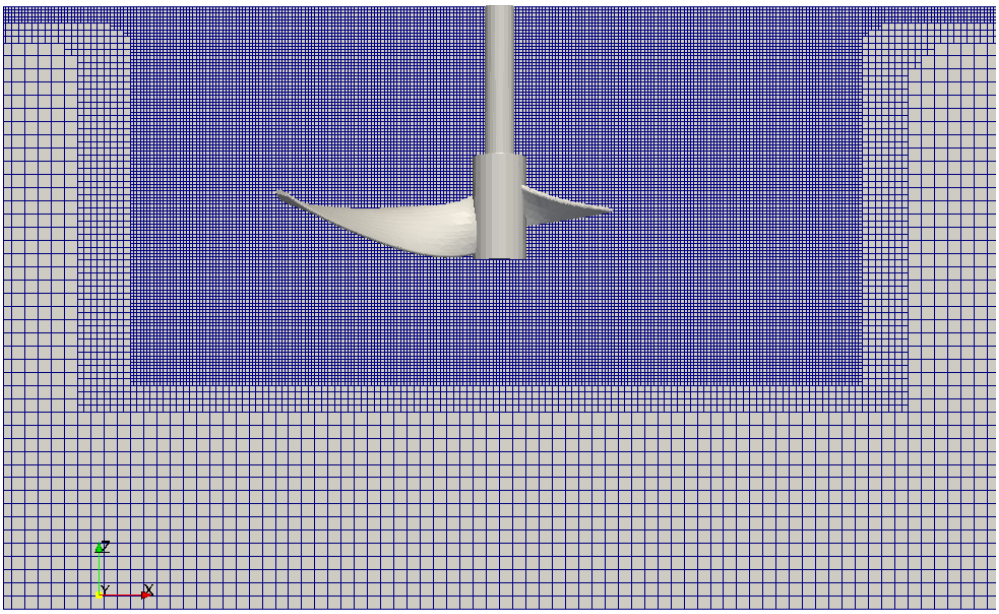


Figure 6.9: Level two lattice (side view)

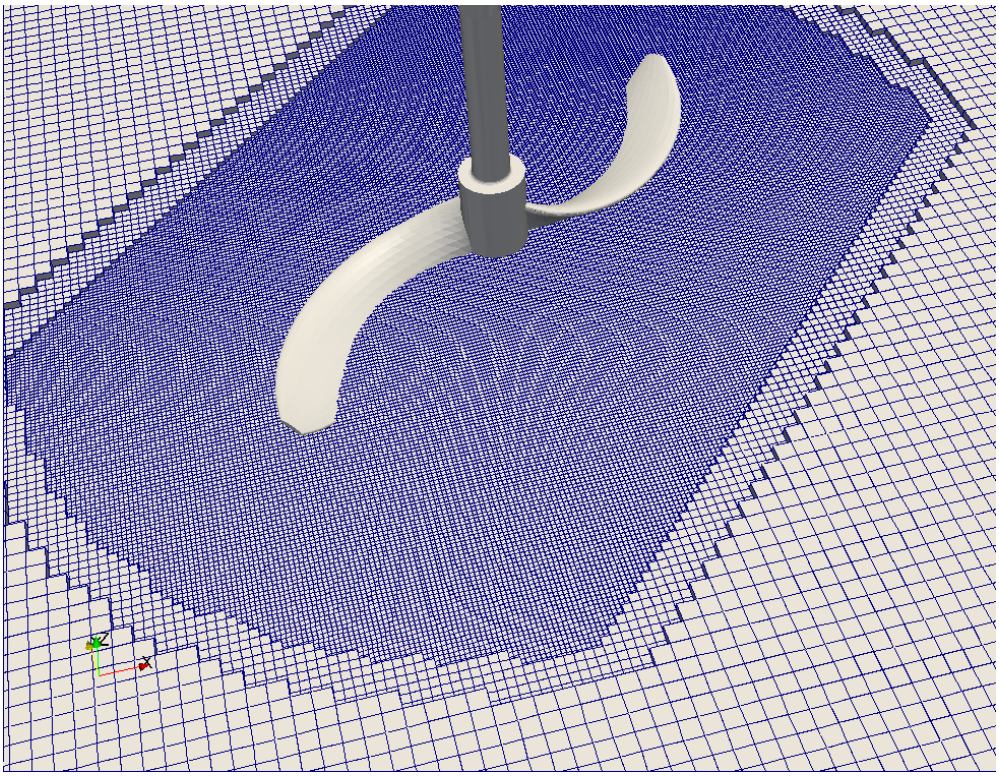
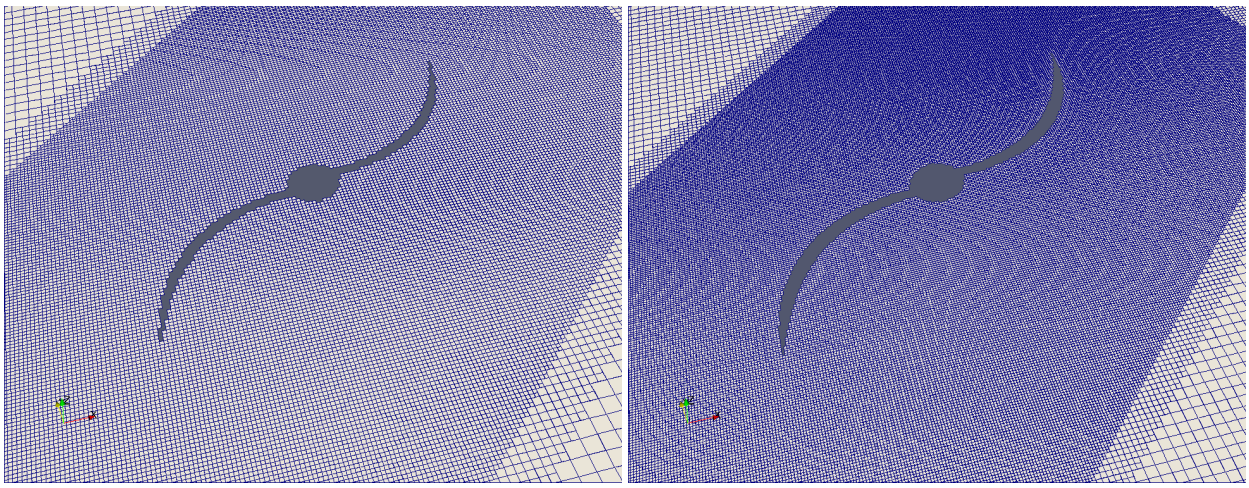


Figure 6.10: Level two lattice (top view)



(a) $D_i/\Delta x = 40$ (b) $D_i/\Delta x = 60$

Figure 6.11: Resulting approximation of the impeller geometry with two differently resolved lattices.

total amount of 15.280.242 cells. This equates to a total amount of 15.921.212 lattice sites. The lattice generation was performed on a single CPU and a total amount of wall clock time, including I/O operations, of 5.61 minutes was needed to complete discretization.

6.2.3 Simulation

Validation data consist of the torque characteristic for the three different non-Newtonian fluids investigated. The agitator speed ranges from about 5 to 60 rpm ($0.08 - 1.00$ 1/s). Lower agitator speeds could not be realized with the present construction of the propeller viscosimeter. In particular, we simulate operating points, where the agitator speed n takes values of the form $n = 0.1 \cdot x$, $\forall x \in [1, 10]$ with unit 1/s. The torque is evaluated through the momentum exchange method, introduced in Section 3.2.3.

In order to experimentally measure the real torque characteristic of the propeller viscosimeter, one has to determine that part of the torque, which arises from unintended friction. It is then excluded from the measured torque. Although not explicitly documented in [53], it can be concluded from the measurement data, that a constant value for the friction of order 10^{-3} Nm was subtracted from the torque measurement for every operating point. Hence, one can expect an error in the torque measurements of [53] that is $\mathcal{O}(1)$ in the calculation of the real torque characteristics. Since the friction to torque ratio is high for fluids that exhibit small absolute values of torque, e.g., fluid *r1*, it is clear that these data have to be treated with some degree of caution.

Standard LBM Simulations

The choice of the simulation Mach number and the associated shift in the modeled viscosity range lies at the heart of the viscosity adaptive LBM. In order to investigate the impact of the simulation Mach number on the torque characteristic of the propeller viscosimeter, a series of simulations for constant Mach numbers from 0.1 to 10^{-4} is conducted with the standard LBM approach.

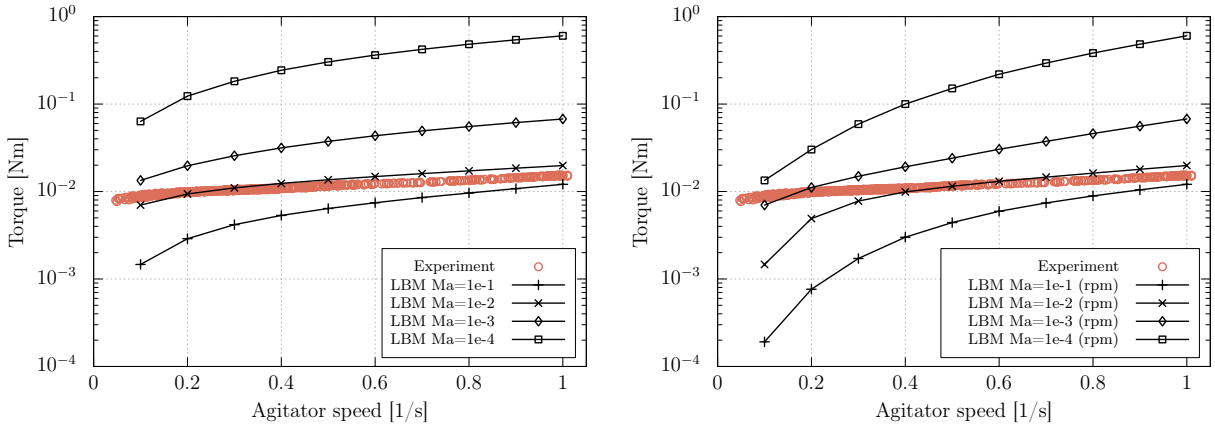
The characteristic flow velocity for the calculation of the simulation Mach number is taken to be the theoretically maximal expected velocity at the tip of the impeller blades,

$$Ma = \frac{U^{Tip}}{c_s}. \tag{6.5}$$

Here, we distinguish two cases. In the first case, we keep the simulation Mach number constant, where $U^{Tip} = D_i \cdot \pi \cdot n$ is computed from the operating point with maximum agitator speed, i.e., $n = 1.0$ 1/s. This implies, that the effective Mach number at each operating point is linearly increased, since the agitator speed is linearly increased, until the effective Mach number and the simulation Mach number coincide. The term effective Mach number is used here, to denote that Mach number, which is computed with the maximum flow speed that really appears in the simulation for each operating point. In the second case, both the effective Mach number and the simulation Mach number are held constant, through linear increase of U^{Tip} throughout the operating points, according to eq. (6.5). The Mach investigation was performed for the material $r4$ with a lattice resolution of $D_i/\Delta x = 60$. Since a level 2 lattice is used, the multi-level approach is used to determine the collision frequencies in the different grid levels.

As it was proposed in Section 5.1, the simulation results crucially depend on the simulation Mach number. This is demonstrably shown in Figure 6.12, where the torque is plotted against the agitator speed. Since the various simulations give very different values for the torque, the torque data is plotted in logarithmic scale. This should be taken into account when judging the results. In the first case, Figure 6.12(a), where the effective mach number is varied linearly with the agitator speed ($U^{Tip} = const$), the torque characteristic with $Ma = 10^{-2}$ provides the best results. For the first operating point the torque is a little underestimated, whereas at higher revolutions per minute the torque is somewhat overestimated. The simulations with $Ma = 10^{-1}$ approach the measurement data for high rpm, but essentially underestimate the torque. The remaining simulations are far from the measurement data and give errors of several hundred percent. In the second case, where the effective Mach number is constant, the simulation results for $Ma = 10^{-2}$ fit to measurement data only for higher revolutions per minute. For lower agitator speeds the results for $Ma = 10^{-3}$ provide reasonable results. Again, the results for the simulation with $Ma = 10^{-1}$ approach the torque measurement for high rpm, while the torque is still underestimated. For the agitator speed $n = 1.0$ 1/s, the results of Figure 6.12(a) and Figure 6.12(b) are identical, because the simulation Mach numbers are the same by

definition.



(a) $U^{Tip} = const.$ (b) $U^{Tip} = f(n)$

Figure 6.12: Torque characteristic for different simulation Mach numbers using standard LBM.

The results from the Mach number investigation confirm, that it is possible to determine a suitable simulation Mach number for the standard LBM for each operating point with the aid of precursor simulations, as we have previously shown in [9].

Viscosity Adaptive LBM Simulations

The flows considered in Chapter 4 and Section 6.1 are force driven and due to the way the lattice is constructed, the discretized physical problem does not suffer from vanishing strain rates. This can be easily seen from Figure 4.8. Moreover, the viscosity range, that physically appear in the discretized channel and pipe flows, is a subset from what can be modeled with the LBM within the stability bounds. By contrast, there is a predominantly region with vanishing strain rate tensor norm for the problem considered in this section. Therefore, the viscosity range that can be modeled is a subset of the range of the discretized physical problem.

Since the flow of the shear thinning materials is driven by the impeller, the maximum shear rate is expected at the tip of the blades. Thus, it is expedient to shift the maximal permissible collision frequency towards the maximum shear rate of the system. We use the MRT approach for the simulations, in order to enlarge the modeled viscosity range. For the same reason, we treat this case with a special version of the multi-level approach introduced in Section 5.4. To this end, the target collision frequency is set to $\Omega_t = 1.985$ in each level and the lower bound for each lattice patch is computed from $\Omega_{min} = 1.05$, set in the finest level. The influence of the multi-level approach will be discussed in the sequel.

The simulations conducted in the previous subsection with the standard LBM approach are rerun using VALBM. All core settings of the LBM remain the same, except for the simulation Mach number, which is set to a starting value of $Ma = 10^{-3}$, built with the U^{Tip} of the operating point $n = 1.0$ 1/s and changed afterwards by the viscosity adaption method. The distributions are initialized with zero velocity equilibrium values and the VAM algorithm is started after 300 time steps. Figure 6.13 shows the results for the torque developing. Depicted are the best results of the standard LBM simulations with $Ma = 10^{-2}$ according to Figure 6.12(a) and the results using VALBM, together with experimental data.

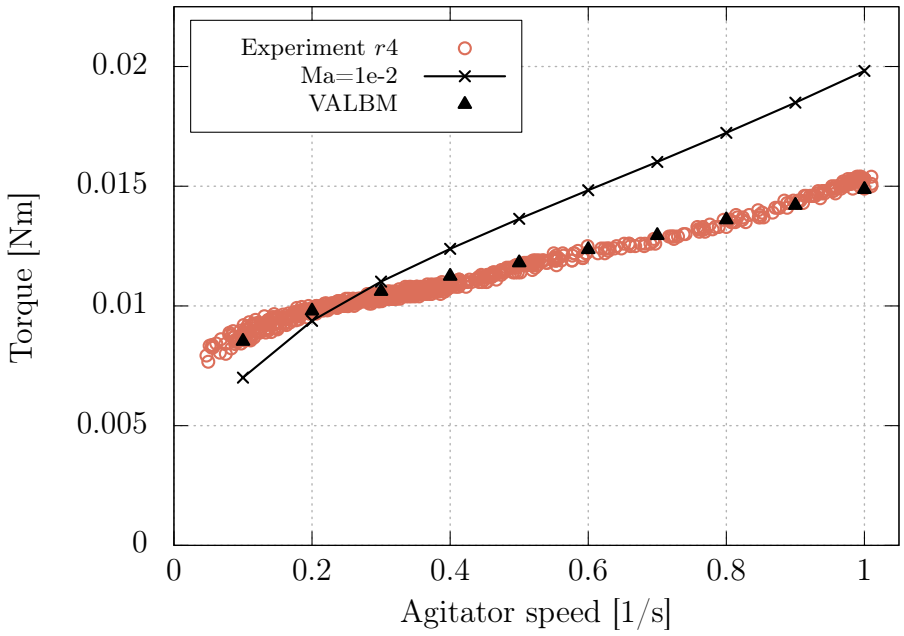


Figure 6.13: Torque characteristic of fluid $r4$ simulated with the standard LBM with $Ma = 10^{-2}$ and VALBM.

It should be noted that this time the ordinate is not in log scale. The complete torque characteristic is reproduced by the VALBM with excellent agreement to experimental data. As stated before, the standard LBM shows good agreement for the operating points $n = 0.2$ 1/s and $n = 0.3$ 1/s. When the agitator speed is increased, there is a different range of shear rates present. The standard LBM cannot account for the changing viscosity range. Hence, the torque characteristic is not captured by the standard LBM simulation. By contrast, the VALBM consistently adapts to the various operating points. The viscosity adaptive LBM results for the remaining fluids $r1$ and $r7$ are shown in Figure 6.14 and Figure 6.15, respectively.

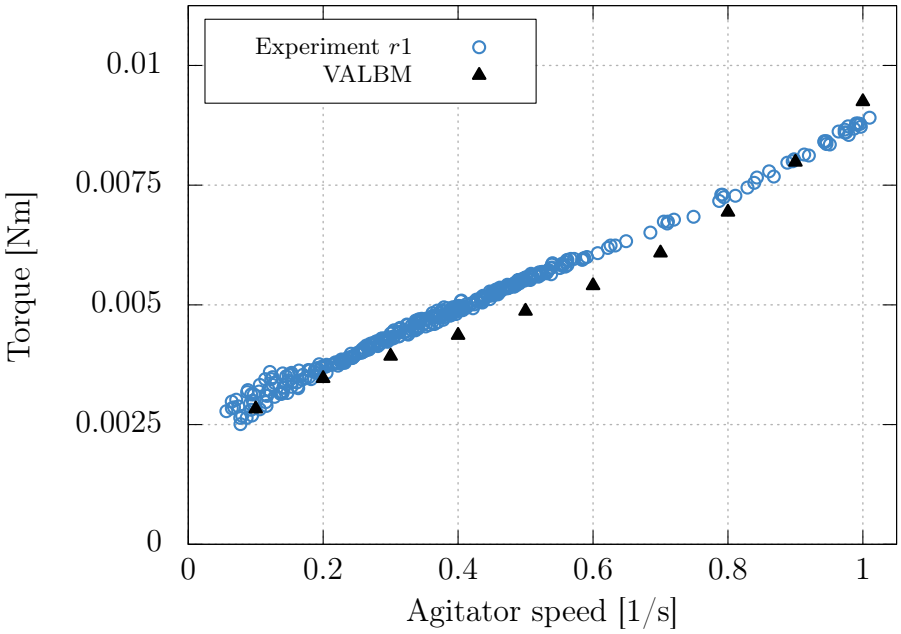


Figure 6.14: Torque characteristic of fluid $r1$ simulated with the VALBM approach.

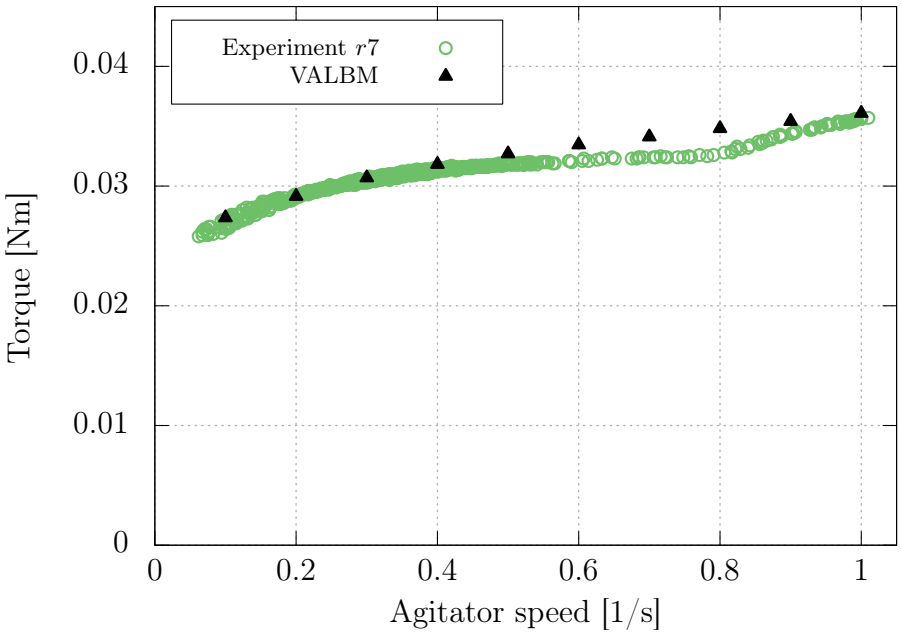


Figure 6.15: Torque characteristic of fluid $r7$ simulated with the VALBM approach.

The torque characteristic for fluid $r1$ is also reproduced with very good agreement to experimental data. For medium agitator speeds, the results from the experiment and the simulation show a deviation up to 10%. As already mentioned, this can be attributed to the high friction to torque ratio, for which the results from [53] should be treated with caution. Concerning the results for fluid $r7$, one can again observe an excellent match of the simulation results to the real measurements of the torque. There is only a small range in the vicinity of $n = 0.8$ 1/s for which a difference of the simulation and the experiment is visible. The deviation of the results in this agitator speed zone is smaller than 5% and therefore within experimental uncertainty.

Mach Number Investigation

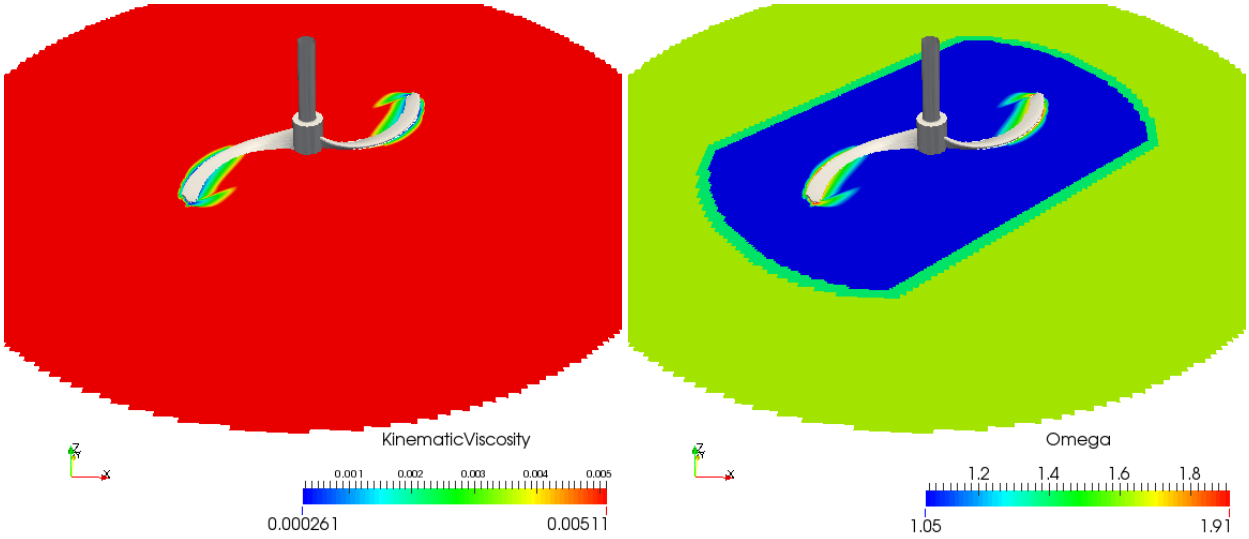
The Mach number investigations done in the previous subsection for the fluid $r4$ using standard LBM are also conducted for the remaining non-Newtonian fluids *r1* and *r7*.

The influence of the simulation Mach number on the modeled viscosity range can be illustrated graphically through the distribution of the kinematic viscosity as well as the collision frequency in the simulation domain. Exemplary, the operating point $n = 0.5$ 1/s for the fluid $r7$ is analyzed. The results of the standard LBM with a simulation Mach number of $Ma = 10^{-2}$ is given in Figure 6.16, which shows a representative slice through the fluid domain at height level of the impeller. The color bars and values shown are local to the slice. Statements on the global values will be made in the sequel in the course of the multi-level investigations.

One can see in Figure 6.16 (a), that only a small portion around the tip of the propeller blades is actually modeled. The rest of the fluid domain has a constant value of the upper kinematic viscosity bound. One can state, that the upper viscosity limit is already hit in the immediate proximity of the impeller, which suppresses the non-Newtonian modeling. Since the collision frequency distribution shown in Figure 6.16 (b) is connected with the kinematic viscosity, the same statements hold. One can see that the upper bound on ν causes different collision frequencies in each level according to eq. (5.7). The simulation predicts a value for the moment force of 0.02214 Nm that is too low, which can be seen from Figure 6.15.

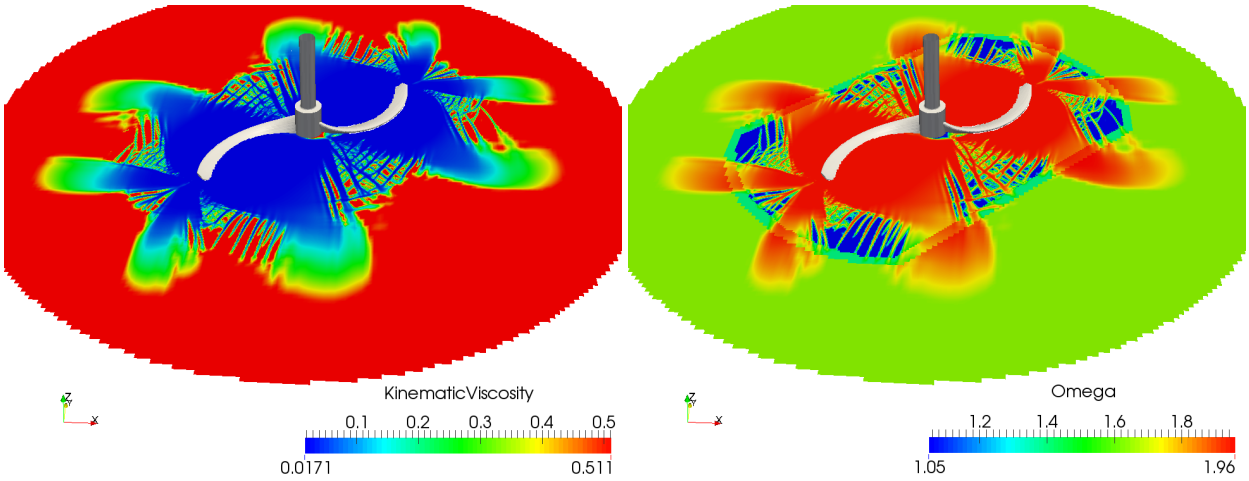
Now, standard LBM simulations for the same operating point are conducted with a Mach number of $Ma = 10^{-4}$. The distribution of the kinematic viscosity and the collision frequency are depicted in Figure 6.17.

In this picture, one can see that there is a large portion of the fluid domain that takes the value of the lower viscosity limit. The value itself is about three times higher than the maximum viscosity bound of the simulation with a Mach number of $Ma = 10^{-2}$. As a consequence of the multi-level approach, there is a continuous distribution of the kinematic viscosity throughout the lattice interfaces. Although the kinematic viscosity distribution is smooth, the collision frequency shows discrete values at the grid transitions. The upper viscosity bound is hit far away from the impeller, due to vanishing strain rates. As a result,



(a) Kinematic viscosity distribution (b) Collision frequency distribution

Figure 6.16: Standard LBM simulation of fluid $r7$ for operating point $n = 0.5$ 1/s with $Ma = 10^{-2}$.

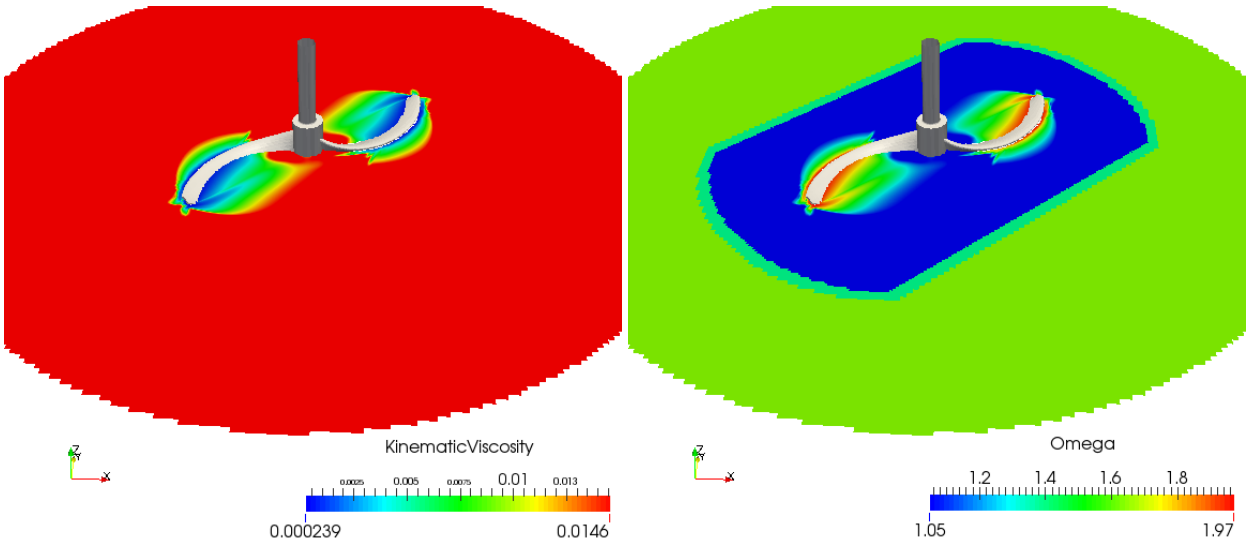


(a) Kinematic viscosity distribution (b) Collision frequency distribution

Figure 6.17: Standard LBM simulation of fluid $r7$ for operating point $n = 0.5$ 1/s with $Ma = 10^{-4}$.

the simulation with $Ma = 10^{-4}$ gives a value for the torque of 0.17112 Nm, which is far too high.

By contrast, Figure 6.18 shows the appropriate distributions in the case of a viscosity adaptive simulation. A moderate region around the propeller is modeled. The results show a upper viscosity bound which is approximately three times the upper bound of the standard LBM simulation with $Ma = 10^{-2}$, while the lower bounds are the same. We will discuss this in more detail in the following subsection. The VALBM predicts a torque of 0.03228 Nm, which is in very good agreement with experimental data as it can be seen in Figure 6.15.



(a) Kinematic viscosity distribution

(b) Collision frequency distribution

Figure 6.18: Viscosity adaptive LBM simulation of fluid $r7$ for operating point $n = 0.5$ 1/s.

Finally, since the Mach number is changed by the VAM, Figure 6.19 shows the evolution of the Mach number in VALBM simulations for all fluids and all operating points investigated. The simulations for the fluids $r1$ and $r4$ show a Mach number evolution that is linear for small agitator speeds. When the agitator speed is increased, there is a rapid increase of the Mach number visible. This is even more pronounced in the case of the fluid $r1$. The simulation Mach number for the fluid $r7$ is almost linearly increased with the agitator speed. This numerically justifies our assumptions made earlier in [9].

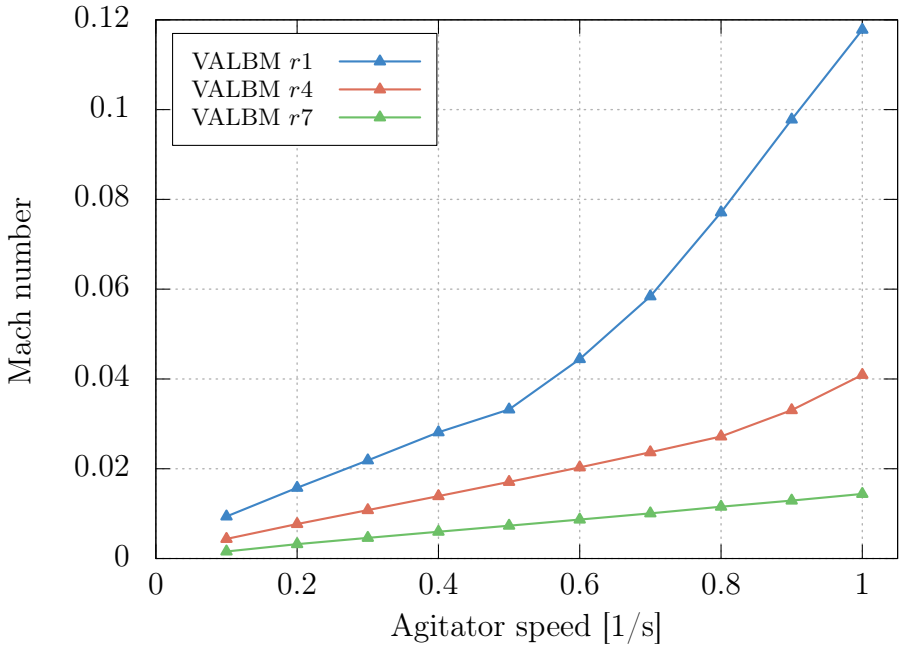


Figure 6.19: Evolution of the Mach number in VALBM simulations of all fluids and all operating points investigated.

Multi-Level Investigation

In this subsection, an investigation of the multi-level approach is conducted on the basis of a medium operating point of $n = 0.5$ 1/s for the fluid $r7$ with a simulation Mach number of $Ma = 10^{-2}$. Recalling the multi-level approach introduced in Section 5.4, special care must be taken to keep simulations with local grid refinement stable. A consistent way to do that, is to limit the modeled viscosity range to the intersecting set of the differently resolved lattice patches. Therefore, the stability borders are redefined in eq. (5.7) to ensure continuity of the Re number throughout the simulation domain. It is immediately clear, that this approach narrows the modeled viscosity range, since the intersecting set diminishes with the number of grid refinement levels used.

As already stated in the previous sections, we can expect the best results by setting the target collision frequency for the viscosity adaption method to $\Omega_t = \Omega_{max}$. Against the backdrop of the multi-level approach, we can investigate the full multi-level approach and a modified version, where we keep the values of $\Omega_{max} = 1.985$ constant in each level in order to enlarge the modeled viscosity range. The lattice is refined, such that both approaches can be analyzed, since the interface is far enough away from the impeller as can be seen in Figure 6.10.

Table 6.2 summarizes the global simulation results for the values of the collision

frequency bounds, the viscosity bounds, and the Torque on the impeller. The entries LBM and VALBM refer to the modified multi-level approach, where the upper collision frequency bound is constant in all levels. By default, the minimal collision frequency is set

Table 6.2: Influence of the different approaches on the viscosity bounds and the torque.

Approach	Ω_{min}	Ω_{max}	ν_{min}	ν_{max}	Torque [Nm]
Multi-level LBM	1.05	1.941	0.0001708	0.0051126	0.02214
LBM	1.05	1.952	0.0001396	0.0051126	0.02215
VALBM	1.05	1.985	0.0001219	0.0145594	0.03228

to $\Omega_{min} = 1.05$ for all simulations in the finest level and computed according to eq. (5.7).

First of all, the standard LBM simulations are compared to each other. The modeled viscosity ranges are transferred to a log-log plot of the kinematic viscosity against the shear rate in Figure 6.20, where the Ostwald de Waele power law with the values for fluid $r7$ from Table 6.1 is drawn.

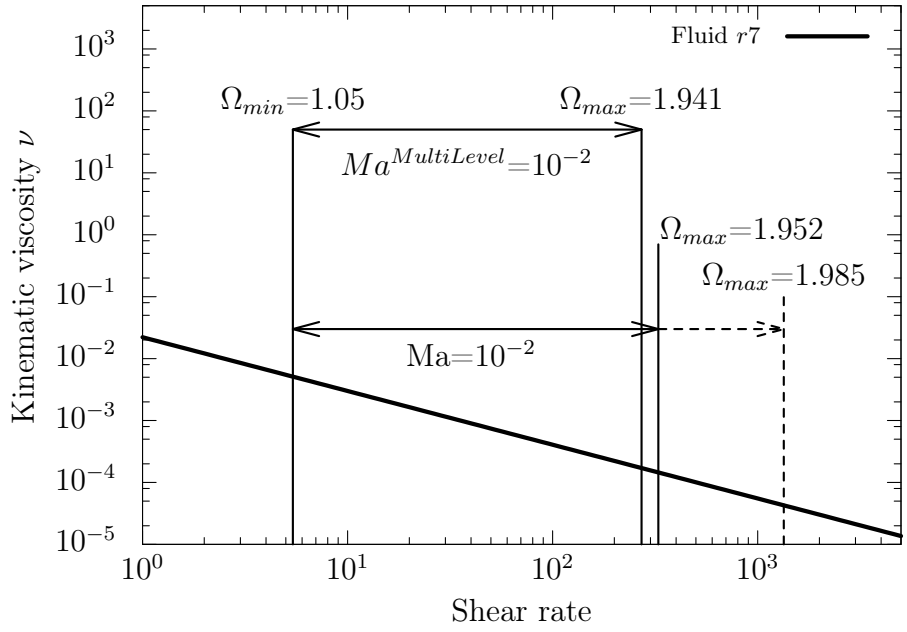


Figure 6.20: Modeled viscosity range of fluid $r7$ by the standard LBM with $Ma = 10^{-2}$ and different multi-level approaches.

It is shown, that for the modified version of the multi-level approach, a simulation Mach

number of $Ma = 10^{-2}$ is disadvantageous. The range that could be modeled with the maximum permissible collision frequency is drawn with a dashed line. It is obvious, that the upper limit is not exhausted, since the actual maximum value of the collision frequency in this simulation is $\Omega_{max} = 1.952$. Consequently, the lower collision frequency bound results in relatively low value of the upper viscosity bound. The full multi-level approach truncates the viscosity range prematurely for high shear rates, while the lower collision frequency bound is the same as for the modified version. One can see from the values in the last column of Table 6.2, that a higher value of the maximal collision frequency in the system has very little effect on the torque. This is self-evident, since only even smaller viscosities are permitted, which do not significantly influence the torque, albeit the flow itself may be slightly affected.

The effect of the viscosity adaption method is shown in Figure 6.21, where both simulations are conducted with the modified multi-level approach. One can see, that the maximum permissible collision frequency is shifted towards the actual maximum shear rate of the system in the VALBM simulation. This is archived through adaption of the simulation Mach number as depicted in Figure 6.19. Consequently, the Ω_{min} limit is shifted towards lower shear rates.

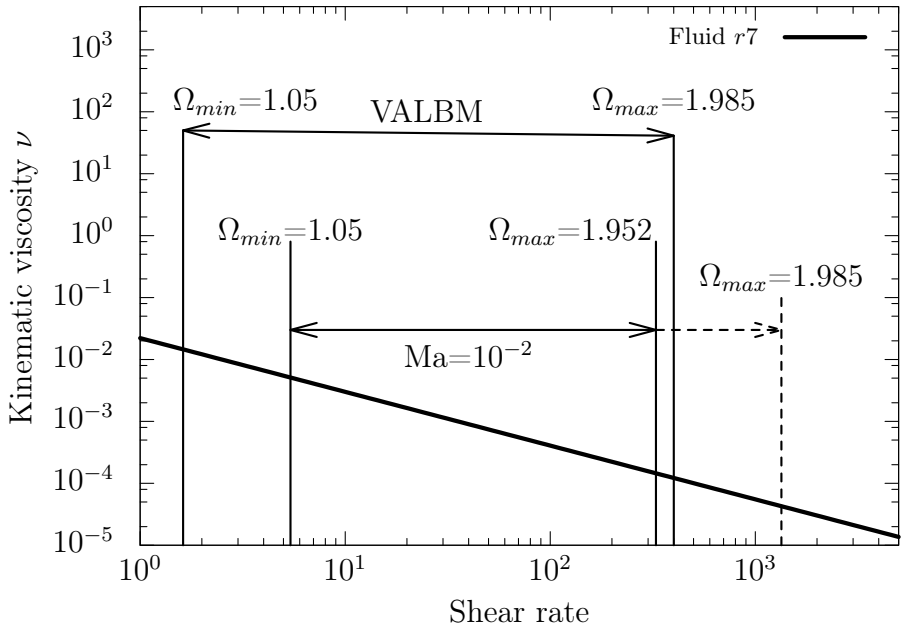


Figure 6.21: Modeled viscosity range of fluid $r7$ by VALBM and the standard LBM with $Ma = 10^{-2}$.

The shift of the upper viscosity bound, caused by the shift of the lower collision

frequency bound Ω_{min}, has major impact on the torque. In low strain rate regions around the impeller, e.g., near the axis of rotation, the fluid is more viscous, which results in a higher overall value of the moment force. Figure 6.21 exemplifies, that the VAM consistently adapts the modeled viscosity range to the actual range of the physical problem, while preserving the stability of Lattice Boltzmann simulations even in cases with local grid refinement.

Concluding Remarks

The results in this section demonstrate the validity of the viscosity adaptive LBM. The Mach number for the simulations of the propeller viscosimeter operating at different revolutions per minute was automatically and consistently adjusted by the VAM algorithm. The results are independent of the Mach number the simulation was started with. All moment forces at each operating point fit excellently to experimental data.

CHAPTER 7

Sensitivity Analysis of the VAM

In this chapter, the influence of the various parameters of the VAM is investigated. Considered in detail, is the adaption interval, the relaxation parameter λ, and the Mach number threshold δ. All of these parameters are studied for the complex propeller viscosimeter case from Section 6.2. We choose a median operating point, i.e., for fluid $r4$ with an agitator speed of $n = 0.5$ 1/s. All simulations are conducted with a starting simulation Mach number of $Ma = 10^{-3}$ and a resolution of $D_i/\Delta x = 50$.

Moreover, the role of the target collision frequency is analyzed with an in-depth study of the accuracy of non-Newtonian Lattice Boltzmann simulations in Section 7.4 for the Hagen-Poiseuille channel flow.

7.1 Influence of the Adaption Interval

In this section, the influence of the adaption interval on the torque and Mach number development is analyzed. To this end, four different intervals, ranging from 2 to 250 time steps are tested.

The minimal possible adaption interval is 2 time steps, since we need to maintain a time consistent time loop, especially in the nested time stepping case. Throughout the investigation, the remaining parameters are kept constant and are set to $\lambda = 0.04$ and $\delta = 10^{-4}$, respectively. Figure 7.1 shows the torque development over time for the first time steps after the start of the VAM algorithm at time step 300.

The interval of 2 time steps produces the highest amplitude of torque change, right at the start of the VAM. This can be a possible trigger for instabilities. On this account, the amplitude of the torque change can be attenuated with an increased adaption interval.

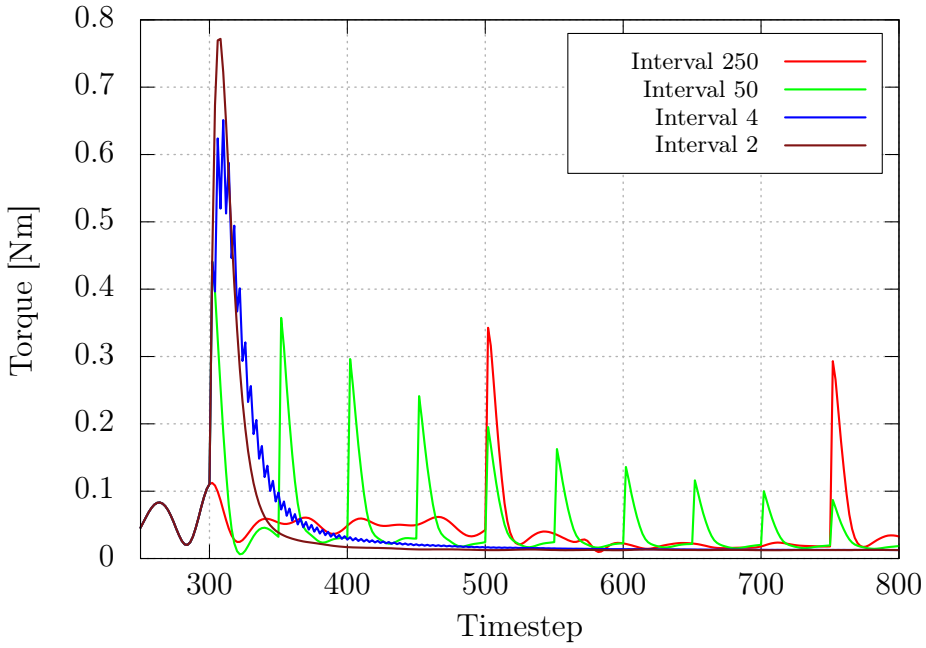


Figure 7.1: Influence of different adaption intervals on the torque at the start of the VAM algorithm at time step 300.

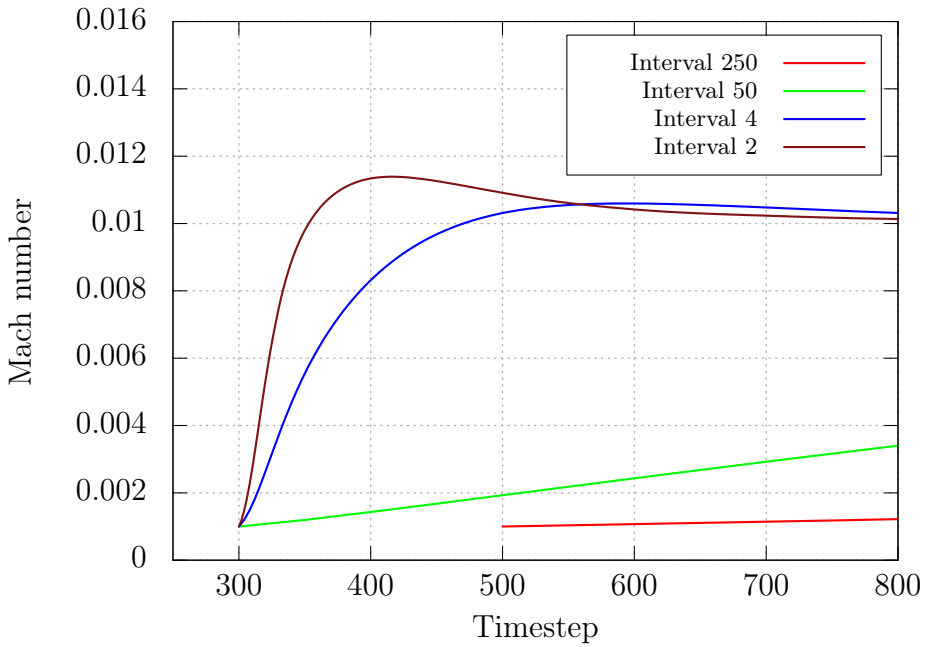


Figure 7.2: Influence of different adaption intervals on the Mach number development at the start of the VAM algorithm at time step 300.

Especially the intervals of 50 and 250 exhibit torque changes for which the amplitude is approximately halved, compared to the interval of 2. At the example of the torque development for an adaption interval of 4 time steps, one can conclude, that the two peaks of torque change, for the intervals 2 and 4, are composed of smaller changes that accumulate. As one can see, smaller adaption intervals converge faster than higher intervals. This corresponds to Figure 7.2, where the Mach number is plotted against the number of time steps. In accordance with the convergence behavior of the torque, the Mach number converges faster for small adaption intervals. It can be seen that the Mach number development, for the interval of 2 time steps, even overshoots the target Mach number. The convergence rate of higher adaption intervals are low. The first Mach number change for the interval of 250 time steps occurs after 500 time steps. Further development of the Mach number is depicted in Figure 7.3.

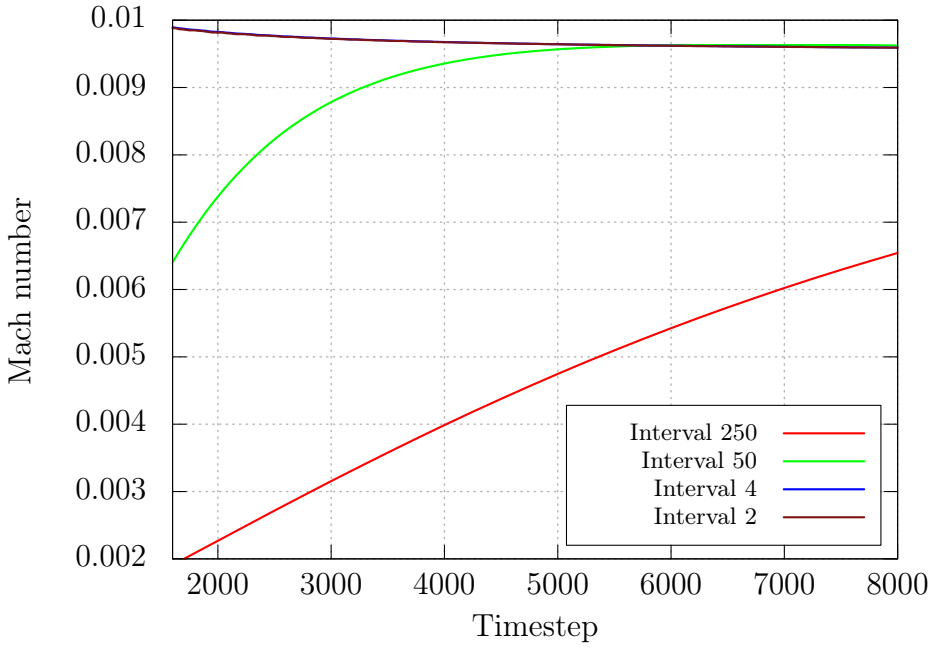


Figure 7.3: Influence of different adaption intervals on the Mach number development for an advancing simulation.

On can see that the evolution of the Mach number, with an adaption interval of 50, matches those of the intervals of 2 and 4 after about 5500 time steps. The interval of 250 posses the lowest convergence rate, but does converge to the same target Mach number, although this is not explicitly shown in this picture. The corresponding torque development for simulations in an advanced state is shown in Figure 7.4. Consistent with the evolution of the Mach number, the torque converges for all adaption values. The peaks in torque,

which stem from the Mach number adaption, already decayed for the intervals 2 and 4, and after 5500 time steps also for the adpation interval of 50. For the interval of 250, the torque development does also match the value of the remaining intervals and the peaks in torque decay, as the Mach number converges. This is also not explicitly shown in this picture.

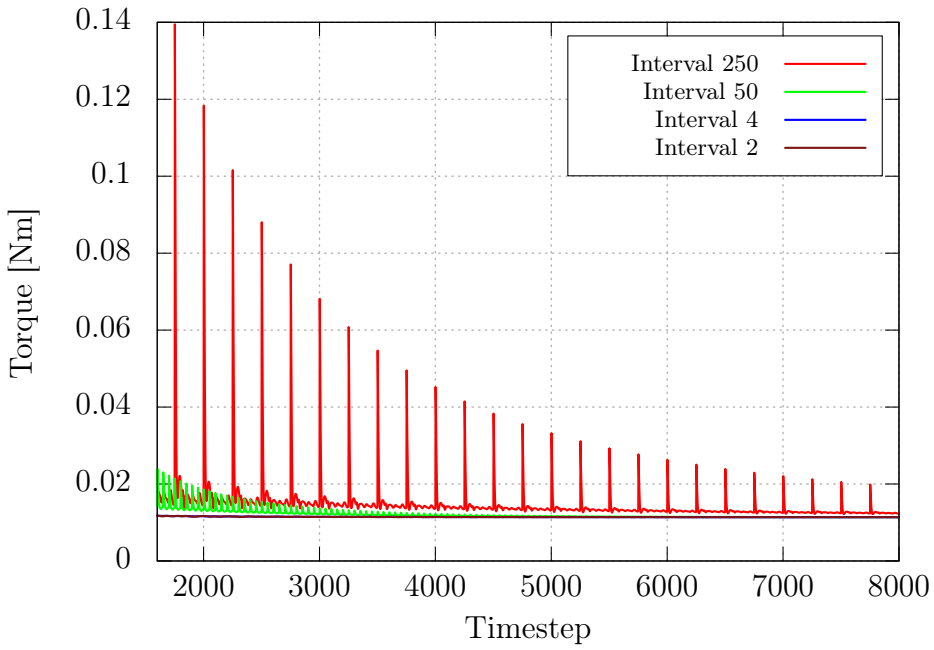


Figure 7.4: Influence of different adaption intervals on the torque development for an advancing simulation.

It can be summarized, that small adaption intervals accumulate to relatively high peaks in torque change. The advantage of a small adaption interval is, that the system is rapidly driven to a physically consistent state, i.e., the simulation Mach number converges rapidly to a value, where the simulated viscosity range fits best to physically occurring shear rates. The drawback of a small adaption interval is, that, due to high changes of the simulation values, instabilities may be triggered. This is more likely for simulations with grid refinement, due to interpolations at the grid interface.

7.2 Influence of the Relaxation Parameter λ

Now, the influence of the relaxation parameter λ is examined. The relaxation parameter λ, introduced in eq. (5.5), is used to control the change in Mach number to stabilize the algorithm. The intention is, to be able to limit the Mach number change when steep gradients are present, e.g., in start-up situations, or to avoid oscillations of the system. Thus, it is an under-relaxation parameter for which $0 < \lambda \leq 1.0$. If there are no convergence problems, the parameter can safely be set to $\lambda = 1.0$. This investigation includes relaxation parameters from 0.5 to $5 \cdot 10^{-4}$. The adaption interval is set to a medium value of 10 time steps and the Mach number threshold is set to $\delta = 10^{-4}$. In Figure 7.5 the torque is plotted against the number of time steps for the four different relaxation parameters used. Comparable to the investigation of the adaption interval, here, high values of λ induce high peaks in the torque development and converge rapidly. The values $\lambda = 5 \cdot 10^{-2}$ and $\lambda = 5 \cdot 10^{-3}$ exhibit comparable changes in the torque during adaption, although the peaks of the latter are smaller especially right at the start of the VAM. One can clearly see, that the relaxation parameter controls their convergence rate.

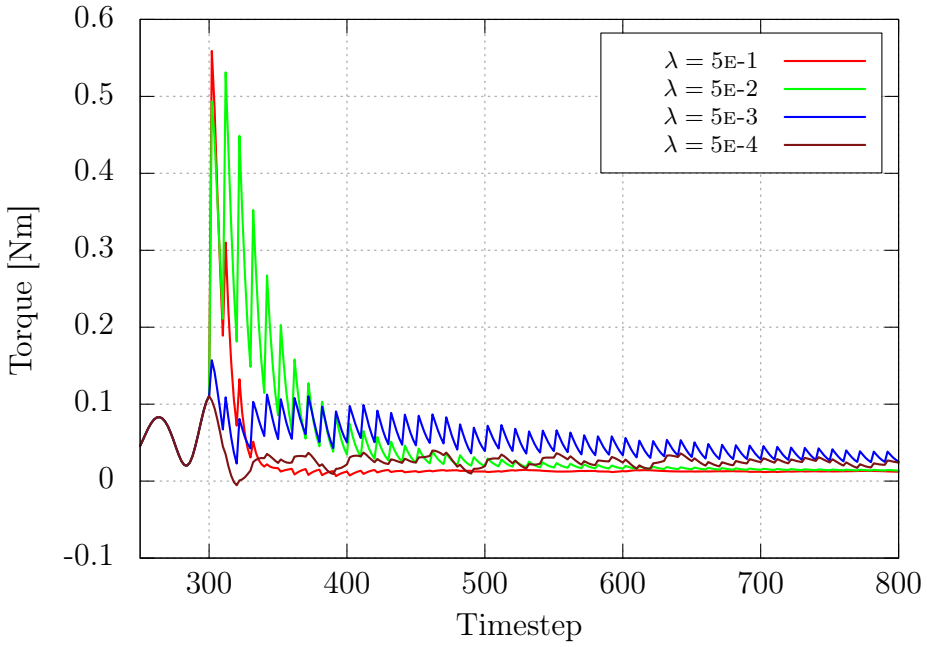


Figure 7.5: Influence of different relaxation parameters on the torque development at the start of the VAM algorithm at time step 300.

For the value of $\lambda = 5 \cdot 10^{-4}$ the change in torque is very small, because the change in Mach number is heavily damped. Figure 7.6 illustrates the corresponding evolution of the

Mach number in the first time steps after the VAM was started. It is a direct consequence of the definition of λ, that high values of the relaxation parameter lead to greater changes in the Mach number and therefore in a faster convergence to the target Mach number. For the relaxation parameter of 0.5, there is an overshoot in the Mach number development. Thus, the convergence rate of $\lambda = 0.5$ and $\lambda = 0.05$ are comparable.

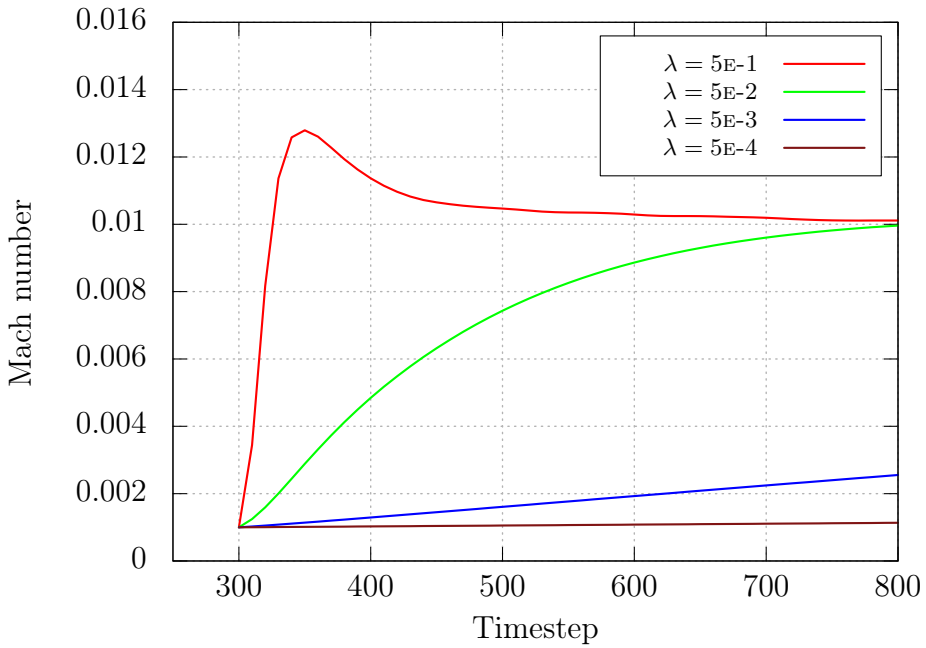


Figure 7.6: Influence of different relaxation parameters on the Mach number development at the start of the VAM algorithm at time step 300.

The remaining convergence rates are disproportionately low, especially the one for $\lambda = 5 \cdot 10^{-4}$. This is verified in the advancing Mach number evolution, shown in Figure 7.7. The Mach numbers for $\lambda = 0.5$ and $\lambda = 0.05$ already coincide. After about 7600 time steps, the Mach number value for a relaxation parameter of $5 \cdot 10^{-3}$ does also match the Mach number of the former parameters. Although not depicted, the simulation with $\lambda = 5 \cdot 10^{-4}$ does also converge to the target Mach number, but with a very small convergence rate. As already stated, the VAM is considered to be converged, if the Mach number change is smaller than the Mach number threshold δ. This is shown for $\lambda = 5 \cdot 10^{-3}$ in Figure 7.7 and is also verified in the corresponding torque development shown in Figure 7.8. After about 7600 time steps, the viscosity adaption method is abandoned and the torque development becomes smooth.

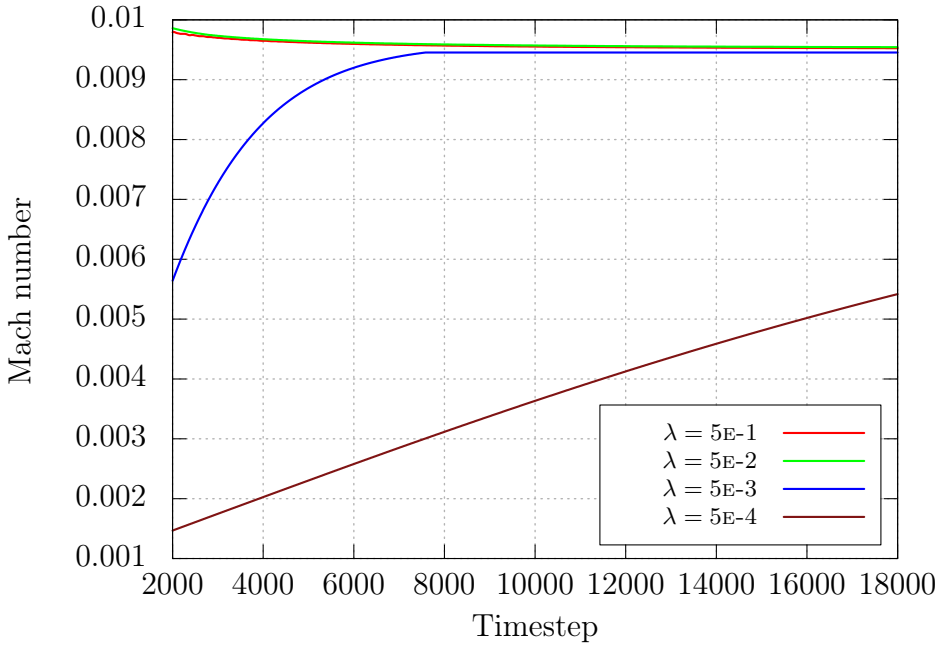


Figure 7.7: Influence of different relaxation parameters on the Mach number development for an advancing simulation.

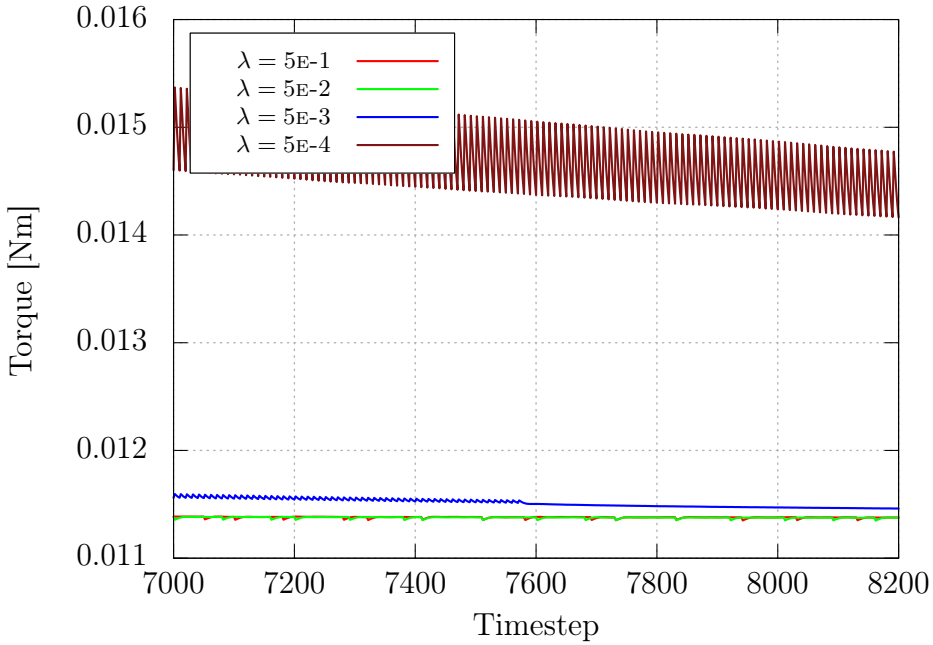


Figure 7.8: Influence of different relaxation parameters on the torque development for an advancing simulation.

The VAM is still active for the simulation with $\lambda = 5 \cdot 10^{-4}$. Since the Mach number change is still relatively high, but gets damped by the relaxation parameter, there are still pronounced peaks visible. These will decay, as the mach number converges in the progressing simulation. One can conclude, small values of λ lead to small peaks in the torque development and therefore may increase stability. By contrast, higher values of λ lead to higher convergence rates. Moderate values are preferable, due to unnecessary overshoots of the Mach number change.

7.3 Influence of the Mach Number Threshold δ

Finally, the impact of the Mach number threshold δ is investigated. It was introduced in eq. (5.6). Its purpose is to reduce the amount of computational effort, since very small changes in Mach number do not significantly improve the result. Thus, the threshold δ is introduced to be able to abandon the adaption process and save computational effort. The question arises, what can be considered to be a good trade off between accuracy and computational work. For this investigation, three different values for the threshold are chosen, ranging from $\delta = 0.01$, that is, one percent in relative Mach number change, to $\delta = 10^{-4}$. Again, the remaining parameters are kept constant and are set to $\lambda = 0.04$ with a medium value for the adaption interval of 10 time steps. The influence of the Mach number threshold on the torque evolution is shown in Figure 7.9, where the torque is plotted against the number of time steps.

One can see, that the adaption process is stopped after different amounts of time steps. After about 700 time steps, the torque development for $\delta = 10^{-2}$ becomes smooth. The same can be stated for the torque evolution with a Mach number threshold of 10^{-3} after about 1000 time steps. At time step 1200 also the simulation with $\delta = 10^{-4}$ converges, but the VAM becomes activated again after about time step 1300. More information about the convergence behavior of the VAM for the different thresholds can be taken from Figure 7.10. Indeed, the VAM converges prematurely after about 700 time steps for the simulation with a relatively high Mach number threshold of 1%.

As it was seen from the torque development for $\delta = 10^{-4}$, the Mach number is constant from about time step 1200 to 1300. Thereafter, the flow seems to change, such that the Mach number is further adjusted by the VAM. By contrast, this flow induced change of shear rates is not high enough to exceed the threshold of $\delta = 10^{-3}$ for which the Mach number is constant from about time step 1000 on. Further evolution of the Mach number is illustrated in Figure 7.11. One can see, that the Mach number for the simulation with medium threshold is sporadically adapted. Mach number adaption is deferred, until the threshold is exceeded and the simulation Mach number changes.

The simulation with $\delta = 10^{-4}$ converges to a value of about $Ma = 0.0096$ and evolves below the Mach number development with medium threshold. Since the simulation with the smallest Mach number threshold is supposed to be the most accurate, relative errors

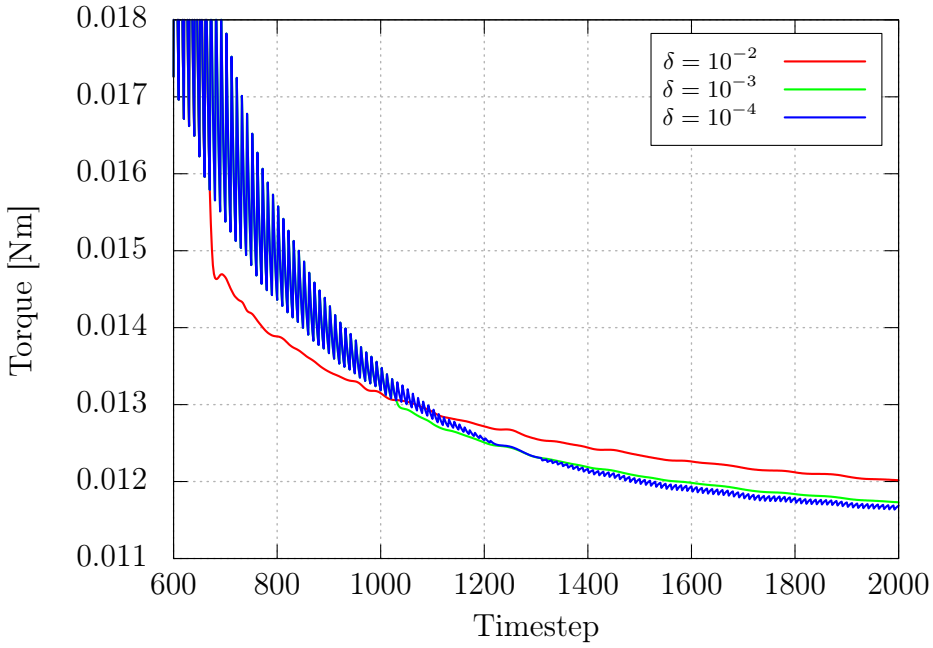


Figure 7.9: Influence of different Mach number thresholds on the torque development after the start of the VAM algorithm at time step 300.

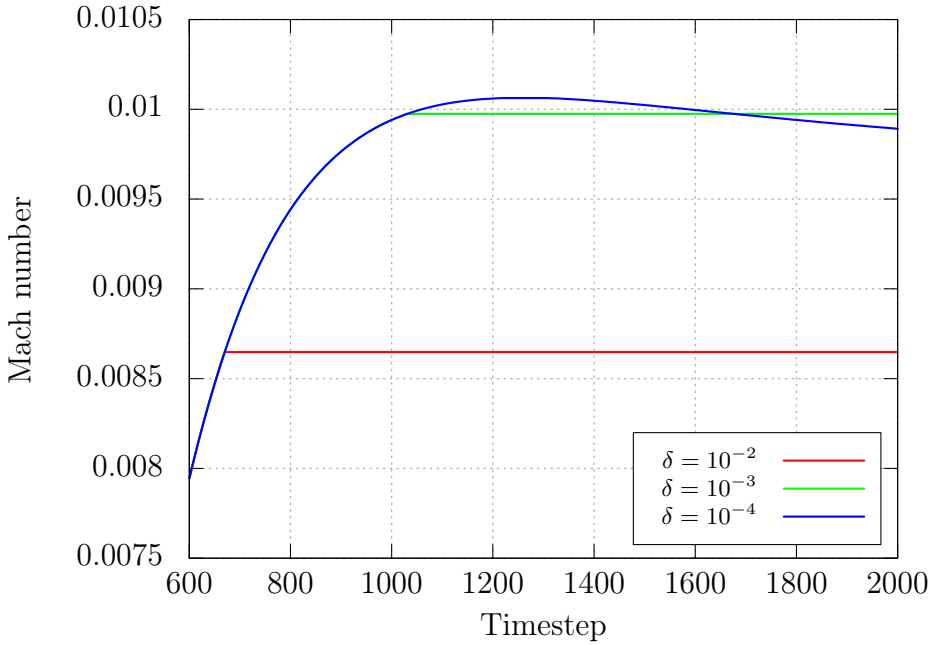


Figure 7.10: Influence of different Mach number thresholds on the Mach number development after the start of the VAM algorithm at time step 300.

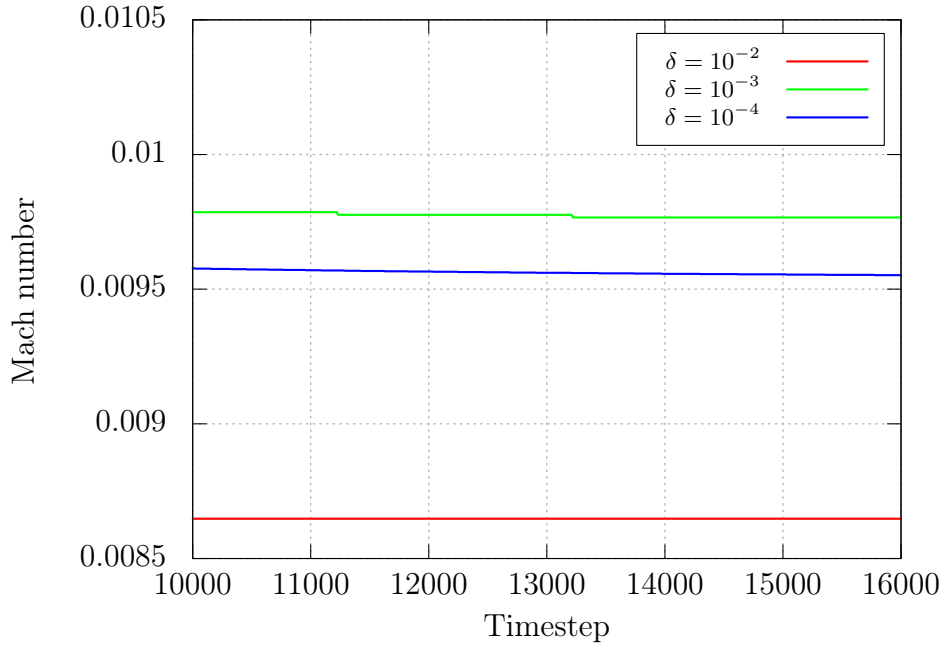


Figure 7.11: Influence of different Mach number thresholds on the Mach number development for an advancing simulation.

for the two remaining thresholds can be computed. Hence, the deviation of the converged Mach number for the simulation with $\delta = 10^{-2}$ amounts to about 10%. By contrast, the relative error in Mach number for $\delta = 10^{-3}$ is about 2%. The effect of these Mach number deviations on the converged values for the torque can be seen in the following Figure 7.12.

Again, the most accurate simulation result lies in between the simulations for $\delta = 10^{-2}$ and $\delta = 10^{-3}$. Accordingly, the deviation in torque from the simulation with $\delta = 10^{-4}$ amounts to $\approx 1.5\%$ and $\approx 0.4\%$, respectively.

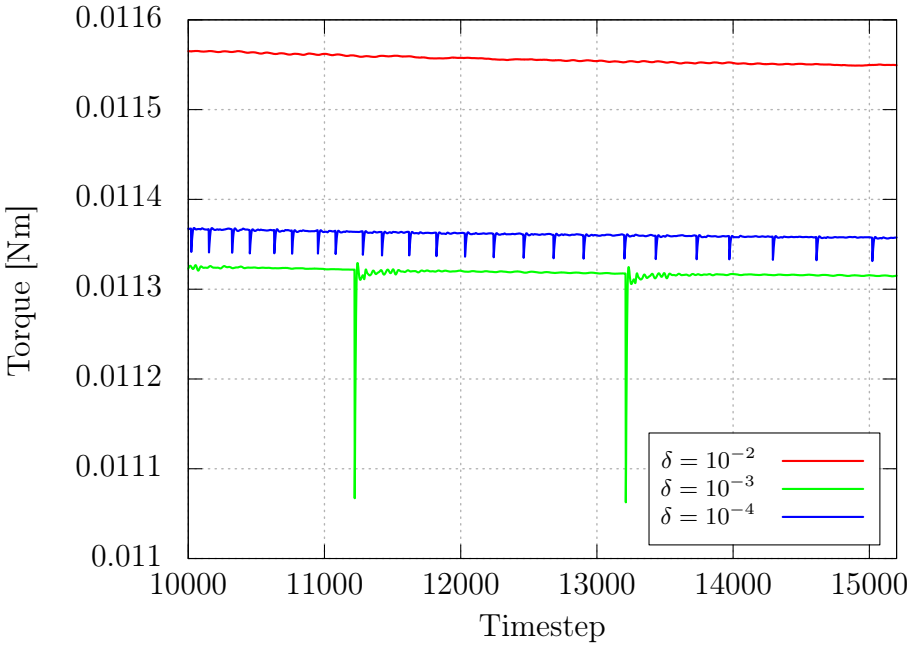


Figure 7.12: Influence of different Mach number thresholds on the torque development for an advancing simulation.

Concluding Remarks

A sensitivity analysis of the VAM to its convergence related input parameters δ, λ, and adaption interval was conducted in the previous sections. Thus, a review on the results of this parameter study should be given here.

It can be stated, that the relaxation parameter λ and the adaption interval only influence the stability and convergence rate of the simulation. Small relaxation parameters and low adaption frequencies result in slow converging VALBM simulation with enhanced stability properties. The parameter study of λ revealed, that a fast Mach number relaxation does not necessary lead to high convergence rates, due to overshoots in the Mach number development. If there are no convergence problems, the relaxation parameter should be set to a medium to high value of λ with an adaption interval of 2 time steps. A medium to small value of one or both can be used to stabilize VALBM simulations. The Mach number threshold is the only parameter with a measurable influence on the accuracy of the VALBM. If the value for δ is set too high, the VAM converges prematurely and this is an obstacle to the achievable simulation quality. In order to give the best results, this parameter should be chosen as $\delta = 10^{-4}$ or less. A medium value of $\delta = 10^{-3}$ seems to be a good trade off between accuracy and computational effort.

Each parameter was varied, while the remaining ones were kept constant. Interplay of the parameters could, thus, not be taken into account. Moreover, the simulations were performed for a constant starting Mach number of $Ma = 10^{-3}$ and the VAM was started after 300 time steps. Especially the influence of the starting point of the VAM algorithm is not analyzed, since it was shown in Section 6.2 and Section 6.1.1 that the converged Mach number is independent from the starting Mach number. Yet, the impact of the quality of the flow field initialization on the overall duration of the simulation is not studied in detail. From all the simulations conducted in this thesis, regarding the computational effort, it can be stated that a zero velocity initialization can be a better initial guess than a standard LBM simulation with a poor choice of the simulation Mach number. An educated guess of the simulation Mach number can speed up VALBM simulations.

7.4 Accuracy of Non-Newtonian Simulations

This section deals with the accuracy of non-Newtonian Lattice Boltzmann simulations. The basis of the simulations in this section is the generalized Newtonian Hagen-Poiseuille channel flow with the analytic solution, error definitions, material data, and simulation setup of Section 4.2.1. A goal of this accuracy analysis is to derive a recommendation for the best choice of the target collision frequency Ω_t, i.e., target simulation Mach number for the VALBM. In particular, the influence of the error contribution E_Ω, introduced in eq. (2.92) is investigated. The Lattice Boltzmann equation can be expressed in terms of the equilibrium distribution only [32]. Therefore, a series expansion is conducted through recursive application of the LBE to express the distribution function in terms of the equilibrium distributions from neighboring nodes at different times in the past. Recalling the LBE, we have,

$$\begin{aligned} f_i(\mathbf{x} + \mathbf{e}_i\Delta t, t + \Delta t) &= f_i(\mathbf{x}, t) + \Omega \cdot (f_i^{eq}(\mathbf{x}, t) - f_i(\mathbf{x}, t)) \\ &= (1 - \Omega) \cdot f_i(\mathbf{x}, t) + \Omega \cdot f_i^{eq}(\mathbf{x}, t). \end{aligned} \tag{7.1}$$

Now, substitution of the $f_i(\mathbf{x}, t)$ on the right hand side through the LBE for $\mathbf{x} - \mathbf{e}_i\Delta t$ and $t - \Delta t$ leads to

$$\begin{aligned} f_i(\mathbf{x} + \mathbf{e}_i\Delta t, t + \Delta t) = (1 - \Omega)^2 \cdot f_i(\mathbf{x} - \mathbf{e}_i\Delta t, t - \Delta t) + \Omega \cdot (1 - \Omega) \cdot f_i^{eq}(\mathbf{x} - \mathbf{e}_i\Delta t, t - \Delta t) \\ + \Omega \cdot f_i^{eq}(\mathbf{x}, t), \end{aligned} \tag{7.2}$$

which is the first order term in the series expansion. Considering the f^{eq} terms, one can derive a series formula

$$f_i(\mathbf{x} + \mathbf{e}_i\Delta t, t + \Delta t) = \Omega \sum_{k=1}^{\infty} (1 - \Omega)^{k-1} f_i^{eq}(\mathbf{x} - k\mathbf{e}_i\Delta t, t - k\Delta t) \tag{7.3}$$

with the number of neighbors k. This is the effective computational stencil that reduces to a nearest neighbor stencil for $k = 1$. The terms $(1 - \Omega)^{k-1}$ represent the weights of the contributions from the neighboring nodes. Thus, for $\Omega = 1$, the stencil only contains nearest neighbor distributions. For $\Omega < 1$ the weights have the same sign for all neighbors. They increase in value with increasing Ω and decrease with increasing k. The weights alternate in sign for even and odd neighbors k when $\Omega > 1$. Hence, it is possible to cancel error terms stemming from numerical diffusion. For all kinds of flows it is therefore wise to stay in the over-relaxation regime $\Omega > 1$ and this is why we choose $\Omega_{min} = 1.05$ in the simulations of the propeller viscosimeter. While the optimum choice of Ω is problem dependent, the form of the weights and the polynominals derived in [32] suggest that a minimum of the error occurs in the vicinity of $\Omega = 1$. According to [32], the choice of $\Omega \neq 1$ introduces additional artificial viscosity that has influence on the viscous and error terms. A more rigorous analysis of the impact of the finite centered difference stencil on the macroscopic equations may be found in [33, 34].

As already observed in Section 4.2.1 and throughout the literature,e.g., [6, 22], shear thinning non-Newtonian power-law fluids with $n < 1$ exhibit higher error values for the Hagen-Poiseuille channel flow than fluids with $n > 1$ on the same grid. Thus, it is an open question if the aforementioned numerical diffusion errors are responsible for this phenomenon, since the collision frequency changes within a non-Newtonian simulation due to strain rate dependent effective viscosities. Numerical diffusion errors are expected to have a minimum in the vicinity $\Omega = 1$, that is, if the distribution is close to equilibrium. We believe the following definition of E_Ω is a good measure to describe the influence of the error stemming from the Ω deviation in a well-defined simulation,

$$E_\Omega = \frac{1}{N} \cdot \sum_{\mathbf{x}} \left| \Omega \cdot (1 - \Omega) \right| . \tag{7.4}$$

The view on this term can be twofold. At first, it should be interpreted as a weighted sum of the collision frequencies in the domain with N lattice sites, where the weights $(1 - \Omega)$ consider the departure from equilibrium. On the other hand, it is the weight of the first next-to-neighbor contributions from the series expression eq. (7.3) of the effective computational stencil. In order to minimize the global absolute error in, e.g., the velocity field, the error E_Ω should be minimized. For non-Newtonian simulations the collision frequency is reformulated to incorporate the Mach number. It reads

$$\Omega = \frac{\dfrac{U \cdot \Delta x}{\sqrt{3} \cdot Ma}}{\left(\frac{k}{\rho} \left| \boldsymbol{\epsilon} \right|^{n-1} + 0.5 \dfrac{U \cdot \Delta x}{\sqrt{3} \cdot Ma} \right)}, \tag{7.5}$$

where the kinematic viscosity depends on the shear rate tensor through the constitutive law used, e.g., the power-law. Hence, Ω also depends implicitly on space and time via

the values $\mathbf{u}$ and ρ in the computation of the strain rate tensor eq. (2.110) and ν. An expression for the error source E_Ω is obtained analytically through

$$E_\Omega(Ma) = \int_x |\Omega \cdot (1 - \Omega)| \, dx, \tag{7.6}$$

where the integral is computed over the whole domain. The evolution of the strain rate can be computed through the derivative of the velocity distribution eq. (4.6). Since the expression eq. (7.6) still depends on the simulation Mach number, a possible best choice of Ma may be obtained by minimizing the functional $E_\Omega(Ma)$, which gives

$$Ma^{opt} = \min_{Ma} \left\{ \int_x |\Omega \cdot (1 - \Omega)| \, dx \right\}. \tag{7.7}$$

The analytic solution for the optimal Mach number Ma^{opt} is computed using the commercial software *Mathematica*.

7.4.1 SRT

In order to determine the influence of the simulation Mach number on the absolute global error in velocity, a series of simulations with Mach numbers $Ma = 0.01 \cdot x$, $\forall x \in [1, 20]$ and additionally the upper limit $Ma = 0.3$ are conducted for different values of the flow behavior index n. For each non-Newtonian fluid, four successively refined lattices are investigated, where the scaling $\Delta t \propto \Delta x^2$ is employed. Thus, the largest value for the Mach number on the finest grid is $Ma = 0.0375$ corresponding to $Ma = 0.3$ on the coarsest grid. The characteristic velocity is set to $U = 1$ throughout this investigation. We are interested in the evolution of the global absolute error in velocity. Since the Hagen-Poiseuille flow is stationary, the simulations are run until eq. (4.5) is fulfilled. For the sake of structuring, all simulation results are presented for the coarsest grid. The evolution of the global absolute error remains the same for all grids, albeit the value of the error itself changes. Hence, conclusion drawn are valid for all lattice resolutions. A grid convergence study is attached, where the convergence rates are investigated for the different Mach numbers. In a first run, we simulate a Newtonian fluid with $n = 1.0$. Figure 7.13 shows a plot of the global absolute error in velocity against the simulation Mach number. In the Newtonian case, we are able to additionally plot against the collision parameter Ω, which is applied on the top abscissa of the figure. The evolution of the error clearly shows a minimum at $Ma \approx 0.13$, which is about one order more accurate than the majority of the simulation results, e.g., $Ma = 0.05$. In terms of the collision frequency, the error minimum is about $\Omega \approx 0.95$. This confirms the statements made above, that a minimum of E_Ω is expected in the vicinity of $\Omega = 1$. However, the global absolute error is always an interplay of different error sources and especially the compressibility error shows its influence for increasing Mach numbers. If the Mach number is increased past the minimum value, the error rises

drastically to its maximum for $Ma = 0.3$. This corresponds to collision frequencies smaller one. By contrast, if the Mach number is decreased beyond the minimum value, the error increases again, but remains approximately constant afterwards. This corresponds to values of $\Omega > 1$.

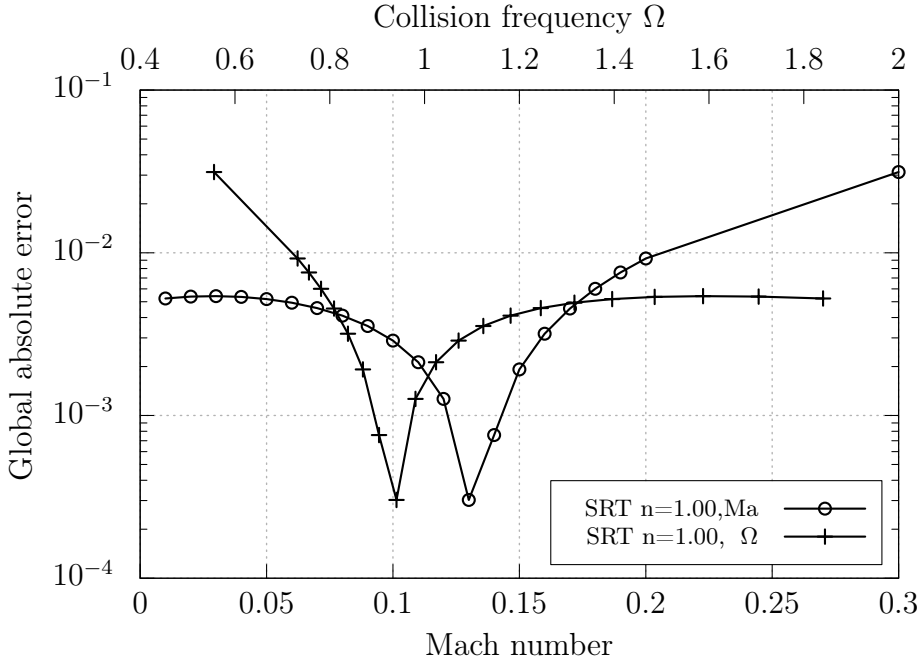


Figure 7.13: Evolution of the global absolute error in velocity for a Newtonian fluid with different values for Mach and Ω.

Since the dimensionless collision frequency varies in non-Newtonian simulations, due to varying local viscosity, it is no longer possible to quantify the simulation through this parameter. Thus, the simulation Mach number is the decisive quantity. In Figure 7.14, the evolution of the error is plotted against the simulation Mach number for different non-Newtonian fluids.

One can see, that there is a minimum in the developing of each non-Newtonian fluid. Comparable to the Newtonian simulations, the error rises for Mach numbers greater than the Mach number for which the error has a minimum. This behavior seems to be more sharp for shear thickening fluids with $n > 1$. By contrast, the developing of the error values for the simulation with $n = 0.6$ is more smooth and the minimum is not very prominent. However, the results show that simulations with small behavior index exhibit higher error values, albeit the errors for $n = 1.25$ and $n = 1.4$ are higher than that of $n = 0.75$ for Mach numbers greater than 0.15. Therefore, the simulations in the literature, e.g., in [6], are conducted in a Mach number regime corresponding to $Ma < 0.1$ in Figure 7.14.

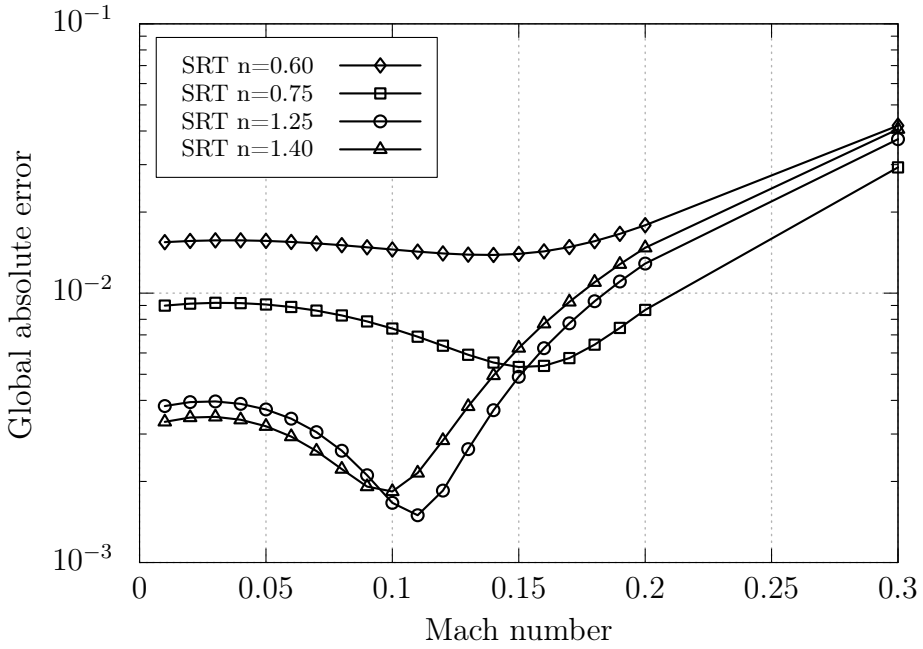


Figure 7.14: Evolution of the global absolute error in velocity for different non-Newtonian fluids with different values for Mach.

Optimum Choice of the Simulation Mach Number

In order to obtain a possible best choice for the simulation Mach number, the functional $E_\Omega(Ma)$ must be minimized. Here, the integral is taken over the channel height

$$Ma^{opt} = \min_{Ma} \left\{ \int_{-h}^{h} |\Omega \cdot (1 - \Omega)| \, dy \right\}, \tag{7.8}$$

for the prediction of the optimal simulation Mach number. The results for the different flow behavior indices and different lattice resolutions is summarized in Table 7.1. It should be noted, that the value for the optimal Mach number for the non-Newtonian fluids does not exactly satisfy the diffusive scaling.

Figure 7.15 shows the results for simulations conducted with the predicted optimum choice of Ma^{opt} on the coarsest grid. It seems that the error E_Ω is the dominant error source in this Mach number regime. Thus, the minimization of this error leads to a minimization of the overall global absolute error in the velocity field. The prediction is excellent for the fluids with $n = 0.75$, $n = 1.25$, and $n = 1.4$. However, for the simulation with $n = 0.6$, the optimal Mach number does not correspond to the minimum of the absolute global error. The error E_Ω may be minimized, but the compressibility error seems to dominate in this regime, since the Mach number is beyond 0.2. This can be regarded

Table 7.1: Computed optimal values Ma^{opt} for the simulation Mach number for different non-Newtonian fluids.

Resolution	Non-Newtonian fluid			
	$Ma^{opt}, n = 0.60$	$Ma^{opt}, n = 0.75$	$Ma^{opt}, n = 1.25$	$Ma^{opt}, n = 1.40$
10	0.2162010	0.1560420	0.1055030	0.103183
20	0.1120670	0.0766983	0.0521843	0.0507991
40	0.0570106	0.0386826	0.0259569	0.0252102
80	0.0287469	0.0194236	0.0129454	0.0125578

the reason why there is no pronounced minimum. The results indicate, that it is possible to determine a best choice of the simulation Mach number for flows where numerical diffusion errors, due to adverse distribution of the collision frequency, are the dominant sources of error.

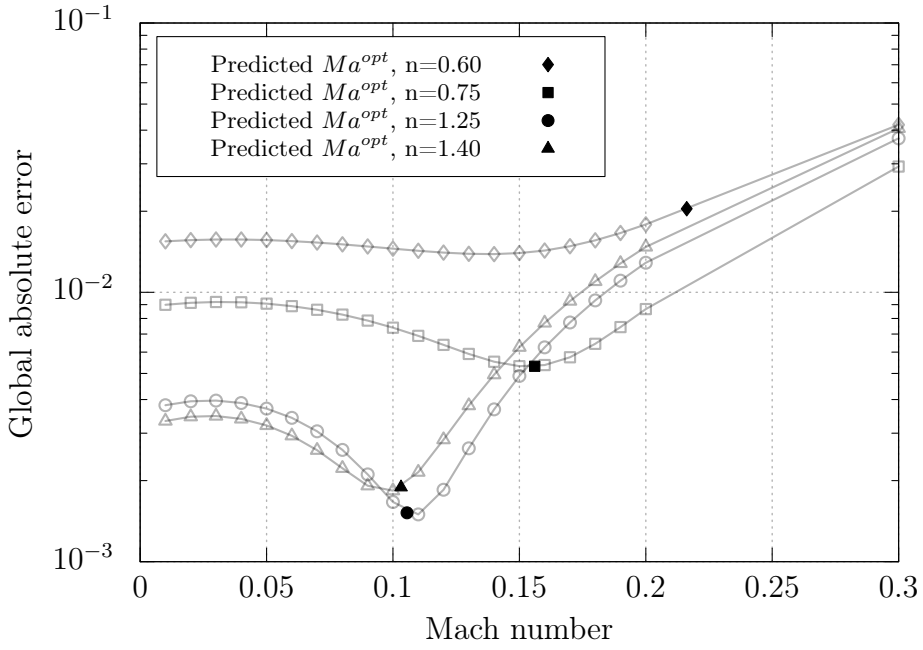


Figure 7.15: Prediction of the E_Ω minimum at Ma^{opt} for different non-Newtonian fluids with different values for Mach.

The minimum of the error E_Ω may not necessarily coincide with the minimum of the global absolute error as in the case of $n = 0.6$, when other sources of error, e.g, the

compressibility error dominate. The optimal Mach number is obtained by solving the minimization problem eq. (7.7). This was done here analytically, since the shear rate distribution is known. Therefore, this minimization problem may be incorporated into the VALBM and used to determine the target simulation Mach number.

Convergence study

Now, the convergence behavior for the investigated fluids and Mach numbers is studied. Four successively refined lattices are used. For the sake of lucidity, the results of the convergence study is exemplary presented in Figure 7.16 for the simulations with $n = 1.4$. Depicted is the error distribution for simulations with different Mach numbers, according to Figure 7.14, on successively refined lattices. This results in a 3D surface for the error distribution. One can see, there is a minimum in the global absolute error for the velocity field on every lattice resolution, similar to the previous investigation results shown in Figure 7.14 for the coarsest grid. The minimum shifts to smaller values of Mach with decreasing lattice spacing. This is a consequence of the diffusive scaling, that is, the consistent reduction of space-time and compressibility error. Since the data is plotted in a triple-log manner, the black line with slope -2 indicates second order convergence. All remaining non-Newtonian fluids show similar error distributions. The 3D error surface can be used to compute the convergence rate for each Mach number. This is done exemplary for the fluids $n = 0.75$ and $n = 1.25$. The results are shown in Figure 7.17.

It is shown that the simulations for $n = 1.25$ exhibits second order convergence behavior for Mach numbers smaller than approximately $Ma = 0.08$. The convergence rate breaks down for higher Mach numbers and has a minimum around $Ma = 0.12$ for which the convergence rate is approximately 1.90. it rises again with increasing Mach numbers, but only to a convergence rate of 1.98 for a Mach number of $Ma = 0.3$. Simulations for the fluid $n = 0.75$ show similar developing of the convergence rates. The simulations for $n = 0.75$ do nowhere exhibit second order convergence behavior in the mathematical sense of $\mathcal{O}(\Delta x^2)$, but closely for Mach numbers smaller than 0.05. Table 7.1 gives the Mach number for these two fluids for which the error has a minimum. It can be stated, that the convergence rate breaks down in the vicinity of Ma^{opt}.

The results indicate, that the choice of a Mach number that gives more accurate results, is accompanied with a loss of convergence rate. However, the results for Ma^{opt} are still one order more accurate than the majority of the simulation results with different Mach numbers even on the finest grid investigated.

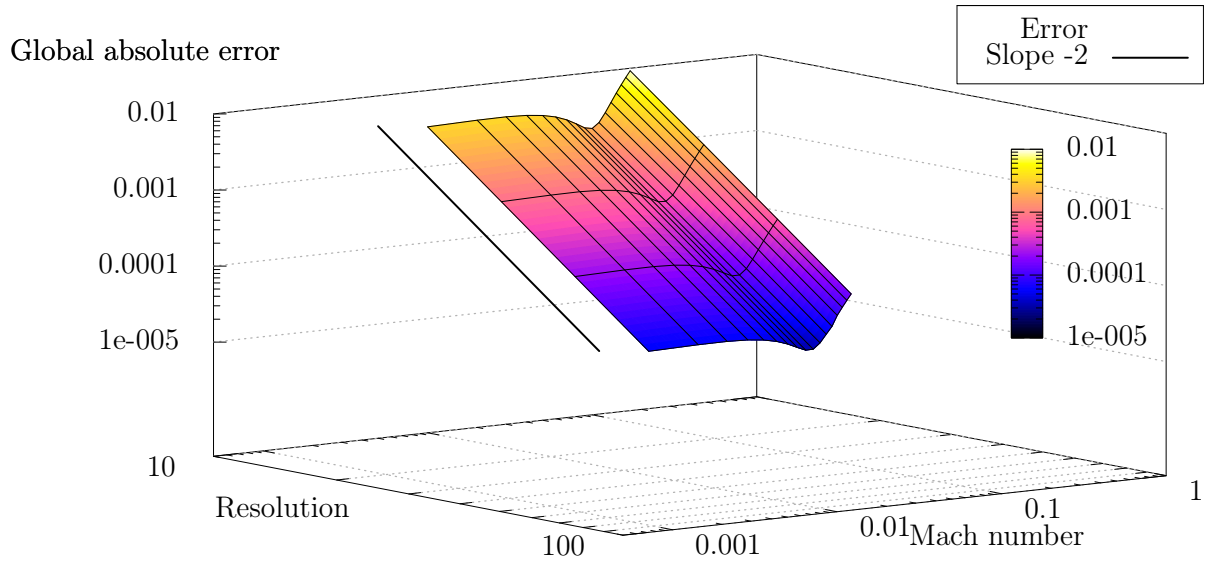


Figure 7.16: Error distribution of non-Newtonian simulations with $n = 1.4$ on successively refined lattices and different values for Mach.

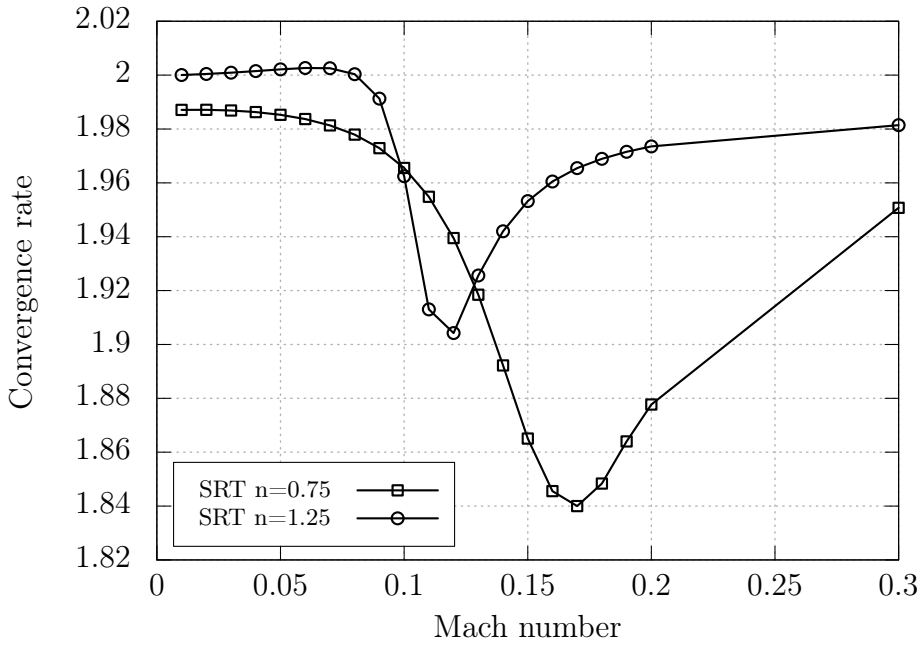


Figure 7.17: Convergence rate for non-Newtonian simulations with $n = 0.75$ and $n = 1.25$ with different values for Mach in the diffusive scaling.

7.4.2 MRT

The simulations done in the SRT case are rerun for the MRT collision operator. Again, the results are shown exemplary for the simulations $n = 0.75$ and $n = 1.25$, since the remaining non-Newtonian simulation show similar results. The advantage of the MRT model is the ability to use the free collision parameters s_{a-e} to tune the simulation. Hence, these parameters are varied. In order to limit the possibilities, the collision rates s_{a-e} are varied uniformly. The results of the non-Newtonian MRT simulations with $n = 0.75$ are shown in Figure 7.18. Plotted is the global absolute error in velocity against the Mach number for selected collision rates s_{a-e}.

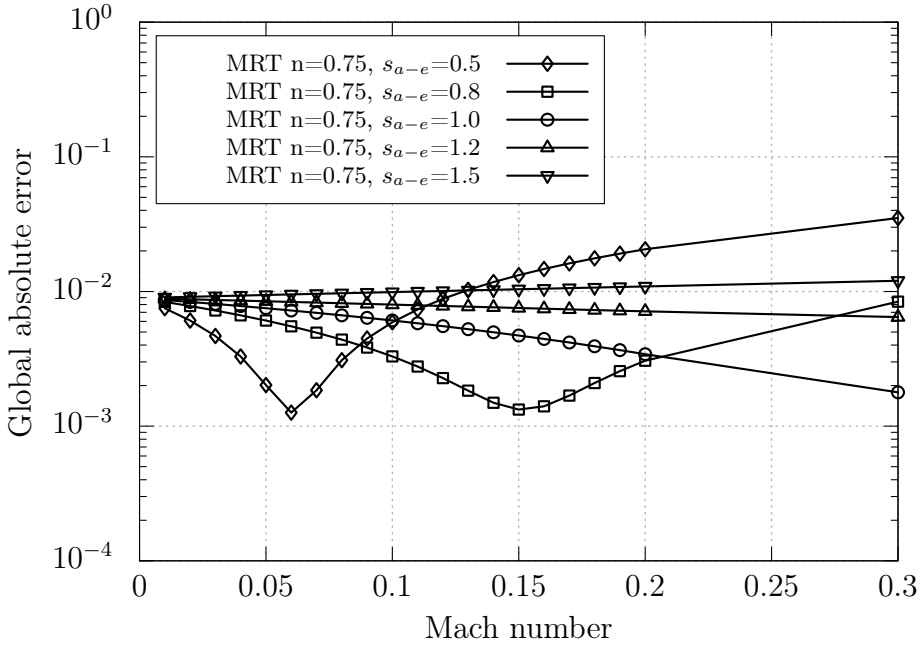


Figure 7.18: Evolution of the global absolute error in velocity for different values for Mach and different values for s_{a-e}.

The results clearly show, that the free collision parameters have major impact on the error distribution. For s_{a-e} smaller than one, the evolution of the error has a distinct minimum for which an accuracy of order 10^{-3} is achieved. Simulations with $s_{a-e} = 1.0$ can be considered a reasonable choice, since all non-physical moments are set to their equilibrium value during collision. The error distribution for this setting shows a declining trend for increasing Mach numbers. With increasing values for s_{a-e}, the error remains approximately constant for all simulation Mach numbers investigated. Figure 7.19 illustrates the simulation results for $n = 1.25$.

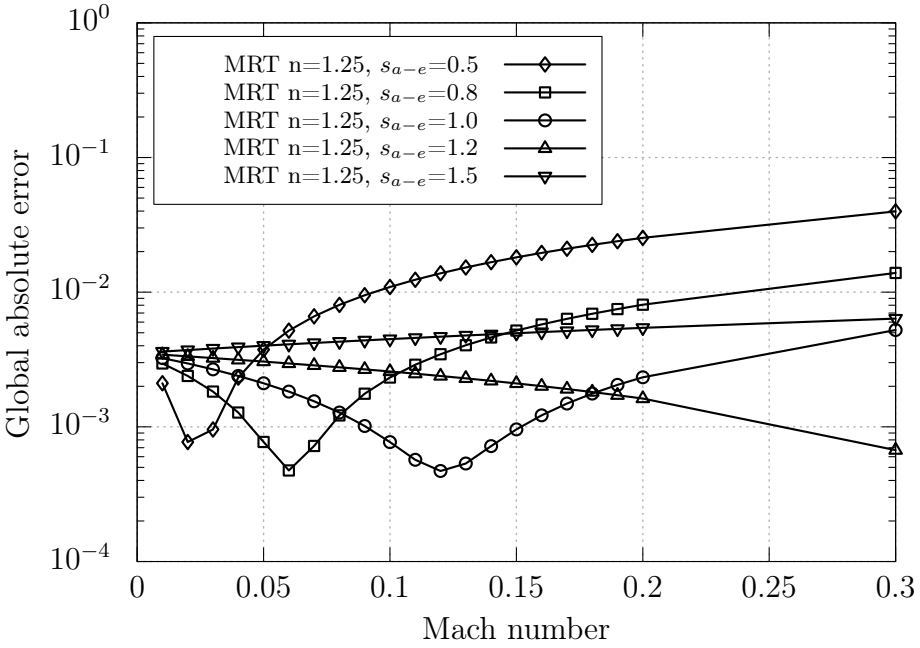


Figure 7.19: Evolution of the global absolute error in velocity for different values for Mach and different values for s_{a-e}.

There is also a strong dependence of the global absolute error on the free collision rates visible. With decreasing values of the collision parameters s_{a-e}, the error minimum is shifted towards smaller Mach numbers. Yet, the actual values for the global absolute error rises again for $s_{a-e} = 0.5$. For the investigated collision rates, a minimal error of order $5 \cdot 10^{-4}$ could be obtained, which is more accurate than the SRT pendant for which a minimal error of order $1 \cdot 10^{-3}$ could be achieved. In the case of $s_{a-e} = 1.2$, the error is declining towards the Mach number limit of $Ma = 0.3$. For the value of $s_{a-e} = 1.5$ the error increases slightly with increasing Mach numbers, but essentially stays in the region of $5 \cdot 10^{-2}$.

Concluding Remarks

The accuracy of the Lattice Boltzmann method for the simulation of non-Newtonian power-law fluids was investigated. To this end, simulation of non-Newtonian fluids with different flow behavior indices are conducted for different Mach numbers and differently resolved lattices. Both for the SRT as well as the MRT collision model. The results confirm, that the simulation Mach number has a major impact on the accuracy of the simulation. While it is clear that the overall error is an interplay of different error sources,

numerical diffusion error and compressibility errors seem to be the dominant sources of error. One goal of this investigation was to deduce a best choice for the simulation Mach number in order to incorporate these recommendations into the viscosity adaptive LBM. Therefore, the expression eq. (7.4) was derived, to quantify additional numerical diffusion errors E_Ω, introduced by the effective computational stencil. This was done analytically for the Hagen-Poiseuille flow by minimization of the functional $E_\Omega(Ma)$, which gives the optimal Mach number Ma^{opt}, eq. (7.7).

Excellent agreement of the results from the simulations with Ma^{opt} and the predicted minimum global error could be obtained in regions where the numerical diffusion error is the dominant source of error. The expressions derived for the prediction refer to the SRT model as can be seen from eq. (7.3), since the collision frequency is not expanded in a collision matrix. Indeed, the results clearly show, that the free relaxation parameters of the MRT model drastically influence the error in velocity and the value of the optimal Mach number for which that error is minimal. Generally, the MRT model shows a smaller value of the overall global absolute error for selected values of the free collision parameters. However, due to the influence of the parameters s_{a-e}, which can be chosen arbitrarily, eq. (7.7) is no longer suitable for the prediction of the optimal choice for the simulation Mach number in the MRT case.

CHAPTER 8

Conclusions and Perspective

8.1 Conclusions

This dissertation described the development, verification, and validation of a viscosity adaptive Lattice Boltzmann Method (VALBM) for the numerical simulation of non-Newtonian fluids, as well as the development and verification of the related software bundle *SAM-Lattice*.

Besides the introduction of the VALBM, a goal of this thesis is to offer assistance in the spirit of a theory guide to students and assistant researchers concerning the theory of the Lattice Boltzmann Method and its implementation in *SAM-Lattice*. In Chapter 2, the mathematical foundation of the LBM was given and the route from the BGK approximation of the Boltzmann equation to the Lattice Boltzmann (BGK) equation was delineated in detail. The derivation is restricted to isothermal flows only. Restrictions of the method, such as low Mach number flows were highlighted and the accuracy of the method was discussed.

In Chapter 3, some fundamental aspects on the implementation of the main components, including the corresponding flow charts were discussed. Actual details to the implementation are given in the comprehensive programmers guides to *SamGenerator* and *SamSolver*. In order to ensure the functionality of the implementation of *SamSolver*, the solver was verified in Chapter 4 for both Newtonian and non-Newtonian fluids, including a convergence study of the method and an investigation on higher order boundary conditions.

The essential novelty of the method, developed in the course of this thesis, is that the numerically modeled viscosity range is consistently adapted to the actual physically exhibited viscosity range through change of the simulation time step and the simulation

Mach number, respectively, while the simulation is running. The algorithm is robust, independent of the Mach number the simulation was started with, and applicable for stationary flows as well as transient flows. The theory of the viscosity adaption method was introduced in Chapter 5. Although the VAM was used here for fluids that can be modeled with the power-law approach, the implementation of the VALBM is straightforward for other non-Newtonian models, e.g., the Carreau-Yasuda or Cross model.

In Chapter 6 the VALBM was verified in the stationary case for the Hagen-Poiseuille channel flow and in the transient case for the Wormersley flow. Moreover, the VALBM was validated for the case of a propeller viscosimeter developed at the chair SAM. To this end, the experimental data of the torque on the impeller of 3 shear thinning non-Newtonian liquids served for the validation. The VALBM showed excellent agreement with experimental data for all of the investigated fluids and in every operating point. For reasons of comparison, a series of standard LBM simulations was carried out with different simulation Mach numbers, which partly showed errors of several hundred percent.

In Chapter 7 the sensitivity to the parameters used in the VAM was analyzed. The result of this study was that the relaxation parameter λ and the adaption interval only influence the convergence and stability of the simulation. If there are no stability problems, a medium to high value of λ and an interval of 2 are suggested for the VAM. By contrast, the Mach number threshold δ saves computational work but limits the possible accuracy. A value of $\delta = 10^{-3}$ seems to be a good trade off between accuracy and computational effort.

Finally, the accuracy of non-Newtonian Lattice Boltzmann simulations with SRT and the MRT model was analyzed in detail. For the non-Newtonian case, an error estimate in the form of a functional was derived on the basis of a series expansion of the Lattice Boltzmann equation. This functional can be solved analytically for the case of the Hagen-Poiseuille channel flow of non-Newtonian fluids. The estimation of the error minimum is excellent in regions where the Ω error is the dominant source of error as opposed to the compressibility error for the SRT collision operator.

8.2 Directions for Further Work

Result of this dissertation is a verified and validated software bundle on the basis of the viscosity adaptive Lattice Boltzmann Method. The work restricts itself on the simulation of isothermal, laminar flows with small Mach numbers. As further research goals concerning the VALBM, the following suggestions are made:

- **Turbulence**
  Since we only dealt with laminar flows, the capabilities of the Lattice Boltzmann Method and in particular of the VALBM for the simulation of turbulent non-Newtonian flows should be investigated. There is already a LES model with the standard Smagorinsky SGS model implemented.

- **Application of the Error Estimate**
  An error estimate for the error contribution E_Ω was derived for the SRT collision model. A possible target simulation Mach number for the VAM can be deduced by minimizing the Ω error estimate, while the simulation is running. This should be implemented in *SamSolver* and the VALBM should be tested with this objective.

- **Derivation for the MRT model**
  The series expansion of the Lattice Boltzmann equation for the derivation of the error estimate was done for the SRT model. The results in Chapter 7 have shown, that this form of the estimate is not suitable for the MRT model, because of the influence of the free collision parameters s_{a-e}. A corresponding derivation could be conducted for the MRT case.

- **Further Testing**
  The VALBM has been shown to provide excellent results for the simulation of real world experiments as in the case of the propeller viscosimeter. However, further testing is needed to explore the cababilities of this new method.

Generally, *SamGenerator* and *SamSolver* should be extended to higher lattices, such as the D3Q39 or D3Q121 velocity discretization, in order to incorporate heat transfer and deal with compressible fluids.

APPENDIX A

MRT matrix

$$
\begin{bmatrix}
(1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) & 1 \\
(-1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) & e^2 \\
(1 & -2 & -2 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) & e^4 \\
(0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0) & e \\
(0 & -2 & 2 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0) & e^3 \\
(0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) & e \\
(0 & 0 & 0 & -2 & 2 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) & e^3 \\
(0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1) & e \\
(0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1) & e^3 \\
(0 & 2 & 2 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2) & e^2 \\
(0 & -2 & -2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2) & e^4 \\
(0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0) & e^2 \\
(0 & 0 & 0 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0) & e^4 \\
(0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) & e^2 \\
(0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1) & e^2 \\
(0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0) & e^2 \\
(0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0) & e^3 \\
(0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) & e^3 \\
(0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1) & e^3
\end{bmatrix}
$$

Matrix **M** for the linear transformation from distribution space to moment space.

Nomenclature

D_i	Impeller diameter
E_Ω	Relaxation parameter error
E_g	Geometry discretization error
E_t	Time error
E_x	Spatial error
E_{BC}	Boundary condition error
E_{Ma}	Compressibility error
H	Height
Kn	Knudsen number
L	Characteristic length scale
Ma	Mach number
Ma^{**}	Final adapted Mach number
Ma^{*}	Intermediate adapted Mach number
N	Number of lattice sites
R	Specific gas constant
Re	Reynolds number
T	Characteristic time scale

T	Temperature
U	Characteristic flow speed
Δg	Geometry discretization step size
Δt	Time step size
Δx	Space step size
Ω	Dimensionless collision time
Ω_t	Target VAM dimensionless collision frequency
Ω_{max}	Maximum dimensionless collision frequency
Ω_{min}	Minimum dimensionless collision frequency
α	Womersley number
$\boldsymbol{\epsilon}$	Macroscopic strain rate tensor
$\boldsymbol{\sigma}$	Macroscopic stress tensor
$\boldsymbol{\xi}$	Absolute molecular velocity vector
$\boldsymbol{\xi}_i$	Discrete molecular velocity vector
κ	Isotropic exponent
λ	VAM Mach relaxation parameter
$\mathbf{M}$	Linear transformation matrix
$\mathbf{a}$	Acceleration
$\mathbf{e}_i$	Molecular velocity vector of the lattice
$\mathbf{f}^F$	External force vector
$\mathbf{m}^T$	Torque vector
$\mathbf{u}$	Macroscopic velocity
$\mathbf{u}_w$	Wall velocity
$\mathbf{x}$	Physical space vector
μ	Dynamic viscosity
μ'	Bulk viscosity

ν	Kinematic viscosity
ν_{max}	Maximum kinematic viscosity
ν_{min}	Minimum kinematic viscosity
ρ	Density
τ	Dimensional collision time
τ_w	Wall shear stress
ξ	Component of the molecular velocity vector $\boldsymbol{\xi}$
c_s	Speed of sound
$f(\mathbf{x}, \xi, t)$	density probability function
f^{eq}	Equilibrium probability distribution function
f^{neq}	Non-equilibrium part of the distribution f
k	Flow consistency index
m	Macroscospic moment of the probability distribution function
n	Agitator speed
n	Flow behavior index
p	Thermodynamic pressure
s_{a-e}	Relaxation parameters for non-hydrodynamic moments
$scale$	Scale factors of grid refinement
w	weighting factor
E	Error
R	Radius
t	Time

Bibliography

[1] T. Akenine-Möller. Fast 3D Triangle-Box Overlap Testing. In *ACM SIGGRAPH 2005 Courses*, page 8. ACM, 2005.

[2] ASME-V&V-20-Committee. *Standard for Verification and Validation in Computational Fluid Dynamics and Heat Transfer*. American Society of Mechanical Engineers, New York, November 2009.

[3] P. L. Bhatnagar, E. P. Gross, and M. Krook. A Model For Collison Processes in Gases. *Phys. Rev.*, 94:511, 1954.

[4] R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport Phenomena*. John Wiley & Sons, 2007.

[5] M. Bouzidi, M. Firdaouss, and P. Lallemand. Momentum Transfer of a Boltzmann-Lattice Fluid with Boundaries. *Physics of Fluids (1994-present)*, 13(11):3452–3459, 2001.

[6] J. Boyd, J. Buick, and S. Green. A Second-Order Accurate Lattice Boltzmann non-Newtonian Flow Model. *Journal of physics A: Mathematical and General*, 39(46):14241, 2006.

[7] R. Brownlee, A. N. Gorban, and J. Levesley. Stability and Stabilization of the Lattice Boltzmann Method. *Physical Review E*, 75(3):036711, 2007.

[8] R. Chandra. *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.

[9] D. Conrad, A. Schneider, and M. Böhle. Numerical Investigation of an Extended Propeller Viscosimeter by Means of Lattice Boltzmann Methods. In *ASME 2013 Fluids Engineering Division Summer Meeting*, pages V01AT03A015–V01AT03A015. American Society of Mechanical Engineers, 2013.

[10] B. Crouse. *Lattice-Boltzmann Strömungssimulationen auf Baumdatenstrukturen*. PhD thesis, TU Braunschweig, 2003.

[11] P. J. Dellar. Bulk and Shear Viscosities in Lattice Boltzmann Equations. *Physical Review E*, 64(3):031203, 2001.

[12] P. J. Dellar. Incompressible Limits of Lattice Boltzmann Equations using Multiple Relaxation Times. *Journal of Computational Physics*, 190(2):351–370, 2003.

[13] P. J. Dellar. An Interpretation and Derivation of the Lattice Boltzmann Method using Strang Splitting. *Computers & Mathematics with Applications*, 65(2):129–141, 2013.

[14] P. J. Dellar. Lattice Boltzmann Algorithms without Cubic Defects in Galilean Invariance on Standard Lattices. *Journal of Computational Physics*, 259:270–283, 2013.

[15] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo. Multiple–Relaxation–Time Lattice Boltzmann Models in Three Dimensions. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.

[16] F. R. Feito, C. J. Torres, and A. Ureña. Orientation, Simplicity and Inclusion Test for Planar Polygons. *Comput. & Graphics*, 19(4):595–600, 1995.

[17] F. R. Feito and J. C. Torres. Inclusion Test for General Polyhedra. *Comput. & Graphics*, 21(1):23–30, 1997.

[18] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*, volume 3. Springer Berlin, 1996.

[19] O. Filippova and D. Hänel. Boundary-Fitting and Local Grid Refinement for Lattice-BGK Models. *International Journal of Modern Physics C*, 9(08):1271–1279, 1998.

[20] O. Filippova and D. Hänel. Grid Refinement for Lattice-BGK Models. *Journal of Computational Physics*, 147(1):219–228, 1998.

[21] S. Freudiger. *Entwcklung eines Parallelen, Adaptiven, Komponentenbasieren Strömungskerns für Hierarische Gitter auf Basis der Lattice Boltzmann Methode.* PhD thesis, TU Braunschweig, 2009.

[22] S. Gabbanelli, G. Drazer, and J. Koplik. Lattice Boltzmann Method for non-Newtonian (Power-law) Fluids. *Phys. Rev. E*, 72:046312, Oct 2005.

[23] I. Ginzburg, F. Verhaeghe, and D. d'Humieres. Two-Relaxation-Time Lattice Boltzmann Scheme: About Parametrization, Velocity, Pressure and Mixed Boundary Conditions. *Communications in computational physics*, 3(2):427–478, 2008.

[24] Z. Guo and C. Shu. *Lattice Boltzmann Method and Its Application in Engineering.* World Scientific Pub Co, 2013.

[25] Z. Guo, C. Zheng, and S. Baochang. Discrete Lattice Effects on the Forcing Term in the Lattice Boltzmann Method. *Phys Rev E*, 65:046308, 2002.

[26] Z. Guo, C. Zheng, and B. Shi. Lattice Boltzmann Equation with Multiple Effective Relaxation Times for Gaseous Microscale Flow. *Physical Review E*, 77(3):036707, 2008.

[27] E. A. Haines and J. R. Wallace. Shaft Culling for Efficient Ray-Cast Radiosity. In *Photorealistic rendering in computer graphics*, pages 122–138. Springer, 1994.

[28] D. Hänel. *Molekulare Gasdynamik*. Springer, 2006.

[29] J. Harris. *Rheology and non-Newtonian Flow*. Longman London, 1977.

[30] X. He, S. Chen, and G. D. Doolen. A Novel Thermal Model for the Lattice Boltzmann Method in Incompressible Limit. *Journal of Computational Physics*, 146(1):282–300, 1998.

[31] S. Hoffmann and R. Lienhart. *OpenMP: Eine Einführung in die Parallele Programmierung mit C/C++*. Informatik im Fokus. Springer, 2009.

[32] D. J. Holdych, D. R. Noble, J. G. Georgiadis, and R. O. Buckius. Truncation Error Analysis of Lattice Boltzmann Methods. *Journal of Computational Physics*, 193(2):595–619, 2004.

[33] M. Junk. A Finite Difference Interpretation of the Lattice Boltzmann Method. *Numerical Methods for Partial Differential Equations*, 17(4):383–402, 2001.

[34] M. Junk, A. Klar, and L.-S. Luo. Asymptotic Analysis of the Lattice Boltzmann Equation. *Journal of Computational Physics*, 210(2):676–704, 2005.

[35] M. Krafczyk. *Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung*. Habilitationsschrift,Lehrstuhl für Bauinformatik, TU-München, 2001.

[36] T. Krüger, F. Varnik, and D. Raabe. Shear Stress in Lattice Boltzmann Simulations. *Physical Review E*, 79(4):046704, 2009.

[37] T. Krüger, F. Varnik, and D. Raabe. Second-Order Convergence of the Deviatoric Stress Tensor in the Standard Bhatnagar-Gross-Krook Lattice Boltzmann Method. *Physical Review E*, 82(2):025701, 2010.

[38] A. J. C. Ladd. Numerical Simulation of Particulate Suspensions via a Discrete Boltzmann Equation. Part 1. Theoretical Foundation. *J. Fluid. Mech.*, 271:285–309, 1994.

[39] A. J. C. Ladd and R. Verberg. Lattice-Boltzmann Simulations of Particle-Fluid Suspensions. *Journal of Statistical Physics*, 104(5-6):1191–1251, 2001.

[40] J. Latt. *Hydrodynamic Limit of Lattice Boltzmann Equations*. PhD thesis, Université de Genéve, 03/09 2007.

[41] C. Loudon and A. Tordesillas. The Use of the Dimensionless Womersley Number to Characterize the Unsteady Nature of Internal Flow. *Journal of theoretical biology*, 191(1):63–78, 1998.

[42] L.-S. Luo. *Lattice-Gas Automata and Lattice Boltzmann Equations for Two-Dimensional Hydrodynamics*. PhD thesis, Georgia Institute of Technology, Atlanta, 1993.

[43] O. Malaspinas. *Lattice Boltzmann Method for the Simulation of Viscoelastic Fluid Flows*. PhD thesis, École Polytechnique Fédérale De Lausanne, 2009.

[44] R. Mei, L.-S. Luo, P. Lallemand, and D. d'Humières. Consistent Initial Conditions for Lattice Boltzmann Simulations. *Computers & Fluids*, 35(8):855–862, 2006.

[45] R. Mei, L.-S. Luo, and W. Shyy. An Accurate Curved Boundary Treatment in the Lattice Boltzmann Method. *Journal of Computational Physics*, 155(2):307–330, 1999.

[46] R. Mei, W. Shyy, D. Yu, and L.-S. Luo. Lattice Boltzmann Method for 3-D Flows with Curved Boundary. *Journal of Computational Physics*, 161(2):680–699, 2000.

[47] R. Mei, D. Yu, W. Shyy, and L.-S. Luo. Force Evaluation in the Lattice Boltzmann Method Involving Curved Geometry. *Physical Review E*, 65(4):041203, 2002.

[48] R. Ouared and B. Chopard. Lattice Boltzmann Simulations of Blood Flow: Non-Newtonian Rheology and Clotting Processes. *Journal of Statistical Physics*, 121:209–221, Oct. 2005.

[49] D. Pritchard, C. R. McArdle, and S. K. Wilson. The Stokes Boundary Layer for a Power-law Fluid. *Journal of Non-Newtonian Fluid Mechanics*, 166(12):745–753, 2011.

[50] Y. Qian, D. d'Humières, and P. Lallemand. Lattice BGK Models for Navier-Stokes Equation. *EPL (Europhysics Letters)*, 17(6):479–484, 1992.

[51] Y. Qian and S. Orzag. Lattice BGK Models for the Navier-Stokes Equation – Nonlinear Deviation in the Compressible Regimes. *EPL (Europhysics Letters)*, 21(3):255–259, 1993.

[52] M. B. Reider and J. D. Sterling. Accuracy of Discrete-Velocity BGK Models for the Simulation of the Incompressible Navier-Stokes Equations. *Computers & fluids*, 24(4):459–467, 1995.

[53] T. Reviol. *Experimentelle und Numerische Untersuchungen eines Modifizierten Propeller-Viskosimeters zur Bestimmung der Fließeigenschaften nicht-Newtonscher Medien mit Inhomogenem Charakter*. PhD thesis, TU Kaiserslautern, 2010.

[54] D. Ricot, S. Marié, P. Sagaut, and C. Bailly. Lattice Boltzmann Method with Selective Viscosity Filter. *Journal of Computational Physics*, 228(12):4478–4490, 2009.

[55] H. Schlichting and K. Gersten. *Boundary-Layer Theory*. Springer, 2000.

[56] A. Schneider. *A Consistent Large Eddy Approach for Lattice Boltzmann Methods and its Application to Complex Flows*. PhD thesis, TU Kaiserslautern, 2015.

[57] X. Shan, X.-F. Yuan, and H. Chen. Kinetic Theory Representation of Hydrodynamics: A Way Beyond the Navier–Stokes Equation. *Journal of Fluid Mechanics*, 550:413–441, 2006.

[58] M. Stiebler, M. Krafczyk, S. Freudiger, and M. Geier. Lattice Boltzmann Large Eddy Simulation of Subcritical Flows Around a Sphere on non-Uniform Grids. *Computers & Mathematics with Applications*, 61(12):3475–3484, 2011.

[59] S. Succi. *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*. Oxford university press, 2001.

[60] F. Tosi, S. Ubertini, S. Succi, H. Chen, and I. V. Karlin. Numerical Stability of Entropic Versus Positivity-Enforcing Lattice Boltzmann Schemes. *Mathematics and Computers in Simulation*, 72(2):227–231, 2006.

[61] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Springer, 2000.

[62] Z. Yang. *Analysis of Lattice Boltzmann Boundary Conditions*. PhD thesis, Universität Konstanz, 2007.

[63] W.-A. Yong and L.-S. Luo. Nonexistence of H-Theorems for the Athermal Lattice Boltzmann Models with Polynomial Equilibria. *Physical Review E*, 67(5):051105, 2003.

[64] D. Yu, R. Mei, and W. Shyy. A Multi-Block Lattice Boltzmann Method for Viscous Fluid Flows. *International Journal for Numerical Methods in Fluids*, 39(2):99–120, 2002.

[65] R. Zhang, C. Sun, Y. Li, R. Satti, R. Shock, J. Hoch, and H. Chen. Lattice Boltzmann Approach for Local Reference Frames. *Communications in Computational Physics*, 9(5):1193, 2011.

Curriculum Vitae

Daniel Conrad

Education

Since 10/2009	PhD in Mechanical Engineering University of Kaiserslautern, Germany Department of Mechanical and Process Engineering - Institute of Fluid Mechanics and Turbo Machinery (SAM)
09/2009	Diplom in Mechanical Engineering University of Kaiserslautern, Germany Diploma Thesis: Implementation of a 2D Lattice Boltzmann (BGK) Algorithm with Turbulence Modeling
10/2008–03/2009	Internship abroad at Sulzer, Winterthur, Switzerland.
10/2004–09/2009	Studies in Mechanical Engineering (9 semesters) Major: Fundamentals of Mechanical Engineering Minors: Energy Technologies University of Kaiserslautern, Germany
04/2003–09/2004	Military service, paratroopers battalion Zweibrücken
03/2003	Graduation (Abitur) at Burggymnasium Kaiserslautern, Germany

Wissenschaftlicher und beruflicher Werdegang

Daniel Conrad

Ausbildung

Seit 10/2009	Promotion in Maschinenbau Technische Universität Kaiserslautern Fachbereich Maschinenbau und Verfahrenstechnik - Lehrstuhl Strömungmechanik und Strömungsmaschinen (SAM)
09/2009	Diplom in Maschinenbau Technische Universität Kaiserslautern Thema der Diplomarbeit: Implementierung eines 2D Lattice Boltzmann (BGK) Algorithmus mit Turbulenzmodellierung
10/2008–03/2009	Auslandspraktikum bei Sulzer, Winterthur, Schweiz.
10/2004–09/2009	Studium des Maschinenbaus (9 Semester) Hauptfach: Grundlagen des Maschinenbaus Nebenfach: Energietechnik Technische Universität Kaiserslautern
04/2003–09/2004	Wehrdienst, Fallschirmjägerbataillon Zweibrücken
03/2003	Abitur am Burggymnasium Kaiserslautern