Implicit, Monolithic Simulation of Newtonian and non-Newtonian Flows on Unstructured Grids Based on the Lattice Boltzmann Equations

---

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

Der Fakultät für Mathematik der

Technischen Universität Dortmund

vorgelegt von

Rashid Mahmood

Implicit, Monolithic Simulation of Newtonian and non-Newtonian Flows on Unstructured Grids Based on the Lattice Boltzmann Equations

Rashid Mahmood

Dissertation eingereicht am: 29. 5. 2012
Tag der mündlichen Prüfung: 16. 7. 2012

Mitglieder der Prüfungskommission

Prof. Dr. Stefan Turek (1. Gutachter, Betreuer)
Prof. Dr. Heribert Blum (2. Gutachter)
Prof. Dr. Joachim Stöckler
Prof. Dr. Rudolf Scharlau
Dr. Matthias Möller

Dedicated to my wife
AYESHA
and to my kids
YASHAL and NASHIT

Acknowledgements

First of all I would like to express my special gratitude to my main supervisor Professor Dr. Stefan Turek for providing me the opportunity to work under his supervision. He was a constant source of inspiration for me and he was always ready to share his invaluable experience with me. I also thank him for his participation in all the discussions and presentations related to my work offering me useful suggestions to improve the quality and standard of my thesis. He is providing an enjoyable working atmosphere, taking care of professional and social aspects even beyond his busy time schedule.

I would like to express my appreciation to my co-supervisor Dr. Thomas Hübner. I am thankful for his participation in discussions related to theory and implementation of Lattice Boltzmann approach and offering me useful suggestions. I would also like to thank Professor Heribert Blum for accepting to review my work.

I thank all the members of the Institute of Applied Mathematics, who supported me in any respect during my stay at TU Dortmund. I would also like to thank Dr. Abderrahim Ouazzi for very helpful discussions and guidance throughout my studies. My sincere thanks are due to Mr. and Mrs. Shafqat Hussain for supporting and facilitating me in quite difficult moments of my life and making my stay at Dortmund memorable.

I greatly acknowledge the financial support by the Higher Education Commission (HEC) of Pakistan.

Finally, I would like to express my deep gratitude to my family who supported me and showed much patience during the last five years of my life. I hope I will be able to fully repay their kindness and understanding.

Dortmund, May 29, 2012

Rashid Mahmood

Contents

1

Introduction

1.1. State-of-the-art

Recently, the Lattice Boltzmann Method (LBM) has appeared as a potential alternative to conventional methods in computational fluid dynamics. The popularity of the Lattice Boltzmann method is based on the simplicity of the scheme, which approximates a quite complicated system, namely the Navier-Stokes equations. The basic LB algorithm consists of two explicit decoupled steps, the stream-step and the collide-step. These steps are performed alternatingly and combined with no-slip boundary conditions for the domain boundaries or rigid obstacles. In the streaming step, particles move along lattice vectors to the neighboring nodes and in the second step, the particles are redistributed after local collisions. This simple LB algorithm can be implemented by roughly a single page of C or Matlab code but the explicit nature of this method implies tight restrictions concerning stability (discussed in [39] for the standard BGK LBM), or the Courant Friedrichs Levy (CFL) condition.

Historically, the lattice Boltzmann method was derived from the Lattice Gas Automata (LGA) and in particular the FHP model (named after Frisch, Hasslacher and Pomeau who published the model in [21]). Later, Chapman and Cowling [11], by using Chapman Enskog multiscale expansion, proved that this model is able to reproduce the weakly compressible Navier Stokes equations by modelling the flow with fictitious particles moving on a regular lattice and obeying very simple collision rules. The LGA comprises of two steps: streaming and collision. The streaming is represented by a lattice pattern and collision is done by a collision operator. From a physical point of view, on macroscopic level, these two steps simulate convection and diffusion phenomena, respectively. The equation for LGA is

$$n_\alpha(\mathbf{x}+\mathbf{e}_\alpha, t+1) = n_\alpha(\mathbf{x},t) + \Omega_\alpha[n(\mathbf{x},t)], \qquad \alpha = 0,1,...,M \tag{1.1}$$

where n_α is a Boolean variable that is used as an indication of the presence with $n_\alpha = 1$ or absence $n_\alpha = 0$ of particles, $\mathbf{e}_\alpha$ is the local constant particle velocity, Ω_α is the collision operator, and M is the number of directions of the particle velocities. The physical variables, density and velocities are defined by

$$\rho = \sum_{\alpha=0}^{M} < n_\alpha > \quad , \quad u_i = \frac{1}{\rho} \sum_{\alpha=0}^{M} < n_\alpha > e_{\alpha i} \tag{1.2}$$

in which $< n_\alpha >$ denotes the ensemble average of n_α in statistical physics. It is observed that simulations generated with a LGA are very noisy due to its Boolean nature [16]. Also, the numerical procedure involves probabilities which reduces the efficiency of a LGA.

In order to overcome the difficulties faced by LGA, the Lattice Boltzmann method was introduced where Boolean variables of LGA are replaced by particle distribution functions f_α and accordingly equation (1.1) takes the form

$$f_\alpha(\mathbf{x}+\mathbf{e}_\alpha, t+1) = f_\alpha(\mathbf{x},t) + \Omega_\alpha[f(\mathbf{x},t)], \qquad \alpha = 0,1,...,M \tag{1.3}$$

This approach eliminates the statistical noise in a LGA and at the same time it retains all the advantages of locality in the kinetic form of a LGA. Due to the complicated nature of the collision term Ω_α, various simplifications have been proposed in the literature. However, the most well accepted version due to its simplicity and efficiency is the Bhatnagar-Gross-Krook (BGK) model [4]. This approximation of the collision term writes the collision operator as a function of the difference between the value of the distribution function and the equilibrium distribution function as

$$\Omega_\alpha = -\frac{1}{\tau}(f_\alpha - f_\alpha^{eq}) \tag{1.4}$$

where τ is the single time relaxation parameter, and f_α^{eq} is obtained by expanding the Maxwell-Boltzmann distribution function in Taylor series of $\mathbf{u}$ up to second order [12, 30], and can be expressed in general as as

$$f_\alpha^{eq} = \rho W_\alpha \Big[1 + \frac{3}{c^2}(\mathbf{e}_\alpha \cdot \mathbf{u}) + \frac{9}{2c^4}(\mathbf{e}_\alpha \cdot \mathbf{u})^2 - \frac{3}{2c^2}\mathbf{u}^2\Big] \tag{1.5}$$

in which $c = \frac{\Delta x}{\Delta t}$ is the particle speed and the coefficients W_α depend on the discrete velocity set $\{\mathbf{e}_\alpha\}$. The macroscopic density ρ and velocity vector $\mathbf{u}$ are related to the distribution function by

$$\rho = \sum_{\alpha=0}^{M} f_\alpha \quad , \quad \rho\mathbf{u} = \sum_{\alpha=0}^{M} f_\alpha e_\alpha \tag{1.6}$$

The pressure can be calculated from $p = c_s^2\rho$ with the speed of sound $c_s = c/\sqrt{3}$ and the viscosity of the fluid is $\nu = c_s^2(\tau - \frac{\Delta t}{2})$.

The BGK choice of the collision operator makes the lattice Boltzmann equation extremely simple and efficient; hence it has been widely used in many different areas including the investigation of multiphase flows and multicomponent flows [25–27], turbulent flows [15, 72], fluid flow in porous media [1, 9, 28], simulation of heat transfer and reactive flows [14, 36, 46] and magnetohydrodynamics [13].

As will be described in section 2.5.2, one of the key features of the LBM with fully structured lattices is that a parallel implementation is very easy. Due to the availability of modern computational hardware, a lot of research is going on, especially in view of High Performance Computing. Advanced Lattice Boltzmann solvers on CPUs and GPUs have been implemented by Tölke and Krafczyk [64], Thürey [63] and Pohl [49]. In the same direction, Geveler et al. [24, 53] developed different kernels based on lattice Boltzmann

methods using multi and manycore architectures. We would like to mention the SKALB project (www.skalb.de) as a recent initiative with the aim of efficiently implementing and the development of Lattice Boltzmann based CFD solvers for the simulation of complex multiphysics applications on petascale class computers.

However, most of the research has been done on the structured framework and only a few references can be found utilizing Lattice Boltzmann approach on unstructured grids. Also, an implicit treatment of LBE appears in only very few places. Some authors tried to work on unstructured grids by combining LBM with finite element and finite volume approaches with more or less success. The motivation of our work is to use implicit time discretization approaches for the lattice Boltzmann equation on unstructured grids introduced recently in [33].

1.2. Contribution of the thesis

The aim of this thesis is twofold, first to overcome the limitations faced by standard Lattice Boltzmann method which is thoroughly discussed in literature and secondly, to extend the new approach initiated by Hübner in [33] towards non-Newtonian fluids, in particular, the generalised Newtonian fluids. In these fluids the viscosity of the fluid is depending on the shear rate, but there are no memory effects as in viscoelastic fluids. The viscosity is either decreasing or increasing with growing shear rate. These two cases correspond to shear-thinning or shear-thickening effects. There are many industrial flow problems in which the non-Newtonian viscosity effects are very important. In case of Newtonian fluids the only nonlinearity is due to the collision term whereas for non-Newtonian case we have additional nonlinearity due to the nonlinear viscosity function.

Unlike the standard LBM, we consider unstructured (off-lattice) discretizations on nonuniform triangular grids in combination with a fully-implicit time-discretization as proposed in [33]. In this work, we treat mainly the discerete Boltzmann equation also called discrete velocity model (DVM) with the Bhatnagar-Gross-Krook collision operator as in [50], given by the following equations

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \nabla f_i = -\frac{1}{\tau}(f_i - f_i^{eq}) \qquad i = 0, 1, ..., M \tag{1.7}$$

which means single-time relaxation of the f_i towards equilibrium f_i^{eq} on a typical timescale τ. We consider the two dimensional case, and therein the D2Q9 model that has the necessary conservation and symmetry properties. The set of discrete lattice vectors $\mathbf{e}_i$ is given as

$$\mathbf{e}_i = \begin{cases} (0,0) & i = 0 \\ (\cos\theta_i, \sin\theta_i) & i = 1,3,5,7 \\ \sqrt{2}(\cos\theta_i, \sin\theta_i) & i = 2,4,6,8 \end{cases} \tag{1.8}$$

where $\theta_i = (i-1)\pi/4$. The so-called 'velocities' $\boldsymbol{\xi}_i = c\mathbf{e}_i$ correspond to the scaled lattice vectors, with the parameter c which determines the speed of sound, $c_s = c/\sqrt{3}$, of the

system. The equilibrium term f_i^{eq} is given by

$$f_i^{eq} = \rho W_i \Big[1 + \frac{3}{c^2}(\boldsymbol{\xi}_i \cdot \mathbf{u}) + \frac{9}{2c^4}(\boldsymbol{\xi}_i \cdot \mathbf{u})^2 - \frac{3}{2c^2}\mathbf{u}^2 \Big] \tag{1.9}$$

with weights W_i equal to $\frac{4}{9}$ for the zero velocity, $\frac{1}{9}$ for the orthogonal and $\frac{1}{36}$ for the diagonal velocities. The distributions are summed up to get the moments of density and momentum:

$$\rho = \sum_{i=0}^{8} f_i \quad , \quad \rho\mathbf{u} = \sum_{i=0}^{8} \boldsymbol{\xi}_i f_i \tag{1.10}$$

In this thesis, we apply a simpler model proposed by He and Luo [29], which reduces the equilibrium term to a quadratic polynomial in the primary solution variables. This simpler equilibrium term is obtained by explicitly substituting $\rho = \rho_0 + \delta\rho$ in which ρ_0 is a constant density and $\delta\rho$ represents the fluctuations in the density and neglecting the terms proportional to $\delta\rho(u/c)$ and $\delta\rho(u/c)^2$ which are of the order $\mathcal{O}(Ma^3)$ or higher, where $Ma = U/c$ is the Mach number. The equilibrium distribution function of this, the so-called 'incompressible model' is given by

$$f_i^{eq} = W_i \Big[\rho + \rho_0 (\frac{3}{c^2}(\boldsymbol{\xi}_i \cdot \mathbf{u}) + \frac{9}{2c^4}(\boldsymbol{\xi}_i \cdot \mathbf{u})^2 - \frac{3}{2c^2}\mathbf{u}^2) \Big] \tag{1.11}$$

with $\rho_0 = 1$ and corresponding weights W_i. This approach approximates the Navier-Stokes equation (see [50]), whereas the viscosity ν is included via $\nu = c_s^2\tau$ in the DVM (see [65]) and the distributions are summed up to get the moments of density and velocity:

$$\rho = \sum_{i=0}^{8} f_i \quad , \quad \rho_0 \cdot \mathbf{u} = \sum_{i=0}^{8} \boldsymbol{\xi}_i f_i \tag{1.12}$$

There is an additional error component associated with this system which is of order $\mathcal{O}(Ma^2)$ (see [22, 52]). This error is termed as the compressibility error. It should be noted that in the equation (1.7), the parameter c appears as a linear scaling factor to the differential operator, and quadratic in the collision term through τ. This means that for small c and large viscosity ν the equation is transport dominated, while increasing c makes it collision dominated, the same for decreasing ν. The interplay between c, h, and ν is thoroughly discussed in [33] since it vastly influences the approximation.

The extension of the Lattice Boltzmann equation to non-Newtonian fluids has recovered limited attention so far, in spite of the fact that a reliable extension of LBE to simulate non-Newtonian fluids would be very valuable. LBE offers excellent possibilities for simulating non-Newtonian fluids due to the reason that the shear tensor can be computed locally, with no need of taking space derivatives of the velocity field [69]. An early extension of the LB model to simulate non-Newtonian fluid flows has been reported by Aharonov and Rothman [2] using the Power-Law model. Gabbenelli et al. [23] have extensively tested the accuracy of the LB model for non-Newtonian fluid flows using the truncated power law model. Kehrwald [37] simulated a shear thinning flow using the Carreau-Yasuda model and instead of using velocity gradients, they used intrinsic quantities of the LB model to calculate the strain rate. A second order accurate LB model was developed by Boyd et al. [6] to study shear thinning and shear thickening flow behaviour for a range of Power-Law

model parameters. Malaspinas et al. [42] simulated the Power Law and Carreau-Yasuda fluids in the cases of steady Poisseuille and 4:1 contraction flows.

Regarding our contribution for the case of non-linear viscosity, we modify the monolithic approach given in chapter 3 to work for shear dependent viscosities. The idea behind this extension is to determine the value of relaxation time locally in such a way that the desired value of viscosity is recovered. One should note that the viscosity itself is related to the local rate of strain $\dot{\gamma}$ through the constitutive equation for the stress tensor as given for non-Newtonian models [23] while $\dot{\gamma}$ in the Lattice Boltzmann framework can be taken as a function of non-equilibrium functions and the relaxation time τ which itself depend on the viscosity. Due to this cycle one cannot get an explicit expression for the viscosity and hence an implicit treatment of the viscosity is very difficult. Therefore, in each non-linear iteration we use the viscosity calculated at the previous iteration and instead of using full Newton method we use a variant of Newton where we treat the advection part fully implicitly like we did in Newtonian case, but we treat the non-linear viscosity in a semi-implicit way.

1.3. Outline of the thesis

The outline of this thesis is the following. In Chapter 2 we give a brief introduction to computational fluid dynamics and some classical approaches to solve CFD problems. In the same chapter, we also present various ingredients of the standard Lattice Boltzmann method which can be considered as a special case of our general approach to Lattice Boltzmann Equation (LBE) and we highlight some advantages of this method over traditional approaches to CFD. The implicit time discretization of the Lattice Boltzmann Equation and our special high order upwind space discretization along with a useful sorting algorithm, is presented in Chapter 3. Various types of initial and boundary conditions used in LBE are given in Chapter 4. We write the resulting nonlinear problems in matrix form in Chapter 5 and discuss various nonlinear and linear solvers for the solution of the problem. In Chapter 6, a detailed solver analysis is given to show the efficiency of our nonlinear and linear solvers examining the *driven cavity* problem at different Reynold numbers and with different start procedures. We also discuss the aspect of temporal accuracy by presenting the results for nonstationary *flow around cylinder*. We modify the configuration of the well known *flow around cylinder* benchmark [58], using a time dependent inflow profile and plot drag, lift and pressure forces acting on the cylinder. Chapter 7 deals with the extension of our new approach towards non-Newtonian flow problems, in particular, using the Power Law and Carreau models for two dimensional channel flow and driven cavity configurations. The results show that our approach is applicable to a wide range of shear-dependent viscosities. Finally, we give a concluding summary in Chapter 8 and some future plans to couple LBE with the discontinuous Galerkin approach in order to get higher order space discretization schemes to reduce the discretization error and the number of unknowns considerably.

2

Computational Fluid Dynamics (CFD)

In this chapter we give a short introduction into Computational Fluid Dynamics (CFD) and some traditional approaches to solve CFD problems.

2.1. Computational Fluid Dynamics

Computational Fluid Dynamics is defined as the set of methodologies that enable the computer to provide us with a numerical simulation of fluid flows. It is a science of predicting fluid flows by solving the mathematical equations which govern the fluid motion by means of numerical procedures. During the last few decades or so, there has been a lot of progress in developing codes for solving a variety of Fluid Dynamics problems and a lot of useful software packages for the simulation of fluids have been released. These packages are based on traditional approaches like finite difference methods, finite element methods, finite volume methods and boundary element methods. In the following sections we briefly describe these approaches together with their pros and cons.

2.2. Finite Difference Methods

Finite Difference methods [45, 47, 54] are one of the oldest methods applied to obtain numerical solution of differential equations. These methods are based on the properties of Taylor expansion and the straightforward application of the definition of derivatives. They are probably the simplest methods to apply on the uniform meshes. A finite difference method proceeds by replacing the derivatives in the differential equations by finite difference approximations. This gives a large algebraic system of equations to be solved in place of the differential equation, something that is easily solved on a computer. The limitation of structureds grid makes it difficult to apply for problems with complex geometries. Another disadvantage of finite difference methods is that a local refinement is not possible or hardly possible.

2.3. Finite Volume Methods

Finite Volume Methods [31, 40] are amongst the most versatile discretization techniques used in CFD. The popularity of FV methods is due to the fact that they are based directly on the integral form of conservation laws instead of the differential form. Due to the integral form the terms in the discretized formulation have direct physical interpretation.

These methods are well adapted for the discretization of various convection dominated partial differential equations. An additional feature of FVMs is the local conservativity of the numerical fluxes, that is the numerical flux is conserved from one discretization cell to its neighbour. This feature makes the finite volume method quite attractive when modelling problems for which the flux is of importance, such as in fluid mechanics, semi conductor device simulation, heat and mass transfer etc. They are able to handle complex geometries. Since these methods are suitable for flux calculations so they are rarely applied in solid mechanics.

2.4. Finite Element Methods

The Finite Element Method [17, 51, 61] is relatively new as compared with finite difference and finite volume methods. The origin of FEM can be traced from the field of structural analysis. In Finite Element Methods, the original PDEs are multiplied by a test function and integrated over the domain, resulting in the weak formulation of the problem. The infinite dimensional space which contains the unknown function is then replaced by a subspace of finite dimension, chosen in such a way that the approximate solution consists of piecewise polynomial functions. Suitable choice of test function leads to a system of algebraic equation. These methods are based on the definition of function values attached to the nodes of the mesh, where the numerical value of the unknown functions, and eventually their derivatives, will have to be determined. These methods can be used for any irregular shaped domain and all types of boundary conditions. Also the accuracy of the solution can be easily improved either by proper refinement of the mesh or by choosing approximations via higher degree polynomials.

2.5. Lattice Boltzmann Method

The Lattice Boltzmann method [29, 62, 73] is relatively new and it contrasts with the traditional approaches to CFD by adopting a bottom-up approach rather than a top-down approach to fluid modeling (see fig.2.1). To achieve this, it describes the fluid at a mesoscopic level and proposes models for the collision between the molecules. Though, historically LBM is a pre-averaged improvement to its predecessor, the Lattice Gas Automata [21, 55], however it can also be considered as a special finite difference form of the continuous Boltzmann equation [30].

2.5.1. Lattice structures

In lattice Boltzmann methods, we divide the computational domain into a large number of cells, each cell is called a lattice. A lattice structure with n lattice directions, defined on a m dimensional space, is commonly identified by the name $DmQn$ lattice. For 2D case, there are generally two types of lattice patterns: square lattice and hexagonal lattice. A square lattice can have 4-speed (D2Q4), 5-speed (D2Q5), 8-speed (D2Q8) or 9-speed (D2Q9) models, and the hexagonal lattice can have 6-speed (D2Q6) and 7-speed (D2Q7) models. A few 2D and 3D lattice structures are given in figures 2.2–2.3. It should be noted that the choice of lattice structure is not arbitrary because not all of these models have sufficient lattice symmetry which is a dominant requirement for the recovery of correct

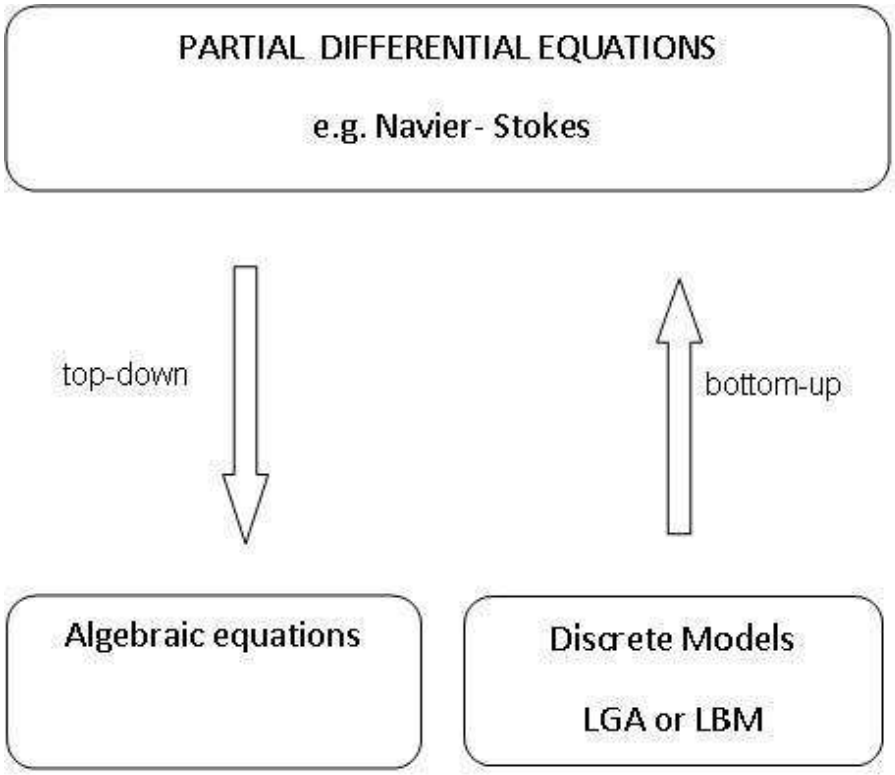


Figure 2.1: Connection between bottom up and top down approaches

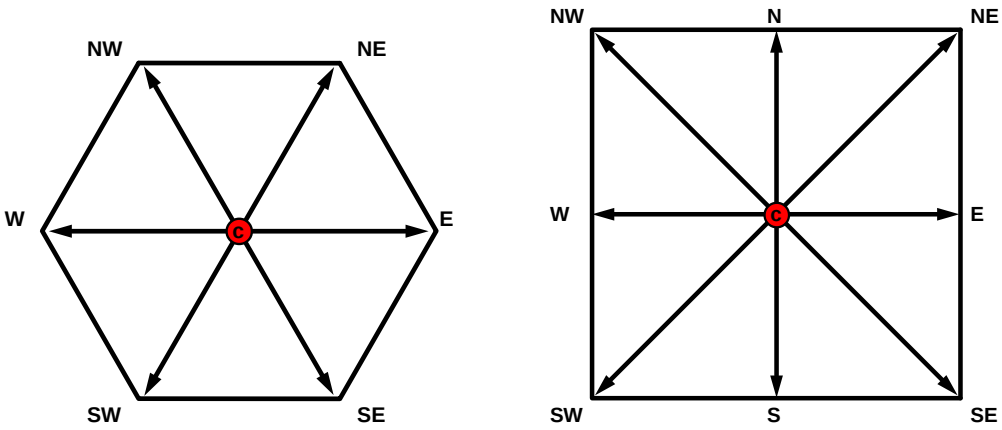
flow equations [21]. For example, the D2Q5 model is unable to recover the Navier-Stokes equations. In order to recover the correct fluid dynamic equations in the macroscopic limit, the set of discrete speeds must satisfy mass, momentum and energy conservation, as well as rotational symmetry. Only a limited class of lattices exhibits the right symmetry to ensure the conservation constraints.

2.5.2. Advantages of Lattice Boltzmann Method

The Lattice Boltzmann Method has several advantages over the traditional Computational Fluid Dynamics approaches. Some of them are:

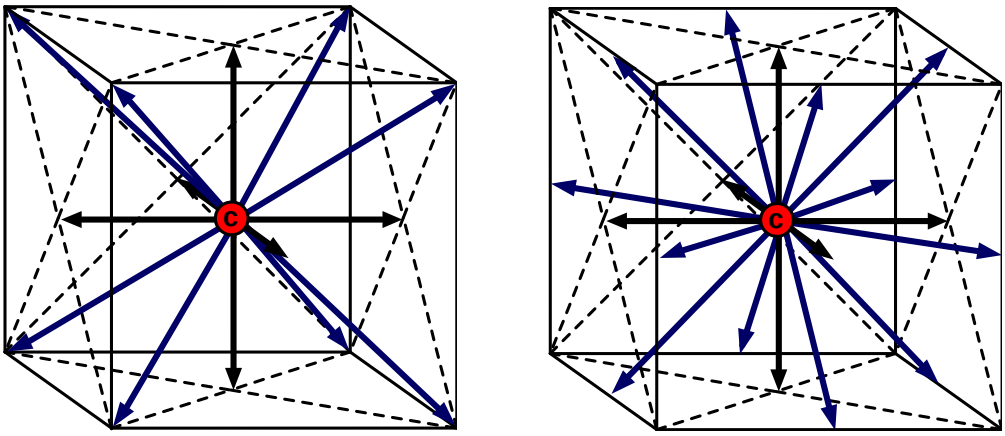
- Simple Explicit Algorithms
- Low memory requirements
- Data Locality
- High Performance Computing on many processor architectures
- No need for explicitly solving pressure dynamics

Because of these attractive features, the Lattice Boltzmann method has been particularly successful in simulations of fluid flows.



(a) D2Q7 (b) D2Q9

Figure 2.2: 2D Lattice sets



(a) D3Q15 (b) D3Q19

Figure 2.3: 3D Lattice sets

2.5.3. Differences between LBM and other CFD methods

As a computational tool, the lattice Boltzmann method differs from methods which are directly based on the Navier-Stokes equations in various aspects. They are summarized below as in [71]:

1. The Navier-Stokes equations are second-order PDEs (macroscopic equations) while the LBE consists of a set of first order PDEs (kinetic equations).

2. The Navier-Stokes solver must deal with non-linear convective term $\mathbf{u}.\nabla\mathbf{u}$, whereas in the LBE models the convection terms are linear and handled by simple advection using uniform data shifting.

3. For incompressible flow, the Navier-Stokes solver needs to solve the Poisson equation for the pressure, which involves global data communication. In the LBE methods, the pressure is obtained through an equation of state and data communication is always local.

4. Usually in the LBE method, the grid Courant-Friedrichs-Lewy (CFL) number is equal to 1, based on the lattice units of $\Delta x = \Delta t = 1$. Also, the coupling between the discretized momentum space and physical space leads to regular square grids.

5. Due to the kinetic nature of the Boltzmann equation, the physics associated with the molecular level interaction can be incorporated more easily in the LBE model.

6. The Navier-Stokes solvers usually employ iterative procedures to obtain a converged solution; the standard LB models are usually explicit and do not need iterative procedures.

2.5.4. Sources of Errors in LBM

Compressibility Error

The Lattice Boltzmann method is a compressible discretization of the Boltzmann equation so there is a small compressibility effect present in the Lattice Boltzmann method. Compressibility errors are caused by the dependence of pressure on density. It has been shown that the Lattice Boltzmann equation represents the Navier-Stokes equations in nearly incompressible limit for the small Mach number limit $(Ma = \frac{U}{c_s} << 1)$. This error is termed as the compressibility error which is of $\mathcal{O}(Ma^2)$ (see [29]).

Discretization Error

As solution of the Lattice Boltzmann equation is generally found using some numerical method so discretization by any method will lead to some discretization error. In context of our finite difference upwind schemes, by assuming an arbitrary discretization of order γ of the convective term, Lattice Boltzmann equation accounts for an additional discretization error $\mathcal{O}(\mathrm{Ma}^{-1}h^{\gamma})$ decreasing with grid spacing h and for higher order space discretization techniques. However, it includes an inverse contribution of the Mach number and makes predicting the convergence of overall numerical scheme difficult [33].

2.6. Dimensionless Numbers

In fluid flows, each of the pressure, viscous and transient forces dominates under certain conditions. In order to investigate the importance of each of these forces, the equations of motion are written in a dimensionless form, through defining dimensionless parameters. The major dimensionless parameters used in this study are:

- **The Reynolds Number**(Re)

  The Reynolds number describes the ratio between convective inertial and viscous forces and is relevant in every flow situation. It is defined as

  $$Re = \rho UL/\eta = UL/\nu$$

  where ρ is the density, U is maximum velocity, L is a characteristic length scale, η is dynamic viscosity and $\nu = \frac{\eta}{\rho}$ is kinematic viscosity. For low Reynold numbers the viscous forces are dominant, whereas for high Reynold numbers, the inertial forces are more relevant. Thus the Reynolds number is an indicator how far the flow field is from turbulence.

- **The Mach Number**(Ma)

  The compressibility effect can be analyzed by considering the dimensionless Mach number

  $$Ma = u/c_s$$

  which relates the modulus of the flow velocity to the speed of sound. In (nearly) incompressible fluids, sound waves travel at infinite speed ($c \to \infty$) so that the Ma number tends to zero. As the lattice Boltzmann method is a "semi-compressible method", the Mach number is used to guarantee the validity of the method.

- **The Strouhal Number**(St)

  The Strouhal number is important in situations describing oscillating flow mechanisms and it is defined as

  $$St = fD/U$$

  where f is the frequency of vortex shedding, D is the characteristic length and U is the velocity of the fluid. It represents a measure of the ratio of inertial forces due to unsteadiness of the flow (local acceleration) to the inertial forces due to changes in velocity from point to point in the flow field (convective acceleration). This type of unsteady flow may develop when a fluid flows past a solid body placed in the moving stream.

- **The Knudsen Number**(Kn)

  The Knudsen number is defined as the ratio between the mean free path of molecules to a representative physical length scale

  $$Kn = \lambda/L$$

  where λ is mean free path of molecules and L is the reference length scale. This number is useful for determining whether statistical mechanics or the continuum

mechanics formulation of fluid dynamics should be used.

The following list shows different flow regimes depending on the values of the Knudsen number.

- ◇ $Kn < 10^{-3}$ continuum regime
- ◇ $10^{-3} < Kn < 10^{-1}$ slip regime
- ◇ $10^{-1} < Kn < 10^{1}$ transitional regime
- ◇ $10^{1} < Kn$ free molecular regime

2.7. Invariants of a tensor

From a tensor τ, three independent scalars can be formed by taking the trace of τ, τ^2 and τ^3

$$\begin{aligned} I &= tr\tau &&= \sum_i \tau_{ii} \\ II &= tr\tau^2 &&= \sum_i \sum_j \tau_{ij}\tau_{ji} \\ III &= tr\tau^3 &&= \sum_i \sum_j \sum_k \tau_{ij}\tau_{jk}\tau_{ki} \end{aligned}$$

They are called invariants of the tensor τ, because their values are independent of the choice of coordinate system to which the components of τ are referred. Other scalars can, of course be formed, but they will be combination of these three. For example

$$\begin{aligned} I_1 &= I \\ I_2 &= \frac{1}{2}(I^2 - II) \\ I_3 &= \frac{1}{6}(I^3 - 3I.II + 2III) = det\tau \end{aligned}$$

The second invariant is very important in many cases, for example, the second invariant of rate of strain tensor is used in constitutive equations when dealing with generalized Newtonian fluids (see section 7.1).

3

Time and Space Discretization of LBE

In this chapter a short introduction into the space and time discretization for the Lattice Boltzmann Equation is presented. Unlike the standard LBM, we consider unstructured (off-lattice) discretizations on nonuniform triangular grids in combination with a fully-implicit time-discretizations developed by Hübner in [33]. Time-independent problems are efficiently solved in a direct way [35] while nonstationary benchmark problems considered in this thesis are also successfully simulated with second order accuracy and large time step independent of the mesh width: In space, we use a FEM-like constant characteristic upwind scheme for the transport-term. This second order accurate finite-difference discretization with a special resorting of the unknowns leads to lower triangular transport matrices which in turn can also be used as a special preconditioner for transport dominated cases as will be described in Section 5.5.1.

3.1. Implicit time-discretization

We start with the time-discretization of the discrete Lattice Boltzmann equation

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \nabla f_i = -\frac{1}{\tau}(f_i - f_i^{eq}). \tag{3.1}$$

A numerical, off-lattice treatment usually begins with integrating the equation (3.1) over the interval $[t_n, t_{n+1}]$. Accordingly, we write

$$f_i^{n+1} - f_i^n + \Delta t \int_{t_n}^{t_{n+1}} \boldsymbol{\xi}_i \cdot \nabla f_i dt = \int_{t_n}^{t_{n+1}} \Omega_i dt$$

with the usual collision term $\Omega_i = -\frac{1}{\tau}(f_i - f_i^{eq})$. We evaluate both integrals appearing in the above equation by a general one-step θ scheme given by

$$f_i^{n+1} - f_i^n + \Delta t[\theta \boldsymbol{\xi}_i \cdot \nabla f_i^{n+1} + (1-\theta)\boldsymbol{\xi}_i \cdot \nabla f_i^n] = \Delta t[\theta \Omega_i^{n+1} + (1-\theta)\Omega_i^n] \tag{3.2}$$

in which θ can be chosen arbitrarily from the interval $[0,1]$ which give rise to explicit and implicit schemes of different orders. For example:

- $\theta = 0$ corresponds to the explicit Euler method:

$$f_i^{n+1} + \Delta t\,(\boldsymbol{\xi}_i \cdot \nabla f_i^n - \Omega_i^n) = f_i^n \tag{3.3}$$

- $\theta = 1$ yields the implicit Euler discretization which is first order in time:

$$f_i^{n+1} + \Delta t \left(\boldsymbol{\xi}_i \cdot \nabla f_i^{n+1} - \Omega_i^{n+1} \right) = f_i^n \tag{3.4}$$

- $\theta = 1/2$ yields the second order Crank-Nicolson scheme:

$$f_i^{n+1} + \frac{\Delta t}{2} \left(\boldsymbol{\xi}_i \cdot \nabla f_i^{n+1} - \Omega_i^{n+1} \right) = \frac{\Delta t}{2} \left(\Omega_i^n - \boldsymbol{\xi}_i \cdot \nabla f_i^n \right) + f_i^n \tag{3.5}$$

In order to treat transient flow problems a high (higher than first) order time-discretization is necessary to obtain sufficient accuracy using moderate up to large time steps, while a first order scheme usually demands micro-timestepping. Furthermore, we can proceed straightforwardly by combining the stability of equation (3.4) with the accuracy of equation (3.5) in the so-called fractional step θ scheme described in [68].
It is also possible to omit the time-dependence for the direct stationary treatment of steady-state problems with equation

$$\boldsymbol{\xi}_i \cdot \nabla f_i(x) + \frac{1}{\tau}(f_i(x) - f_i^{eq}(x)) = 0 \tag{3.6}$$

which is also used in Chapter 7 for the case of non-constant viscosity.

Independent of the applied time-scheme the discretization needs to be completed in space, which will be accomplished by a special finite difference technique in the next section.

3.2. The short-characteristic discretization in space

The convective term $\boldsymbol{\xi}_i \cdot \nabla f_i$ appearing in equation (3.1) can be discretized in space using finite differences [12], finite elements [41], [19] or finite volume method [48]. In this section we describe a quite general finite difference scheme for the discretization of the convective term, since we allow unstructured meshes. This task has been performed very efficiently in [32], [34] using a backward difference scheme of up to second order accuracy. For each of the constant characteristics we regard the transport problem as a simple one dimensional differential equation. Actually, this procedure does not only apply for the set of lattice velocities, but for any arbitrary characteristic β (see Fig. 3.1). Therefore, and for reason of simplicity, we assume a hyperbolic equation with pure convection

$$\mathbf{n}_\beta \cdot \nabla u(x) = f(x), \tag{3.7}$$

for a function $u : R^d \to R$ with the unity vector $\mathbf{n}_\beta$. In the following we describe the construction of the so-called upwind discretization procedure for the transport term in (3.7), of first and second order accuracy, respectively.

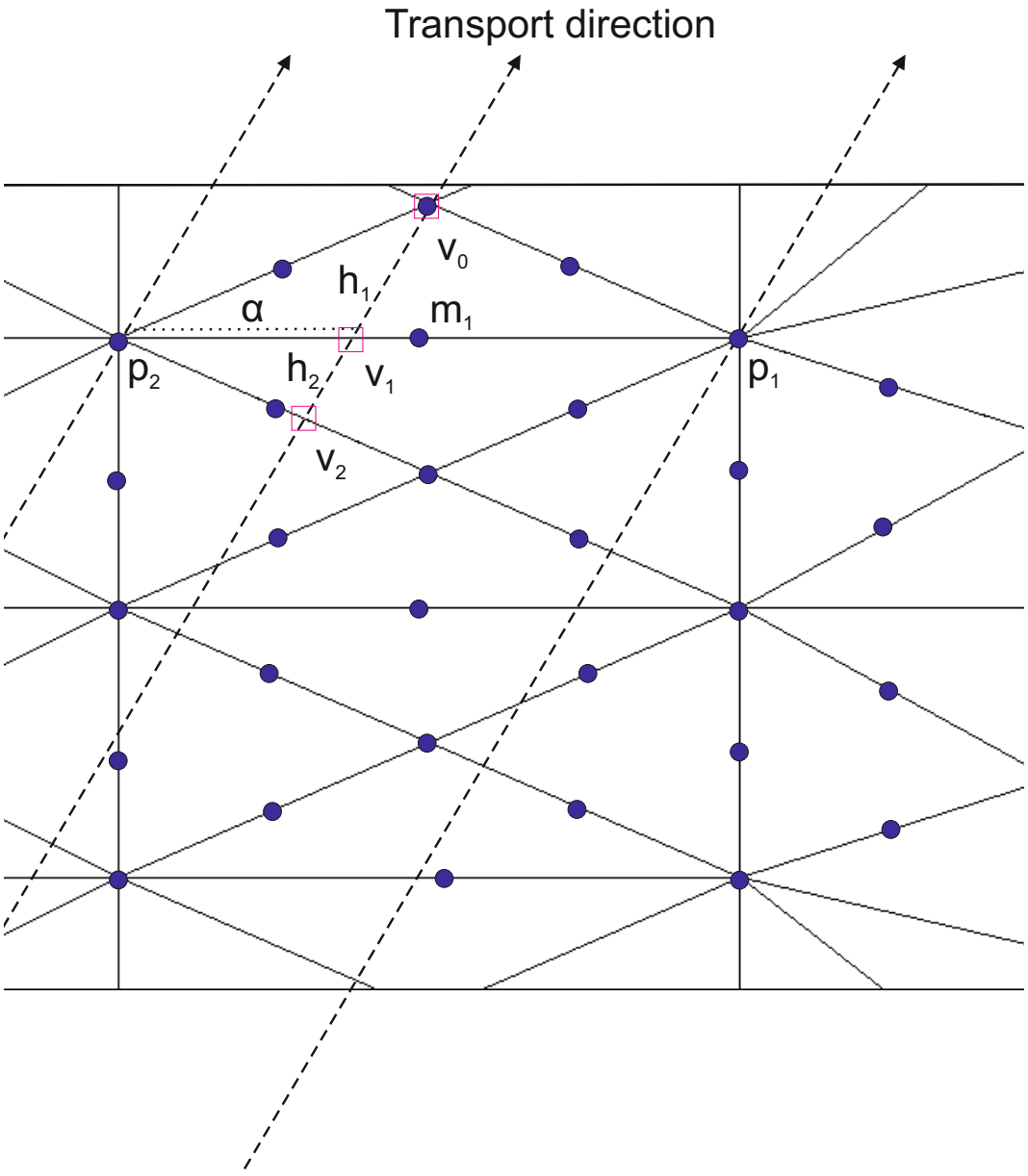


Figure 3.1: Constant characteristic upwind

3.2.1. First order upwind

Due to the constant characteristics in the Boltzmann equation as well as in the exemplary convection equation (3.7), we can view the problem as purely one dimensional as in Fig. 3.2. Consequently, we can write the spatial derivative as

$$\mathbf{n}_\beta \cdot \nabla u(v_0) = u'(v_0) = \frac{u(v_0) - u(v_1)}{h_1} + O(h_1), \tag{3.8}$$

using an upwinding of first order (to be denoted as upw1). This approximation yields a backward difference quotient and we denote the linear operator as

$$\nabla_{upw1} u(v_0) := \frac{u(v_0) - u(v_1)}{h_1}. \tag{3.9}$$

This means we discretize in each grid-point using local and backward information, in Fig. 3.1 represented by nodes v_0 and v_1, respectively.

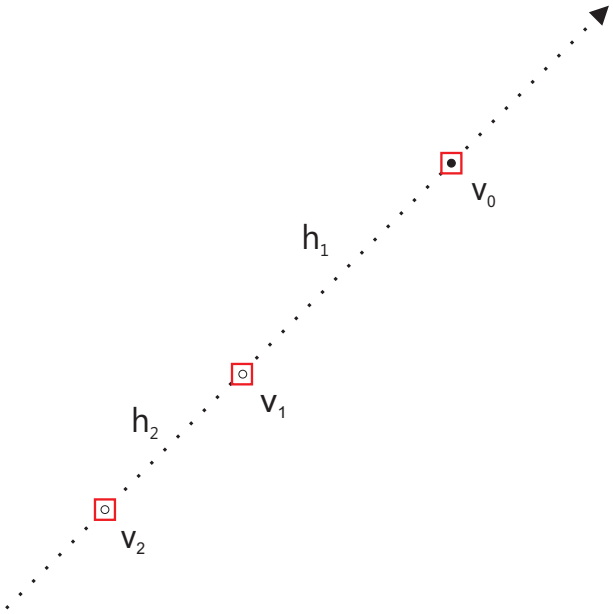


Figure 3.2: 1D view along the characteristic

Interpolation

Using unstructured meshes, we cannot expect to come across actual grid nodes going back along the characteristics. Instead, we assume virtual nodes right on the intersection with the edge of the next backward element. The function value in the virtual node has to be interpolated from the function values in the neighbouring nodes. For first order upwind we apply linear interpolation between the solution in the corners p_1^1, p_1^2 of the respective edge, obtaining

$$u(v_1) = \alpha_1 u(p_1^1) + (1 - \alpha_1) u(p_1^2).$$

This scheme is of first order for the interpolation of $u(v_1)$ and therefore sufficient for the overall consistency of the discretization scheme (3.9).

3.2.2. Second order upwind

It is obvious that the difference quotient for the second order upwinding needs in total 3 function values, from one local and two backward nodes, to achieve the desired accuracy on arbitrary grids. The coefficients in the scheme can be found using the so-called 'polynomial fitting' technique. Due to our constant characterisitic approach, we assume a simplified problem in 1D as previously, but need to take into account a non-equidistant distribution of the grid-points. A second order scheme is supposed to give us exact results for the first differential if the solution of the equation is a quadratic polynomial. That is why we make the ansatz

$$u(x) = a + bx + cx^2$$

and use three grid-points (the local point where to evaluate u' and two upwind points) to determine the unknown coefficients. For the polynomial fitting it is sufficient to assume:

$$\begin{aligned} x_0 &= 0 \\ x_1 &= -h_1 \\ x_2 &= -h_1 - h_2 = -r \cdot h_1 \quad r > 1 \end{aligned}$$

In order to approximate the first differential of u in x_0, we differentiate the ansatz function and obtain

$$\frac{\partial u}{\partial x} = b + 2cx$$

Evaluating the differential in 0 shows that we need to determine the coefficient $b = u'(0)$. We obtain a determined linear system for the coefficients a, b, c when we evaluate the ansatz function in the three points $u_0 = u(x_0),\ u_1 = u(x_1),\ u_2 = u(x_2)$

$$\begin{array}{rcl} u(0) & = & a \\ u(h_1) & = & a - bh_1 + ch_1^2 \\ u(-rh_1) & = & a - brh_1 + c(rh_1)^2 \end{array} \longrightarrow \begin{pmatrix} 1 & 0 & 0 \\ 1 & -h_1 & h_1^2 \\ 1 & -rh_1 & (rh_1)^2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix}$$

The unique solution for the coeficient b is

$$b = \frac{-(1-r^2)u_0 - r^2 u_1 + u_2}{h_1(r^2 - r)} = (\frac{\partial u}{\partial x})_0$$

We summarize that we found the coefficients for the *second order upwind* scheme (to be denoted as upw2) in v_0 (with $h_1 + h_2 = rh_1$): The equidistant case ($h_1 = h_2 \Leftrightarrow r = 2$) results in the well known scheme

$$u'(v_0) \sim \frac{3u(v_0) - 4u(v_1) + u(v_2)}{2h_1}$$

Analogously, the LB differential operator will be discretized for each of the 8 constant characteristics using a scaling by parameter c:

$$\boldsymbol{\xi}_i \cdot \nabla f_i = \begin{cases} c\mathbf{n}_i \cdot \nabla f_i & , \quad \text{othogonal vectors} \\ \sqrt{2}c\mathbf{n}_i \cdot \nabla f_i & , \quad \text{diagonal vectors} \end{cases}$$

Additionally the diagonal characteristics have a scaling factor of $\sqrt{2}$, due to the construction of the Lattice Boltzmann scheme.

FEM like Interpolation:

To achieve the overall second order of the previously described scheme, it is not sufficient to use first order interpolation between the corners on the considered edge. We need additional degrees of freedom and place them in the edge midpoints. Thus, we obtain a triangulation corresponding to second order finite elements with 6 degrees of freedom each (see Fig. 3.1). Function values in each virtual node, for example in v_1, are interpolated from 3 edge values in the following way.

$$u(v_1) = \lambda_1 u(p_1^1) + \lambda_2 u(p_1^2) + \lambda_m u(m_1)$$

With weights depending on the value α_1:

$$\lambda_1 = (1-\alpha_1)(1-2\alpha_1)\,,\ \lambda_2 = \alpha_1(2\alpha_1-1)\,,\ \lambda_m = 4\alpha_1(1-\alpha_1)$$

The following exemplary α values show how the interpolation scheme collapses back into the actual grid nodes.

$$\begin{aligned}
\alpha_1 &= 0.0 \quad \Rightarrow \quad \lambda_1 = 1, \lambda_2 = 0, \lambda_m = 0 \\
\alpha_1 &= 1.0 \quad \Rightarrow \quad \lambda_1 = 0, \lambda_2 = 1, \lambda_m = 0 \\
\alpha_1 &= 0.5 \quad \Rightarrow \quad \lambda_1 = 0, \lambda_2 = 0, \lambda_m = 1
\end{aligned}$$

3.3. Special sorting technique

Due to the upwind discretization using information from 'backward' nodes, one is inclined to think that, starting from the inflow boundary and 'traversing' the domain to the opposite wall, one can directly solve a pure convection problem as in Eq. (3.7). In fact, as described in [34], for each constant characteristic β, resp., lattice vector $\boldsymbol{\xi}_i$, it is possible to find a numbering of the grid nodes so that the resulting discretization matrix is lower triangular. The numbering is determined in a preprocessing routine and differs for each direction. At simulation time, whenever one has to solve a transport step (for example in preconditioning), it is possible without actually inverting a matrix but by following the numbering and performing a simple backward insert of the solution starting from the given boundary values. The sorting algorithm is based on topological sorting from the field of graph theory (see [18]) and can be written in pseudo-code (see Alg. 3.1) which was previously presented in [32].

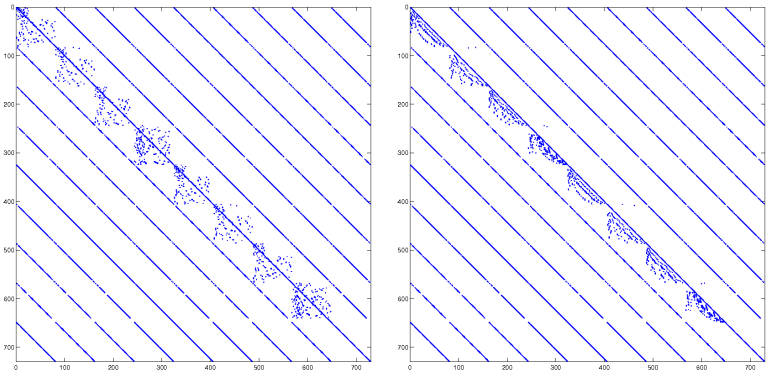
Algorithm 3.1 Topological Sorting.

ORDER (QUEUE[*], IN-NODES[*][*], OUT-NODES[*][*], NVT)

0. INIT:
i.) QUEUE[*]=0, OUTDEG[*] = 0 , k = 1
ii.) FOR EACH ENTRY IN OUT-NODES[i][*] **DO** OUTDEG[i]++
iii.) FOR EACH i **WITH** OUTDEG[i]=0 **DO** i $\rightarrow$ QUEUE

1. DO WHILE k $<$ NVT
a.) v=QUEUE[k]
b.) IF v=0 **THEN OUTPUT** 'Graph is cyclic!', **STOP!**
c.) FOR EACH j **IN** IN-NODES[v][*] **DO:**
d.) OUTDEG[j]- -
e.) **IF** OUTDEG[j]=0 **THEN** j $\rightarrow$ QUEUE
f.) END FOR
g.) k = k + 1

The effect of this sorting algorithm is presented in figure 3.3 (also given in [33]). After applying the according permutation matrices to the 8 transport parts situated on the block diagonal of the system matrix, we see a change in the original location. The entries that belong to the finite difference discretization of the differential operator are permuted into a lower triangular allocation in the matrix. The off diagonals contain the collision-term entries.



(a) Unsorted (b) Sorted

Figure 3.3: Symbolic representation of effect of sorting algorithm

3.4. Algebraic systems

The finite difference upwinding of second order presented in section (3.2.2) yields lower triangular transport matrices due to the resorting of unknowns using special algorithm. However, this FD approach has a special consequence in the context of the Boltzmann equation. Let $\nabla_{h,i}$ be an approximation of order γ of the normal derivative in the i-th direction. Then, the transport in Eq. (3.1) yields the asymptotic relation

$$\boldsymbol{\xi}_i \cdot \nabla f_i = c(\boldsymbol{e}_i \cdot \nabla f_i) = c(\nabla_{h,i} f_i + O(h^\gamma)) = c\nabla_{h,i} f_i + O(c\, h^\gamma) \tag{3.10}$$

for unity lattice vectors $\boldsymbol{e}_i$. In total, the error due to finite grid spacing is amplified by the sound parameter c. With the relation $Ma = O(1/c)$ we have therefore a specific discretization error $O(\mathrm{Ma}^{-1}h^\gamma)$ with an inverse Mach number influence which is opposed to the compressibility error $O(\mathrm{Ma}^2)$. A balancing of the two contributions is achieved by setting the simulation parameters according to

$$h^\gamma = O(\mathrm{Ma}^3) = O(1/c^3)$$

which then yields the optimum asymptotic and quadratic convergence in the Mach number. The assumption was confirmed in [35] by a numerical analysis of the L_2-error for different steady-state CFD problems.

However, in order to write the algebraic system we identify two operators. First, we sum up the linear (discrete) transport and identity terms from Eq. (3.1) and denote

$$\mathbf{T}_i f_i := f_i + \theta\Delta t[\boldsymbol{\xi}_i \cdot \nabla f_i + \frac{1}{\tau} f_i]$$

or

$$\mathbf{T}_i^h f_i := f_i + \theta\Delta t[c\nabla_{h,i} f_i + \frac{1}{\tau} f_i]$$

Second, take the equilibrium (the remaining part of the collision-term)

$$\sum_k \omega_{ik} f_k := f_i^{eq}(\rho, \mathbf{u}) = W_i \left[\rho + \rho_0 \left(\frac{3}{c^2}(\boldsymbol{\xi}_i \cdot \mathbf{u}) + \frac{9}{2c^4}(\boldsymbol{\xi}_i \cdot \mathbf{u})^2 - \frac{3}{2c^2}\mathbf{u}^2\right)\right].$$

In short, we have the nonlinear form

$$f_i^{eq} = \sum_k \omega_{ik} f_k \quad \text{with} \quad \omega_{ik} = \omega(i,k,c,\mathbf{u}) \tag{3.11}$$

the corresponding weights obtained after resolving the equilibrium (1.9) in terms of the distributions f_i. The discrete equation obtained so far in these terms is

$$\mathbf{T}_i f_i^{n+1} - \frac{\theta \Delta t}{\tau} \sum_k \omega_{ik} f_k^{n+1} = g_i^n \tag{3.12}$$

where the right hand side is taken from the previous time step

$$g_i^n = f_i^n - (1-\theta)\Delta t \left(c \nabla_{h,i} f_i^n + \frac{1}{\tau} f_i^n - \frac{1}{\tau} f_i^{n,eq} \right).$$

The corresponding nonlinear algebraic block-system can be written as

$$\left(\begin{bmatrix} \mathbf{T_0} & & & \\ & \mathbf{T_1} & & \\ & & \ddots & \\ & & & \mathbf{T_8} \end{bmatrix} - \frac{\theta \Delta t}{\tau} \begin{bmatrix} \omega_{00} & \omega_{01} & \dots & \omega_{08} \\ \omega_{10} & \omega_{11} & & \vdots \\ \vdots & & \ddots & \vdots \\ \omega_{80} & \dots & \dots & \omega_{88} \end{bmatrix} \right) \begin{bmatrix} f_0^{n+1} \\ f_1^{n+1} \\ \vdots \\ f_8^{n+1} \end{bmatrix} = \begin{bmatrix} g_0^n \\ g_1^n \\ \vdots \\ g_8^n \end{bmatrix} \tag{3.13}$$

and we have to deal with a coupled algebraic system of equations which consists of a nonlinear operator introduced through the equilibrium term f_i^{eq} and a linear operator (corresponding to the discrete transport and identity terms). The former is purely local, while for the latter we obtain the favourable triangular matrix property due to a renumbering of the unknowns in the finite difference upwinding scheme.

3.5. Linearization of collision

The implicit treatment of collisions gives rise to nonlinearity in the primary variables f_i. In the macroscopic variables we have the two quadratic terms $(\boldsymbol{\xi}_i \cdot \mathbf{u})^2$ and $\mathbf{u}^2$ appearing in the equilibrium term

$$f_i^{eq} \;=\; W_i \left(\rho + \rho_0 \left(\frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{(u_1^2 + u_2^2)}{2c_s^2} \right) \right)$$

of the (incompressible) SRTBGK model, but their linearization is quite straightforward: The general product $u_\alpha u_\beta$ composed of velocity-components is substituted by $u_\alpha \tilde{u}_\beta$ where $\tilde{\mathbf{u}}$ can be chosen for example from the solution at the previous time-step or as the last iterate in a fixed-point nonlinear solver (see also Section 5.1.1). In the primary distributions we write $\tilde{\mathbf{u}} = \sum_i \boldsymbol{\xi}_i \tilde{f}_i$. This ansatz is sufficient for a linearization of the above schemes and we complete it by eliminating the macroscopic variables. We apply the summation (1.12) for ρ and $\mathbf{u} = (u_1, u_2)^T$ and introduce the constant coefficients $D_{ik} = \mathbf{e}_i \cdot \mathbf{e}_k$, formally devising

$$(\boldsymbol{\xi}_i \cdot \mathbf{u}) = c^2 \sum_k D_{ik} f_k \quad , \quad u_1 = c \sum_k D_{1k} f_k \quad , \quad u_2 = c \sum_k D_{2k} f_k.$$

Altogether, the equilibrium is linearized as

$$\begin{aligned} f_i^{eq} &= W_i\left(\rho + \frac{(\boldsymbol{\xi}_i\cdot\mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i\cdot\mathbf{u})(\boldsymbol{\xi}_i\cdot\tilde{\mathbf{u}})}{2c_s^4} - \frac{(u_1\tilde{u}_1+u_2\tilde{u}_2)}{2c_s^2}\right) \\ &= W_i\Big(\sum_k f_k + 3\sum_k D_{ik}f_k \\ &\quad + \frac{9}{2}\sum_k D_{ik}f_k\sum_k D_{ik}\tilde{f}_k \\ &\quad - \frac{3}{2}(\sum_k D_{1k}f_k\sum_k D_{1k}\tilde{f}_k + \sum_k D_{2k}f_k\sum_k D_{2k}\tilde{f}_k)\Big) \\ &=: \sum_k \tilde{\omega}_{ik}f_k. \end{aligned} \tag{3.14}$$

Linearizing the collisions in this way we get rid of macroscopic terms and all c in the denominator cancel out. Even without them, one has to keep in mind the low Mach number nature of the approximation of the BGK model, i.e. the limit of $\mathrm{Ma} = \frac{u}{c_s} \to 0$, and c^2 appearing in the dominating term $\frac{1}{\tau}$. However, the resulting coefficients are independentent of c, so formally $\tilde{\omega}_{ik} = \tilde{\omega}_{i,k}(\tilde{f})$ holds.

3.6. Generalized equilibrium formulation

For an efficient treatment of the linear subproblems, the so-called *generalized equilibrium formulation* for the monolithic steady approach was introduced in [35]. Now we formulate it for the time stepping variant. We start with the general algebraic system 3.12, now stripped from the temporal indices, linearized and rearranged to

$$\mathbf{T}_k f_k = \frac{\theta\Delta t}{\tau} f_k^{eq} + g_k \quad k = 0,\dots,8 \tag{3.15}$$

where the temporal indices were omitted, can be rewritten as

$$f_k = \mathbf{T}_k^{-1}\left(\frac{\theta\Delta t}{\tau} f_k^{eq} + g_k\right) \quad k = 0,\dots,8 \tag{3.16}$$

and for all $i = 0,\dots,8$ we multiply with the corresponding weights $\tilde{\omega}_{ik}$ from equation (3.14) which yields:

$$\tilde{\omega}_{ik}f_k = \tilde{\omega}_{ik}\mathbf{T}_k^{-1}\left(\frac{\theta\Delta t}{\tau} f_k^{eq} + g_k\right) \quad k = 0,\dots,8$$

Summing up over k finally gives us for each linearized (generalized) equilibrium term $f_i^{eq} = \sum_k \tilde{\omega}_{ik}f_k$:

$$f_i^{eq} = \sum_k \tilde{\omega}_{ik}\mathbf{T}_k^{-1}\frac{\theta\Delta t}{\tau} f_k^{eq} + \sum_k \tilde{\omega}_{ik}\mathbf{T}_k^{-1} g_k \quad i = 0,\dots,8$$

$$\left(I - \frac{\theta\Delta t}{\tau}\begin{bmatrix} \tilde{\omega}_{00}\mathbf{T_0^{-1}} & \tilde{\omega}_{01}\mathbf{T_1^{-1}} & \dots & \tilde{\omega}_{08}\mathbf{T_8^{-1}} \\ \tilde{\omega}_{10}\mathbf{T_0^{-1}} & \tilde{\omega}_{11}\mathbf{T_1^{-1}} & & \vdots \\ \vdots & & \ddots & \vdots \\ \tilde{\omega}_{80}\mathbf{T_0^{-1}} & \dots & \dots & \tilde{\omega}_{88}\mathbf{T_8^{-1}} \end{bmatrix}\right)\begin{bmatrix} f_0^{eq} \\ f_1^{eq} \\ \vdots \\ f_8^{eq} \end{bmatrix} = \begin{bmatrix} \sum_k \tilde{\omega}_{0k}\mathbf{T_k^{-1}} g_k \\ \sum_k \tilde{\omega}_{1k}\mathbf{T_k^{-1}} g_k \\ \vdots \\ \sum_k \tilde{\omega}_{8k}\mathbf{T_k^{-1}} g_k \end{bmatrix} \tag{3.17}$$

Above implicit system matrix is applied in linear iterative solvers and allows additional preconditioning. The diagonal entries of the inverse transport matrices are explicitly known, so we can assemble the local collision entries and apply the inverse of each 9x9 block to the system. We denote the solution of the plain system (3.17) as GEF, while the block-Jacobian variant is denoted GEF($\backslash$).

4

Initial and Boundary Conditions for LBE

This chapter is devoted to describe various initial and boundary conditions that can be used while implementing the Lattice Boltzmann equation. There is a lot of literature available on different types of boundary conditions depending on the problem description and given data. One of the major difficulties while implementing boundary conditions in LBE is that one has to translate given information from macroscopic variables to distribution functions f_i, since it is the only variable to be evaluated in the Lattice Boltzmann algorithms. First, we describe initial conditions and then we present different types of boundary conditions for LBE.

4.1. Initial Conditions

In the Lattice Boltzmann equation framework there arise some problems from the discrepancy between using distributions as primary simulation variables and including macroscopic moments, for example as initial conditions or taking saved pressure and velocity to consistently recover f_i. Although it is easy to transform distributions into macroscopic variables, we do not have a simple inverse mapping, even the number of variables is not equal. To overcome these problems, we can prescribe the initial conditions in two ways. First way is to set random value between 0 and 1 for the distribution functions f_i. The other possibility is to define flow field first and calculate the local equilibrium distribution function f_i^{eq} which are known in terms of density and velocity and then use these f_i^{eq} as an initial condition for f_i i.e. $f_i = f_i^{eq}$. The error that results from this approximation can be overcome by discarding the first few steps and measuring the parameters of the flow afterwards. In the LB context, initial conditions have been suggested differently by many authors, see for example [8, 44, 59].

4.2. Boundary Conditions

The boundary conditions play an important role for the stability and convergence of the numerical method designed for the mathematical model under consideration because a numerical method includes not only the particular model equation, but also well defined initial and boundary conditions, guaranteeing both the existence and uniqueness of the approximate solution for the particular flow configuration. To solve a problem uniquely there is a need of correct boundary conditions. In case of boundary conditions there are techniques that avoid the inverse mapping as in the case of initial conditions. These tech-

niques will be discussed in the later sections.

In general, the boundary may consist of different parts

$$\Gamma = \Gamma^{-} \cup \Gamma^{+} \cup \Gamma^{\circ}$$

in which Γ^{-}, Γ^{+} and Γ° are the inflow part, outflow part and solid walls of the boundary respectively and they are defined as:

$$\Gamma_i^{-} := \{f_i(x) \quad | \quad x \in \partial\Omega, \quad \mathbf{n}_x \cdot \mathbf{e}_i < 0\}$$

$$\Gamma_i^{+} := \{f_i(x) \quad | \quad x \in \partial\Omega, \quad \mathbf{n}_x \cdot \mathbf{e}_i > 0\}$$

$$\Gamma_i^{\circ} := \{f_i(x) \quad | \quad x \in \partial\Omega, \quad \mathbf{n}_x \cdot \mathbf{e}_i = 0\}$$

where $\mathbf{n}_x$ is outward unit normal. Since we are using unstructured meshes, so boundary configuration is quite easy in our case. These are adapted to the geometry of the domain, therefore we can easily place the boundary nodes right on the wall. This is not the case with standard LBM which works on uniform grids where a boundary node can occur which is between a fluid node and an outside node. A special treatment is required to obtain higher order accurate schemes.

In our implementation we worked consecutively, starting from the eastern point and successively numbering the directions anticlockwise. Expressed as cardinal points, $\mathbf{e}_i$, $i = 1, \ldots, 8$ were taken from

$$\{E, NE, N, NW, W, SW, S, SE\}$$

with the rest particle as 9th direction. However, in literature it is common to take first the orthogonal vectors, followed by the diagonals, with the numbering according to

$$\{E, N, W, S, \ NE, NW, SW, SE\}.$$

We will apply the second convention while discussing boundary schemes.

4.2.1. No Slip or Bounce Back scheme

The bounce-back scheme is the most popular method to handle stationary no-slip boundaries. The basic idea behind the bounce-back scheme is very simple and it states that an incoming particle distribution towards the boundary is bounced back into the fluid. It means we prescribe

$$f_i(\mathbf{x}) = f_{-i}(\mathbf{x}) \quad \text{on} \quad \Gamma_i^{-}. \tag{4.1}$$

It follows that opposing contributions in the sum $\sum_i \boldsymbol{\xi}_i f_i$ results in a zero momentum, implying that this bounce back scheme leads to no-slip boundary conditions. Because of its simplicity and easy implementation, this scheme is the most popular way for no-slip boundary conditions used in the Lattice Boltzmann method for simulating fluid flows.

4.2.2. Periodic Boundary Conditions

In some situations, a periodic boundary condition may be required. For example, when a flow region consists of a number of same modules where the flow pattern repeats itself module after module, only one module is actually required to be modelled together with a periodic boundary condition. These conditions are easy to implement by assigning the incoming populations on the inlet to the corresponding outgoing populations on the outlet and assigning the incoming populations on the outlet to the corresponding outgoing populations on the inlet. Despite the simplicity, it perfectly fits the simulations of fluid flow on symmetric geometries, such as a simple 2D channel. However, for complex asymmetric geometries, such as backward facing step, we cannot use the periodic boundary conditions.

4.2.3. LADD Scheme

CFD models simulated in this work mostly define a slip-velocity $\mathbf{u}_{bc}$. In order to impose this Dirichlet boundary conditions we chose mostly a scheme described by Ladd in [38]. Therein, the above bounce-back treatment is extended by adding components of the momentum of the wall, obtaining

$$f_i = f_{-i} + 2\rho_0 \cdot W_i \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}_{bc}}{c_s^2}. \tag{4.2}$$

The implementation of (4.2) is simple, as example we give the resulting equations for the D2Q9 model in case of a south wall aligned with the x-axis:

$$\begin{aligned} f_2 &= f_4 + \frac{2}{3}\frac{u_x}{c} \\ f_5 &= f_7 + \frac{1}{6}\frac{u_x}{c} + \frac{1}{6}\frac{u_y}{c} \\ f_6 &= f_8 - \frac{1}{6}\frac{u_x}{c} + \frac{1}{6}\frac{u_y}{c} \end{aligned}$$

4.2.4. ZOU HE Scheme

Zou and He [74] introduced a new bounce-back scheme which is based on the idea of 'applying the bounce-back rule to off-equilibrium parts'. From the assumption

$$f_i + f_i^{eq} = f_{-i} + f_{-i}^{eq}$$

one can compute a scheme which differs from Ladd's conditions especially in the case of a slip boundary. We present the modified scheme again for a south wall:

$$\begin{aligned} f_2 &= f_4 + \frac{2}{3}\frac{u_x}{c} \\ f_5 &= f_7 - \frac{1}{2}(f_1 - f_3) + \frac{1}{3}\frac{u_x}{c} + \frac{1}{6}\frac{u_x}{c} + \frac{1}{6}\frac{u_y}{c} \\ f_6 &= f_8 + \frac{1}{2}(f_1 - f_3) - \frac{1}{3}u_x - \frac{1}{6}\frac{u_x}{c} + \frac{1}{6}\frac{u_y}{c} \end{aligned}$$

In practice it means that the macroscopic slip-condition is enforced by taking into account the distributions aligned with the wall and an additional term of the wall velocity.

4.3. Special cases of boundary conditions

There are certain cases where the boundary conditions need special attention, for example, how to implement boundary conditions in case if there exist singular points in the computational domain, how to treat the Neumann boundary part. We discuss these cases in the following sections. At the end, we will show how to implement boundary conditions directly into the system matrix for an implicit treatment.

4.3.1. Singular points

There are some situations for which we have special boundary nodes that are not by definition in the set of Γ^-, that is we cannot apply the scheme described above, therefore we have to introduce a special treatment. In Fig. 4.1a we have a pair of opposing distributions exactly in the corner enclosing the flow domain, which cannot be treated by the bounce-back equation (4.5), nor by the upwind-discretisation of the LBE, because in this case both f_i and f_{-i} are facing out of the domain and are located in a singular point. In addition to this case, the grid processing routines which are part of FEATFLOW [67] define some points at concave walls (secant running through solid) as incoming boundary values, for example the north and south distributions in Fig. 4.1b). In both cases, it is out of question to neglect these values because we need the complete sum of 9 distributions to obtain the local density, resp., pressure. We found and implemented two consistent ways how to treat special boundaries. First, assuming a no-slip boundary with $\mathbf{u} = 0$, we insert the moments $\mathbf{u}$ and ρ into the equilibrium term, resulting in

$$f_i^{eq}(\rho, 0) = W_i \rho. \tag{4.3}$$

Identifying the distribution f_i with its equilibrium, which is a common scheme for boundary-conditions, we can account for the missing equation by

$$f_i := f_i^{eq}(\rho, 0) = W_i \sum f_k.$$

We can derive the opposing distribution in an analoguous way, or just identify $f_{-i} := f_i$ according to Eq. (4.5) for the considered no-slip case where opposing contributions must cancel out.

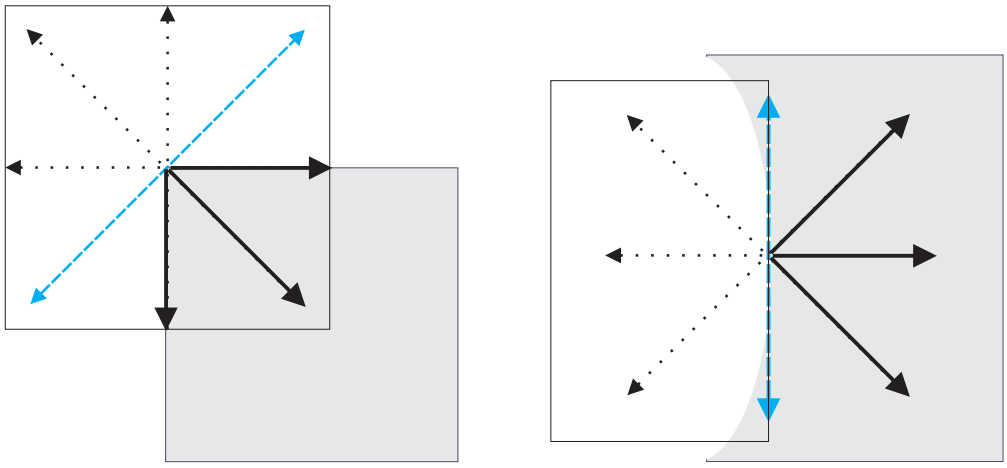
4.3.2. Boundary by extrapolation

The second way to treat exceptional configurations is especially suited for the case of a slip boundary with $\mathbf{u} \neq 0$, where equation (4.3) does not hold and we want to compute values of f_i and f_{-i} in the corner with a nonzero wall-velocity. In this case we want to derive the unknowns by extrapolation and to this purpose use our finite difference discretization. We exploit upwind information of an adequate characteristic and use known values from inside of the domain in the constant extrapolation scheme

$$f_4(v_0) = f_4(v_1) = \lambda_1 f_4(p_1^1) + \lambda_m f_4(m_1) + \lambda_2 f_4(p_1^2), \tag{4.4}$$

or the linear extrapolation

$$f_4(v_0) = (1 + \frac{1}{\alpha - 1}) f_4(v_1) - \frac{1}{\alpha - 1} f_4(v_2).$$



(a) Treatment in corners (b) Concave case

Figure 4.1: Special boundary configurations

4.3.3. Implementation of boundary conditions into the system-matrix

In this work we treat the boundary conditions implicitly and we directly implement them into the system matrix. There are two main reasons for this treatment. First, in the case of the direct stationary solution an implicit treatment is necessary for the efficient solution of the nonlinear equations and second, for nonstationary flow it improves stability when using large time steps. Most of the results in this thesis are given for Ladd's scheme and in our monolithic approach we include the boundary conditions into our matrix in a fully implicit way as

$$f_i - f_{-i} = 2\rho_0 \cdot W_i \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}_{bc}}{c_s^2}. \tag{4.5}$$

The components of velocity $\mathbf{u}_{bc}$ appear in the right hand side of our algebraic system. However, it is possible to use an explicit boundary scheme in the nonstationary model of Eq. (3.2). Then the bounce back contribution of the outgoing f_{-i} is taken from the last time step, and appears in the right hand side:

$$f_i^{n+1} = f_{-i}^{n} + 2\rho_0 \cdot W_i \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}_{bc}}{c_s^2} \tag{4.6}$$

4.3.4. Neumann boundary

The final configuration to be discussed is the Neumann boundary conditions used for example for the outflow of the channel in the *flow around cylinder* benchmark [58]. To satisfy the natural condition $\frac{\partial \mathbf{u}}{\partial n} = 0$, we use the constant extrapolation given in equation (4.4), following the characteristic pointing in the outward direction n. We apply this scheme on all distributions $f_k, k \neq 0$ except the rest particles. This corresponds to the treatment in the standard Lattice Boltzmann method, where the values are 'copied' from the last layer before the boundary in the structured mesh.

5

Nonlinear and Linear Solvers

5.1. Nonlinear solvers

The system of algebraic equations derived in Chapter 3 is nonlinear due to the presence of the terms $u_\alpha u_\beta$ in the equilibrium distribution function occuring in the collision operator which implies a quadratic nonlinearity $f_i f_j$ in the primary solution variables. After using the linearization given in section 3.4 it is possible to update $\tilde{f} = f^n$ only once every step. Then we have multiple possibilities, for example to apply 2nd order extrapolation $\tilde{f} = 2f^n - f^{n-1}$ in a (pseudo) time-stepping with very small Δt which yields a semi-implicit scheme. The stability and accuracy of such a scheme would be poor and a fully stationary approach is even impossible. A better approach in view of stability would be to iterate $\tilde{f}$ until a fixed point is reached. However, much more efficient would be to use Newton's method to obtain full control of nonlinear defect.

5.1.1. Fixed point iteration

After the space and time discretization introduced in Chapter 3 , we get a non-linear coupled algebraic system. We write the non-linear system in the form

$$\mathcal{N}(\mathbf{x})\mathbf{x} = \mathbf{g} \tag{5.1}$$

with $\mathbf{x}$ representing the solution vector for the distributions f_i and $\mathcal{N}(x)$ the full operator consisting of discrete transport and collision. Equation (5.1) can be solved by simple fixed-point iteration

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \sigma\mathcal{N}(\mathbf{x}^n)^{-1}\Big[\mathcal{N}(\mathbf{x}^n)\mathbf{x}^n - \mathbf{g}\Big] \quad , \quad \sigma > 0 \tag{5.2}$$

with optional damping by a factor σ. The results given in Chapter 6 show that the convergence of this basic scheme is too poor for large time steps and in case of strong nonlinearities appropriate damping is recommended, but the convergence can be easily and significantly improved by a Newton scheme described in the next section.

5.1.2. Newton method

As alternative to the fixed-point iteration we present Newton's scheme, (for details see Appendix B) which is well known for its quadratic convergence behaviour. To start the procedure we write system (5.1) in residual form

$$\mathcal{R}(\mathbf{x}) = \mathcal{N}(\mathbf{x})\mathbf{x} - \mathbf{g} = \mathbf{0}. \tag{5.3}$$

The Jacobian $\left[\frac{\partial \mathcal{R}(\mathbf{x}^n)}{\partial \mathbf{x}}\right]$ is then used in the following iterative scheme:

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \left[\frac{\partial \mathcal{R}(\mathbf{x}^n)}{\partial \mathbf{x}}\right]^{-1} \mathcal{R}(\mathbf{x}^n) \tag{5.4}$$

The computation of Jacobian matrix usually represents a major task in the implementation of Newton's method. In most cases the Jacobian matrix is not available in analytic form so it is, therefore, approximated by using a finite difference quotient but fortunately, in our case the Jacobian can be obtained analytically. In short, summing up the partial derivatives $\frac{\partial f_i^{eq}}{\partial f_k}$ for all k and linearization yields:

$$\begin{aligned} df_i^{eq} &= W_i\Big(\sum_k f_k + 3\sum_k D_{ik} f_k + \frac{9}{c^2}\sum_k D_{ik} f_k(\boldsymbol{\xi}_i \cdot \tilde{\mathbf{u}}) - \frac{3}{c}\Big(\sum_k D_{1k} f_k \tilde{u}_1 + \sum_k D_{2k} f_k \tilde{u}_2\Big)\Big) \\ &=: \sum_k \bar{\omega}_{ik} f_k \end{aligned} \tag{5.5}$$

So, the derivation simply results in a scaling by 2 in each quadratic term and gives us the linearized collision entries $\bar{\omega}_{ik}$ of the Jacobian, quite similar to $\tilde{\omega}_{ik}$ in Eq. (3.14). The remaining terms of the Jacobian are trivial, they include the constant coefficients due to the transport difference quotient or possible identity terms from the time-stepping variants. The Jacobian used to solve the steady state equation is therefore given by

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^n)}{\partial \mathbf{x}}\right] = \left(\begin{bmatrix} \mathbf{T}_0 & & & & \\ & \mathbf{T}_1 & & & \\ & & \mathbf{T}_2 & & \\ & & & \ddots & \\ & & & & \mathbf{T}_8 \end{bmatrix} - \frac{1}{\tau} \begin{bmatrix} \bar{\omega}_{00} & \bar{\omega}_{01} & \bar{\omega}_{02} & \cdots & \bar{\omega}_{08} \\ \bar{\omega}_{10} & \bar{\omega}_{11} & \bar{\omega}_{21} & & \\ \bar{\omega}_{20} & \bar{\omega}_{21} & \bar{\omega}_{22} & & \vdots \\ \vdots & & & \ddots & \\ \bar{\omega}_{80} & & \cdots & & \bar{\omega}_{88} \end{bmatrix} \right)$$

while in an analoguous way the GEF monolithic approach results in the Jacobian

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^n)}{\partial \mathbf{x}}\right]^{GEF} = \left(\mathbf{I} - \begin{bmatrix} \bar{\omega}_{00}\frac{1}{\tau}\mathbf{T}_0^{-1} & \bar{\omega}_{01}\frac{1}{\tau}\mathbf{T}_1^{-1} & \cdots & \bar{\omega}_{08}\frac{1}{\tau}\mathbf{T}_8^{-1} \\ \bar{\omega}_{10}\frac{1}{\tau}\mathbf{T}_0^{-1} & \bar{\omega}_{11}\frac{1}{\tau}\mathbf{T}_1^{-1} & & \vdots \\ \vdots & & \ddots & \vdots \\ \bar{\omega}_{80}\frac{1}{\tau}\mathbf{T}_0^{-1} & \cdots & \cdots & \bar{\omega}_{88}\frac{1}{\tau}\mathbf{T}_8^{-1} \end{bmatrix} \right).$$

The time stepping variant of the Jacobian is given by

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^l)}{\partial \mathbf{x}}\right] = \left(\begin{bmatrix} \mathbf{T}_0 & & & \\ & \mathbf{T}_1 & & \\ & & \ddots & \\ & & & \mathbf{T}_8 \end{bmatrix} - \frac{\theta \Delta t}{\tau} \begin{bmatrix} \bar{\omega}_{00} & \bar{\omega}_{01} & \dots & \bar{\omega}_{08} \\ \bar{\omega}_{10} & \bar{\omega}_{11} & & \vdots \\ \vdots & & \ddots & \vdots \\ \bar{\omega}_{80} & \dots & \dots & \bar{\omega}_{88} \end{bmatrix} \right)$$

Similarly, an improved condition number is obtained by including the inverse transport operator into the system with the GEF approach

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^l)}{\partial \mathbf{x}}\right]^{GEF} = \left(\mathbf{I} - \frac{\theta \Delta t}{\tau} \begin{bmatrix} \bar{\omega}_{00}\mathbf{T}_0^{-1} & \bar{\omega}_{01}\mathbf{T}_1^{-1} & \cdots & \bar{\omega}_{08}\mathbf{T}_8^{-1} \\ \bar{\omega}_{10}\mathbf{T}_0^{-1} & \bar{\omega}_{11}\mathbf{T}_1^{-1} & & \vdots \\ \vdots & & \ddots & \vdots \\ \bar{\omega}_{80}\mathbf{T}_0^{-1} & \cdots & \cdots & \bar{\omega}_{88}\mathbf{T}_8^{-1} \end{bmatrix} \right).$$

We summarize the whole procedure, now combining the algebraic reformulation (GEF) with the Newton iteration in Algorithm 5.1.

Algorithm 5.1 The Newton method with GEF approach

1) In every time-step, the nonlinear system for the distributions f_i^{n+1} reads

$$\mathbf{T}_i f_i^{n+1} - \frac{\theta \Delta t}{\tau} \sum_k w_{ik} f_k^{n+1} = g_i^n \quad i = 0, ..., 8.$$

2) In each nonlinear iteration, perform a Newton iteration for $l = 0, 1, \ldots$ with the resulting linear block-system

$$\mathbf{x}^{l+1} = \mathbf{x}^l - \left[\frac{\partial \mathcal{R}(\mathbf{x}^l)}{\partial \mathbf{x}}\right]^{-1} \mathcal{R}(\mathbf{x}^l)$$

3) Solve the linear system

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^l)}{\partial \mathbf{x}}\right] \Delta \mathbf{x} = -\mathcal{R}(\mathbf{x}^l)$$

with the resulting linear block-system

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^l)}{\partial \mathbf{x}}\right] = \left(\begin{bmatrix} \mathbf{T}_0 & & & \\ & \mathbf{T}_1 & & \\ & & \ddots & \\ & & & \mathbf{T}_8 \end{bmatrix} - \frac{\theta \Delta t}{\tau} \begin{bmatrix} \bar{\omega}_{00} & \bar{\omega}_{01} & \ldots & \bar{\omega}_{08} \\ \bar{\omega}_{10} & \bar{\omega}_{11} & & \vdots \\ \vdots & & \ddots & \vdots \\ \bar{\omega}_{80} & \ldots & \ldots & \bar{\omega}_{88} \end{bmatrix} \right).$$

or with an improved condition number by including the inverse transport operator into the system with the GEF approach

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^l)}{\partial \mathbf{x}}\right]^{GEF} = \left(\mathbf{I} - \frac{\theta \Delta t}{\tau} \begin{bmatrix} \bar{\omega}_{00}\mathbf{T}_0^{-1} & \bar{\omega}_{01}\mathbf{T}_1^{-1} & \cdots & \bar{\omega}_{08}\mathbf{T}_8^{-1} \\ \bar{\omega}_{10}\mathbf{T}_0^{-1} & \bar{\omega}_{11}\mathbf{T}_1^{-1} & & \vdots \\ \vdots & & \ddots & \vdots \\ \bar{\omega}_{80}\mathbf{T}_0^{-1} & \cdots & \cdots & \bar{\omega}_{88}\mathbf{T}_8^{-1} \end{bmatrix} \right).$$

4) Update the solution

$$\mathbf{x}^{l+1} = \mathbf{x}^l + \sigma \Delta \mathbf{x}^{(l)}$$

where $\sigma > 0$ is the damping parameter.

5.2. Solution of Linear Problems

After having done with the linearization of operator $\mathcal{N}(\mathbf{x})$, we obtain a linear system matrix A corresponding to the time and space discretization. We need to solve a linear system $Ax = b$ (x representing the discrete solution vector of all f_i) in each non-linear step and it is the most time consuming part of a simulation process. Beforehand, let us look at the obtained structure of the linear system to have a basic understanding of the problem. The main diagonal of the system matrix can be determined by two basic numbering techniques. In the first case which is standard, listing for every direction the spatial nodes just as they were identified by the grid generator, gives nine so-called 'transport blocks' each of size $n \times n$. The solution vector is then given by

$$\underbrace{f_1(x_1),\ldots,f_1(x_n)}_{\mathbf{f_1}},\underbrace{\ldots}_{\mathbf{f_i}},\underbrace{f_9(x_1),\ldots,f_9(x_n)}_{\mathbf{f_9}}.$$

In the second case, if we use the local unknowns as the first index, we obtain a narrower block-diagonal of 9×9 blocks, with the solution vector enumerated as

$$\underbrace{f_1(x_1),\ldots,f_9(x_1)}_{\mathbf{f}(\mathbf{x_1})},\underbrace{\ldots}_{\mathbf{f}(\mathbf{x_i})},\underbrace{f_1(x_n),\ldots,f_9(x_n)}_{\mathbf{f}(\mathbf{x_n})}.$$

In both cases we see additional entries outside of the block-diagonal, otherwise the solution of the system would be trivial. All in all, the matrix allocation is dependent on the node numbering (for example to obtain lower triangular matrices as described in Section 3.3, see Fig. 3.3) and on the primary index being either the spatial or directional variable. In detail, the collisions cause in every grid-point a coupling of the distributions. For each direction we have 9 matrix-entries on the off-diagonals, it means a 9 by 9 collision block for the distributions in one grid point. If boundary conditions are treated implicitly, the incoming distribution f_i is coupled to the outgoing f_{-i} which can be regarded as a small collision. In contrast to this example of *local* coupling, nodes from *neighbouring* nodes are mainly coupled through the transport term, in this case we talk about distributions with the same constant characteristic. Altogether, the resulting matrix allocation is quite sparse.
Linear solvers are generally divided into two broad categories, the direct solvers and iterative solvers. The choice of the solution method is largely independent of the underlying discretization techniques but the size and structure of the matrix A need to be taken into account.

5.2.1. Direct Methods

Direct methods for solving linear systems accomplish the task in one step. The input parameters are the matrix A and the right-hand side b. The result is the solution vector x. The solution obtained by direct methods is exact upto the machine accuracy. The most commonly used direct methods are the Gaussian Elimination and LU decomposition. The complexity of these methods is $\mathcal{O}(n^3)$ [60, 66], where n is the number of degrees of freedom associated with matrix A. Another direct method is the Cholesky factorization for symmetric positive definite matrices (SPD) which also has the same complexity as Gaussian Elimination. Direct methods are often used for small dense problems and become

prohibitively expensive for large problems due to their asymptotically cubic runtime. Although the associated coefficient matrices are sparse but unfortunately their structure is such that after a few steps of Gaussian elimination, the computational effort grows significantly because most of the zero elements are replaced by nonzero ones (see [7] for more details). Moreover, one major disadvantage of these methods is a need to form an explicit matrix which in practice requires a lot of storage.

5.2.2. Iterative Methods

Iterative methods solve linear systems using a sequence of explicit updates starting with an initial guess which must be supplied as another input parameter. Such an algorithm is said to be convergent if each update brings the solution closer to that exact solution.Iterative methods use a rather small amount of computer memory. Many of them do not even require that the matrix A is available. All they need is a subroutine that evaluates the residual of the linear system for a given tentative solution.

5.2.3. Stationary iterative solvers

The basic iterative methods used for solving large linear systems $Ax = b$, where A is a given matrix and b is a given vector, are the stationary iterative solvers. Stationary iterative methods are those which can be expressed in the simple form.

$$x^{(k)} = Bx^{(k-1)} + c$$

(where neither the matrix B nor the vector c depend upon the iteration count k) [3]. These iterative methods are based on relaxation of the coordinates. Starting with an initial guess, these methods modify the components of the approximation, one or a few at a time and in a certain order, until the convergence is reached. Nowadays these methods are rarely used as stand-alone iterative solvers due to the poor convergence. Nevertheless, they have not lost their importance. Indeed they are often used in conjunction with modern efficient iterative methods, for example in the Krylov space method for preconditioning, and in the multigrid method for smoothing. The four main stationary methods are the following:

- Jacobi method (JAC),
- Gauss-Seidel method (GS),
- Successive overrelaxation method (SOR),
- Symmetric successive overrelaxation method (SSOR).

All these methods can be formulated as a *defect-correction* as follows

$$x^{(k)} = x^{(k-1)} + \omega P^{-1}(b - Ax^{(k-1)}),$$

where P is matrix for which we classify for $A = L + D + R$:

$$\begin{aligned}
&\text{JAC:} && P = D \\
&\text{GS:} && P = D + L \\
&\text{SOR:} && P = D + \omega L, \qquad 0 < \omega < 2 \\
&\text{SSOR :} && P = (D + \omega L)D^{-1}(D + \omega U).
\end{aligned}$$

Here, L, D, R denote the lower triangular, diagonal and upper triangular part of the matrix A.

5.2.4. Nonstationary iterative solvers

Nonstationary iterative methods are based on computations which involve information that changes at each iteration k. A well known class in this category is the class of Krylov space methods, i.e., methods that seek to generate better approximations from the Krylov subspace. These methods find an approximation x_k to the linear system $Ax = b$ over the m-th Krylov subspace $K_m(A, r_0) + x_0$, where x_0 is the initial solution and r_0 represents the initial residual vector, given by $r_0 = b - Ax_0$. A Krylov subspace of dimension $m \in \mathbb{N}$ generated by a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $r \in \mathbb{R}^n$ is the subspace spanned by the vectors of the Krylov sequence:

$$K_m(A, r) = \text{span}\left\{r, Ar, A^2r, \cdots, A^{m-1}r\right\}.$$

The dimension of this subspace increases by one for every step of the approximation process. These methods are extensively used nowadays for the solution of linear systems arising from the discretization of partial differential equations (PDEs). Commonly known Krylov space methods are

- Conjugate Gradient (CG),
- Generalized Minimal Residual (GMRES),
- Biconjugate Gradient (BiCG),
- Quasi-Minimal Residual (QMR),
- Conjugate Gradient Squared (CGS),
- Biconjugate Gradient Stabilized (BiCGStab),

A comprehensive study regarding the Krylov space methods can be found in [3].

Bi-Conjugate Gradient Stabilised Method (BiCGStab)

The Bi-Conjugate Gradient Stabilised method was originally developed by van der Vorst [70] in 1992 from the CGS, BiCG and GMRES methods, for the solution of non-symmetric linear system of equations while avoiding the often highly irregular convergence behavior of CGS and BiCG, and the large storage requirements of GMRES. This algorithm requires six auxiliary vectors, and performs two preconditioning steps in each iteration. Furthermore, the two matrix-vector product are also perform in each BiCGStab step. For many problems, the BiCGStab algorithm converges rather smoothly and also often faster than BiCG and CGS. Like other iterative methods, BiCGStab method is usually combined with a preconditioning to speed up its convergence. We apply the standard algorithm for the preconditioned BiCGStab method in our numerical study (see [70]). The pseudocode for the Preconditioned BiCGStab is given in Algorithm 5.2.

Algorithm 5.2 The Preconditioned BiConjugate Gradient Stabilized Method (BiCGStab).

Choose $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
Choose r^* (for example, $r^* = r^{(0)}$)
for $i = 1, 2, \ldots$
$\rho_{i-1} = r^{*T} r^{i-1}$
if $\rho_{i-1} = 0$ method fails
if $i = 1$
$p^{(i)} = r^{i-1}$
else
$\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
$p^{(i)} = r^{i-1} + \beta_{i-1}(p^{(i-1)} - \omega_{i-1} v_{i-1})$
endif
Solve $M\hat{p} = p^{(i)}$
$v_i = A\hat{p}$
$\alpha_i = \rho_{i-1}/r^{*T} v_i$
$s = r_{i-1} - \alpha_i v_i$
check norm of s; if small enough: set $x^{(i)} = x^{(i-1)} + \alpha_i \hat{p}$ and stop
Solve $M\hat{s} = s$
$t = A\hat{s}$
$\omega_i = t^T s / t^T t$
$x^{(i)} = x^{(i-1)} + \alpha_i \hat{p} + \omega_i \hat{s}$
$r^{(i)} = s - \omega_i t$
check convergence; continue if necessary
for continuation it is necessary that $\omega_i \neq 0$
end

Generalized Minimal Residual Method (GMRES)

The Generalized Minimal Residual Method (GMRES) was proposed by Saad and Schultz [57] in 1986 to find the solution of non-symmetric linear systems. This method is an extension of the minimal residual method (MINRES) which is only applicable for the solution of symmetric systems. In the GMRES method, a sequence of orthogonal vectors is generated and all the previously computed vectors in the orthogonal sequence are required to calculate the next iteration. Consequently, the required storage for these vectors often exceeds the available memory limit before reaching the stopping criterion which is the drawback of this method. To remedy this difficulty, a restarted version of this method, usually referred by GMRES(m) is commonly used where the algorithm is restarted in every m steps for fixed integer m. The GMRES method has an advantage over BiCGStab method due to its convergence behavior since it is more stable in practice for ill-conditioned systems. Moreover, only one matrix-vector multiplication is performed in the GMRES while the BiCGStab requires two matrix-vector products. The pseudocode for the Preconditioned GMRES(m) is given in Algorithm 5.3.

Algorithm 5.3 The Preconditioned Generalized Minimal Residual Method (GMRES(m)).

Choose the initial guess $x^{(0)}$
for $j = 1, 2, \ldots$
Solve r form $Mr = b - Ax^{(0)}$
$v^{(1)} = r/\|r\|_2$
$s := \|r\|_2 e_1$
for $i = 1, 2, \ldots, m$
Solve w form $Mw = Av^{(i)}$
for $k = 1, \ldots, i$
$h_{k,i} = (w, v^{(k)})$
$w = w - h_{k,i} - v^{(k)}$
end $h_{i+1,i} = \|w\|_2$
$v^{(i+1)} = w/h_{i+1,i}$
apply $J_1, \ldots, J_{i-1}$on$(h_{1,i}, ..., h_{i+1}, i)$
construct J_i, acting on ith and $(i+1)$st component
of $h., i$, such that $(i+1)$st component of $J_i h_{.,i}$ is 0
$s := J_i s$
if $s(i+1)$ is small enough then (UPDATE$(\tilde{x}, i)$ and quit)
end
UPDATE$(\tilde{x}, i)$
end
end

5.3. Multigrid method

Multigrid methods are well known for being the fastest numerical methods to solve linear systems arising from the discretization of PDEs. These methods are generally applied for problems showing level-dependent convergence behaviour. In contrast to other iterative methods, the convergence rate of multigrid methods is usually independent of the mesh level. A multigrid method needs a hierarchy of grids and appropriate grid transfer operators which are responsible for data communication between the grids. To convert a coarse grid solution to fine grid solution, a prolongation operator is used while a restriction operator is used for the opposite transfer. Beside these prolongation and restriction operators, we need to use an efficient coarse grid solver and choose an iterative solution method as smoother. We present a standard two-grid algorithm in Algorithm 5.4 which is used mainly in our tests. A two grid algorithm starts at the fine level with s_1 number of pre-smoothing steps, performs a correction by solving a coarse grid auxiliary problem and ends with s_2 number of post-smoothing steps. In practice, we iterated the algorithm with a certain number $s = s_1 + s_2$ of smoothing steps until the linear defect is sufficiently reduced. A full multigrid algorithm is obtained when the routine is called recursively in the correction step, and the inverse matrix is applied only on the coarsest level. Such V-cycle can be further altered by modifying the number of defect-correction calls on each level. However, here we would like to mention that better results were obtained using 'multigrid as preconditioner', i.e. performing one coarse-grid correction gaining several digits of accuracy in every GMRES iteration.

Algorithm 5.4 Basic twogrid algorithm

TG $(\mathbf{x}, \mathbf{b}, l)$
auxiliary vectors $\mathbf{d}, \mathbf{v}$

pre-smoothing	$\mathbf{x}$	$=$	$S_l^{s_1}(\mathbf{x}, \mathbf{b})$
compute defect	$\mathbf{d}_l$	$=$	$A_l\mathbf{x} - \mathbf{b}$
restriction	$\mathbf{d}_{l-1}$	$=$	$R\mathbf{d}_l$
correction	$\mathbf{v}_{l-1}$	$=$	$A_{l-1}^{-1}(\mathbf{d}_{l-1})$
prolongation	$\mathbf{v}_l$	$=$	$P\mathbf{v}_{l-1}$
assemble solution	$\mathbf{x}$	$=$	$\mathbf{x} - \mathbf{v}_l$
post-smoothing	$\mathbf{x}$	$=$	$S_l^{s_2}(\mathbf{x}, \mathbf{b})$

5.3.1. Prolongation and Restriction

One of the main ingredients of multigrid are grid transfer operators which are responsible for prolongation and restriction. The prolongation is performed elementwise, whereas an efficient implementation of the restriction is more difficult. For the restriction operator, which is the transposed to the prolongation, values from a node need to collect information from neighboring elements. In unstructured meshes, the number of elements meeting in one point is arbitrary, while it amounts to 4 in the case of a uniform rectangular mesh, resp., 6 elements in the case of a uniform triangular mesh in 2D case. The details about the prolongation and restriction operators used in this thesis can be found in [33].

5.4. Preconditioning Techniques

The convergence of the iterative methods is mainly dependent on the spectral properties of the system matrix A since the rate of convergence for most iterative linear solvers degrades as the condition number of the matrix A increases. Preconditioning is a technique by which the condition number of the system matrix is improved to speed up the convergence rate of the iterative methods. It is an important ingredient behind the success of iterative methods [56]. In the subject of numerical analysis, a preconditioner P for a matrix A can be chosen in a way such that $P^{-1}A$ has a smaller condition number than A. Rather than solving the original system, we solve the system $Ax = b$ indirectly by solving

$$P^{-1}Ax = P^{-1}b.$$

One may apply the preconditioner P either from the left or right. For a symmetric positive definite matrix A, it is also suggested that the preconditioner P should be symmetric positive definite.
There is always a trade-off between the cost of applying P^{-1} and the improvement of the convergence properties. The two extreme cases corresponds to the choices $P = I$ and $P = A$. If we choose $P = I$ then this is 'do nothing' case since P^{-1} is also identity matrix and we are solving the original system and hence no preconditioning of the system matrix at all. If we choose

$$P = A,$$

in this case we have perfect preconditioning as the condition number becomes one and the solution of the system is reached only in one iteration. However, it is not much useful as calculating the inverse of the matrix A is not so easy. Thus only parts of the matrix A are taken while developing preconditioners as described in the next section.

5.4.1. Commonly used Preconditioners

In actual practice, the preconditioner P is chosen depending on the properties of the system matrix A and it serves as a key component of the iterative solution method. A few general comments concerning the design of a preconditioner are:

- $P^{-1}A$ should be close to the identity matrix or at least have a spectrum that is clustered.
- The operation $x \mapsto P^{-1}x$ should be cheap.

The most common used preconditioners [3, 56] are

Jacobi preconditioner:	$P = D$
Gauss-Seidel preconditioner:	$P = D + L$
SOR preconditioner:	$P = D + \omega L, \qquad 0 < \omega < 2$
SSOR preconditioner :	$P = (D + \omega L)D^{-1}(D + \omega U).$

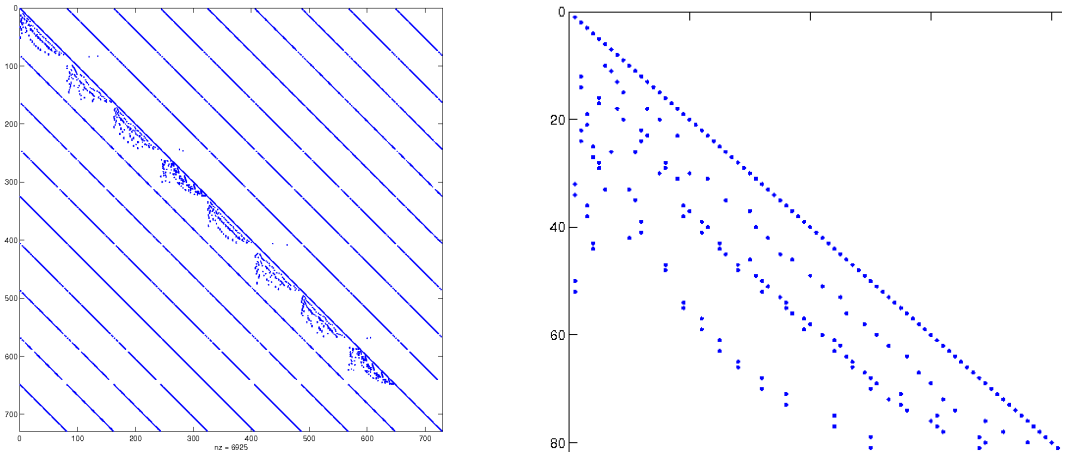
Here, L, D, R denote the lower triangular, diagonal and upper triangular part of the matrix A.

5.5. Special Preconditioners for LBE

In this thesis, mainly we used two Krylov space solvers namely the BiCG-stab and GMRES method. In fact both of them are rarely used as standalone solvers. In practice, they are always applied in collaboration with some preconditioner. Therefore, we have designed some special preconditioners for these Krylov space solvers to meet the ill conditioning of our linear subproblems. We describe these preconditioners in the following subsections.

5.5.1. Transport Preconditioner(tr-pre)

For transport dominated cases, it is useful to choose the convection part of the matrix as a preconditioner. In this case we improve condition number of $P^{-1}A$ significantly and independently of the mesh size as seen from the eigenvalue analysis given in [33]. Here we can directly invert P consisting of lower triangular blocks-taking for every direction the prepared numbering as shown in Fig. 5.1. This results in a linear run time in the number of unknowns. But in case of stiff configurations due to low Mach number, 'tr-pre' is not useful, so we give an alternative approach in the next section.



(a) special characteristic sorting (b) transport-block for one direction

Figure 5.1: Preconditioning by transport part

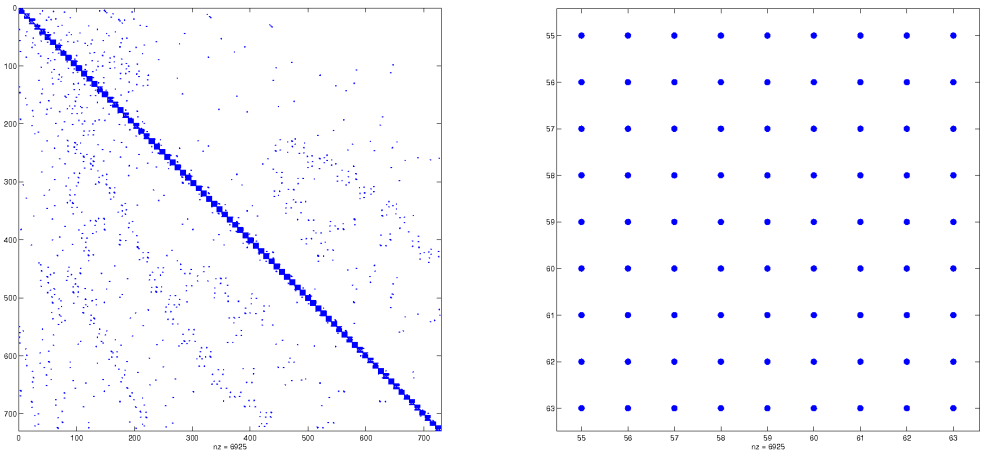
5.5.2. Block-diagonal collision preconditioner(bl-jac)

In case of large values of c, we are in a collision dominated regime and our linear system has a very bad condition number. In this situation 'tr-pre' is not suitable. As an alternative we construct another preconditioner that takes into account the collision term. We implemented another numbering of the unknowns taking the coordinate as primary index instead of the direction. We start the numbering with the 9 characteristics f_i, $i = 1,\ldots,9$ for the first point x_1 and continue in this way with 9 successive values each for all remaining points. Consequently, the local collisions form a diagonal consisting of 9×9 blocks. At the same time, the entries for the spacial coupling due to transport are freely scattered off the block-diagonal (see Fig. 5.2). The solution process of the preconditioner (to be de-

noted as 'bl-jac') is accomplished efficiently, as we need to invert therein a block-Jacobian matrix C. The inverse C^{-1} consist of the inverses of n decoupled matrices), i.e.

$$C^{-1} = diag(C_1, C_2, \ldots, C_n)^{-1} = diag(C_1^{-1}, C_2^{-1}, \ldots, C_n^{-1}) \qquad , \qquad C_i \in R^{9\times 9}$$

This inverse can be calculated by using Gaussian elimination of each 9×9 system. The overall runtime, even using this plain direct solver, is of the order $O(n)$ due to the constant number of operations to invert a matrix with 81 entries. Unfortunately, bl-jac does not longer contain information about the transport part of the Boltzmann equation, we lost it in giving up the special numbering and consequently the lower triangular form of the transport blocks. This means that even we can expect good results for small systems with $c \to \infty$, the convergence will be strongly dependent on the level, resp., number of unknowns. The following, final section concerning preconditioning will show a technique which combines the advantages of both numbering techniques.



(a) local collision sort (b) one 9×9 block

Figure 5.2: Block-Jacobian preconditioning by collision part

5.5.3. Special preconditioning of the GEF system matrix

As described in Section 3.6, we see that the inverse transport becomes a part of the generalized equilibrium formulation, and the special numbering following the characteristics gives a significant gain applying the implicit system matrix. Most importantly, in contrast to the original system which solves for the distributions f_i, the new formulation allows additional preconditioning.

Additional preconditioning by collisions

The GEF matrix consists mainly of weighted inverses of transport matrices, easily to be solved due to their lower triangular form. For the same reason we can apply additional preconditioning to the system and therefore enhance the GEF approach, by taking advantage of this triangular matrices to construct a preconditioner. The system is composed of blocks

$$G_{ik} = I - \tilde{\omega}_{ik} \frac{1}{\tau} T_k^{-1} \in R^{n\times n}$$

with I the corresponding identity matrix. So each G_{ik} contains information about the (possibly dominating) collisions mainly due to the relaxation rate $\frac{1}{\tau}$. Even though the overall structure of the G_{ik} is not known explicitly, we know $diag(G_{ik})$, they are easily obtained since $diag(T_i^{-1}) = diag(T_i)^{-1}$ with T_i lower triangular. We construct a local preconditioner (similar to the previous section) by collecting the 81 matrix entries per grid point and calculating the corresponding 9×9 inverse matrices. The full block-Jacobian preconditioner can be expressed as (or assembled into) the global matrix

$$C^{-1} = \begin{bmatrix} diag(G_{00}) & diag(G_{01}) & \cdots & diag(G_{08}) \\ diag(G_{10}) & diag(G_{11} & & \vdots \\ \vdots & & \ddots & \vdots \\ diag(G_{80}) & \cdots & \cdots & diag(G_{88}) \end{bmatrix}^{-1}$$

although the implementation is matrix-free. However, applying above C^{-1} in iterative solvers we expect an additional stabilizing effect in the case of large c, while due to the basic construction the solution of GEF performs well even without preconditioning.

6

Newtonian Results

This chapter presents results for the case of Newtonian incompressible fluids to guarantee the success of our implicit treatment of Lattice Boltzmann equation on unstructured grids and to show the efficiency of our non-linear and linear solvers. In order to obtain quantitative and qualitative results based on our discretization schemes we chose two different classical problems the *Driven Cavity* problem and the well known *Flow around Cylinder* benchmark [58].

6.1. Driven Cavity Problem

As a first test case we consider the *Driven Cavity* problem which has been extensively used as a benchmark problem to evaluate numerical techniques that solves the Navier-Stokes equations. The computational domain is a unit square with origin located at the lower left corner. A grid on this unit square is presented in figure **??**. Starting from a coarse mesh we get one level of refinement by connecting the edge midpoints which give rise to four new element from each element. To obtain the given macroscopic boundary conditions for velocity we use Ladd's scheme described in chapter 4. The left, right and bottom walls have no-slip conditions, while the flow is driven by a uniform translation of upper lid with $\mathbf{u} = \binom{1}{0}$. Due to these boundary conditions, no fluid can leave or enter the square domain but the flow turns around, with small separation zones along the boundary walls. The complexity of the problem depends on the speed of the lid, viscosity of the fluid and shape of the cavity.

The Reynolds number $Re = \frac{U_{ref} L_{ref}}{\nu}$ in our case is simplified to $Re = \frac{1}{\nu}$ so the only parameter responsible for the change in the *Re* number is ν. For the driven cavity problem and small up to moderate Re numbers, the flow is not time dependent. However, we consider this case to analyze our basic solvers and the influence of time-stepping on the convergence of linear, resp., nonlinear solvers. We performed simulations of the driven cavity problem at various Reynolds numbers with different start procedures. Our quantitative results are mainly based on this test case.

6.2. Solver Analysis

In this section we present a detailed solver analysis to compare the performance of our non-linear and linear solvers. To solve the coupled non-linear problems given in section 3.4, we used the Fixed point and Newton method in each time step. We performed simulations at different Reynolds numbers and with different start procedures. Regarding our linear solvers, we also discuss the advantages of our special preconditioners designed for transport dominated and collision dominated cases. We also discuss the GEF formulation which combines the advantages of both the preconditioners and with this approach we see a significant improvements in the convergence rates. We should keep in mind that small c means transport dominated equations, while large c means dominant collisions. However, c_{opt} seems to be in an intermediate range, where both effects have to be considered. Consequently, it is by far not trivial to solve systems with a larger number of unknowns.

	Mesh information			upw1	upw2
Level	Elements	Vertices	Midpoints	Total unknowns	Total unknowns
0	2	4	5	4	9
1	8	9	16	9	25
2	32	25	56	25	81
3	128	81	208	81	289
4	512	289	800	289	1 089
5	2 048	1 089	3 136	1 089	4 225
6	8 192	4 225	12 416	4 225	16 641
7	32 768	16 641	49 408	16 641	66 049

Table 6.1: *driven cavity* mesh information

6.2.1. Nonlinear Solver Analysis

In this section we present the results for the nonlinear solvers, comparing the performance of the fixed point iteration with the Newton method. A comparison is given in Table 6.2 for the Driven Cavity configuration with zero starting solution. Using different values of Δt, the simulations are run until a specified criterion for a steady state is reached. Despite of the low Reynolds number of 100, we need many fixed point iterations for a time step of $\Delta t = 1$, while the Newton results are very good. Smaller time steps, typically around $\Delta t = 0.001$ act as a damping of the nonlinearity and the fixed point scheme is improving significantly.

		$c = 1$		$c = 10$		$c = 100$	
Δt	Grid points	Newton	FP	Newton	FP	Newton	FP
	289	4	14	4	10	3	7
	1089	4	22	4	13	4	9
1	4225	4	28	4	22	4	10
	16641	4	28	5	26	5	10
	66049	6	39	7	32	6	26
	289	2	8	3	9	3	5
	1089	2	9	3	8	3	6
0.1	4225	2	10	3	8	3	6
	16641	3	12	4	10	4	8
	66049	4	13	5	12	5	10
	289	1	3	2	4	2	4
	1089	1	3	2	4	2	4
0.01	4225	1	3	2	5	2	4
	16641	2	5	2	5	3	5
	66049	3	5	4	6	5	7
	289	1	2	1	2	2	3
	1089	1	2	1	2	2	4
0.001	4225	1	2	1	3	2	4
	16641	2	3	2	4	2	4
	66049	2	3	3	4	3	5

Table 6.2: Driven Cavity: upw2, Averaged no. of non-linear iterations to reduce the non-linear defect by 10^{-6}, using Implicit Euler, $Re = 100$

6.2.2. Linear Solver Analysis

In this section, we compare the performance of used Krylov Space solvers BiCG-Stab and GMRES, in context of our special preconditioners. We continued the simulations until the criteria for steady state is reached and then we calculated the average number of linear iterations per time step. The results for BiCG-stab are given in the tables 6.3–6.4. It could be seen from these results that our preconditioners are highly adapted to the values of the sound parameter c. More linear iterations are needed for increasing c. It is also observed that it is easier to solve for smaller time steps due to the better conditioning of the system matrix. However, with BiCG-stab scheme, the convergence behaviour can fail unexpectedly, even for the moderate configuration with higher values of c. Therefore, we also implemented a GMRES solver which is well known for its monotone convergence behaviour. The corresponding results are given in tables 6.5–6.7.

		BiCGSTAB			
Δt	Grid points	tr-pre	bl-jac	GEF	GEF(\)
0.1	289	13	21	12	9
	1089	16	43	16	13
	4225	19	96	19	16
0.01	289	4	6	4	3
	1089	4	9	4	3
	4225	4	15	4	4
0.001	289	3	3	2	2
	1089	3	3	2	2
	4225	2	4	2	2

Table 6.3: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Implicit Euler, $Re = 100$, $c = 1$

		BiCGSTAB			
Δt	Grid points	tr-pre	bl-jac	GEF	GEF(\)
0.1	289	101	54	96	42
	1089	183	130	164	95
	4225	317	321	278	170
0.01	289	33	19	21	12
	1089	47	34	28	17
	4225	66	64	39	27
0.001	289	9	5	8	4
	1089	10	7	8	5
	4225	11	11	8	7

Table 6.4: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Implicit Euler, $Re = 100$, $c = 10$

Δt	Grid points	GMRES tr-pre	GMRES bl-jac	GMRES GEF	GMRES GEF(\)
0.1	289	23	39	21	17
	1089	28	76	27	23
	4225	31	157	31	29
0.01	289	8	11	7	5
	1089	8	16	7	6
	4225	8	27	7	6
0.001	289	4	5	4	3
	1089	4	6	4	3
	4225	4	7	4	3

Table 6.5: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Implicit Euler, $Re = 100$, $c = 1$

Δt	Grid points	GMRES tr-pre	GMRES bl-jac	GMRES GEF	GMRES GEF(\)
0.1	289	124	71	126	57
	1089	225	173	225	125
	4225	411	446	349	239
0.01	289	53	33	47	18
	1089	71	59	66	28
	4225	102	109	95	45
0.001	289	15	10	14	7
	1089	16	14	16	9
	4225	18	21	17	12

Table 6.6: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Implicit Euler, $Re = 100$, $c = 10$

Δt	Grid points	GMRES tr-pre	GMRES bl-jac	GMRES GEF	GMRES GEF(\)
0.1	289	290	108	346	110
	1089	784	383	645	389
	4225	831	652	912	866
0.01	289	195	69	226	62
	1089	373	174	444	149
	4225	626	415	726	325
0.001	289	81	29	63	23
	1089	137	53	97	42
	4225	215	107	161	86

Table 6.7: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Implicit Euler, $Re = 100$, $c = 100$

Δt	Grid points	BiCGSTAB tr-pre	BiCGSTAB bl-jac	BiCGSTAB GEF	BiCGSTAB GEF(\)
0.1	289	10	15	8	7
	1089	12	29	10	9
	4225	13	61	12	11
0.01	289	4	4	3	3
	1089	4	6	3	3
	4225	4	9	3	3
0.001	289	2	2	2	2
	1089	2	3	2	2
	4225	2	3	2	2

Table 6.8: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Crank Nicolson, $Re = 100$, $c = 1$

Δt	Grid points	BiCGSTAB tr-pre	BiCGSTAB bl-jac	BiCGSTAB GEF	BiCGSTAB GEF(\)
0.1	289	78	38	68	34
	1089	124	89	113	66
	4225	185	179	182	124
0.01	289	25	12	21	10
	1089	29	19	27	14
	4225	37	35	35	22
0.001	289	8	4	6	5
	1089	8	5	6	5
	4225	9	8	6	5

Table 6.9: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Crank Nicolson, $Re = 100$, $c = 10$

Δt	Grid points	GMRES tr-pre	GMRES bl-jac	GMRES GEF	GMRES GEF (\)
0.1	289	19	26	14	13
	1089	21	49	17	16
	4225	22	99	20	19
0.01	289	6	8	6	6
	1089	7	11	6	6
	4225	6	17	6	6
0.001	289	4	4	4	4
	1089	4	5	4	4
	4225	4	6	4	4

Table 6.10: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Crank Nicolson, $Re = 100$, $c = 1$

Δt	Grid points	GMRES tr-pre	bl-jac	GEF	GEF(\)
0.1	289	112	49	87	45
	1089	190	121	149	93
	4225	262	236	251	180
0.01	289	43	22	32	18
	1089	52	34	41	26
	4225	64	46	54	38
0.001	289	14	8	10	9
	1089	14	10	11	9
	4225	14	14	11	9

Table 6.11: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Crank Nicolson, $Re = 100$, $c = 10$

Δt	Grid points	GMRES tr-pre	bl-jac	GEF	GEF(\)
0.1	289	412	106	284	110
	1089	670	276	541	276
	4225	822	745	623	792
0.01	289	208	54	134	45
	1089	384	109	249	91
	4225	735	203	483	227
0.001	289	72	18	47	15
	1089	113	33	73	27
	4225	161	60	112	46

Table 6.12: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Crank Nicolson, $Re = 100$, $c = 100$

6.3. Solver Analysis with prolongation approach

In this section, we use the same configuration but we perform the solver analysis with a different start procedure. Here we use the idea of prolongation which means calculating the solution at some coarse level and using it as starting solution for a higher level. In practice it is easy to prolongate the solution obtained at some level 'l' and use it as starting solution for the level '$l+1$'. In this way the gained initial defect is non-zero, but in the first non-linear step it does not grow so much as compared with zero starting solution.

6.3.1. Non-linear Solver Analysis

In table 6.13, the converged solution is taken from the previous level by prolongation as starting guess. For the smaller Δt we performed the corresponding number of time steps to reach 0.5 seconds, showing the averaged number of nonlinear iterations for one timestep. We also compare with the full nonlinear sweep of the stationary solver ($\Delta t = \infty$) and one with $\Delta t = 1$, observing that the two configurations give quite similar results.

		$c=1$		$c=10$		$c=100$	
Δt	Grid points	Newton	FP	Newton	FP	Newton	FP
	1089	4	18	3	11	3	6
∞	4225	3	20	3	17	3	10
	16641	3	22	3	21	4	16
	66049	3	23	4	22	5	26
	1089	3	18	3	10	3	6
1	4225	3	19	3	15	3	9
	16641	3	20	3	19	4	14
	66049	3	20	4	20	4	19
	1089	2	10	3	7	3	5
0.1	4225	2	11	3	8	3	6
	16641	2	11	3	10	3	8
	66049	2	11	3	11	3	10
	1089	2	3	2	3	2	4
0.01	4225	2	4	2	4	2	4
	16641	2	4	2	4	2	4
	66049	2	4	2	4	2	4
	1089	1	2	1	2	1	2
0.001	4225	1	2	1	2	1	2
	16641	1	2	1	2	1	2
	66049	1	2	1	2	1	2

Table 6.13: Driven Cavity $Re = 100$: Absolute ($\Delta t = \infty$ and 1) and averaged No. of nonlinear iterations per time step to reduce the nonlinear defect by 10^{-6}, using Implicit Euler

6.3.2. Linear Solver Analysis

We present the linear solvers results only for the case of GMRES. We take the converged solution from the monolithic stationary solver and perform only one time step to calculate the average number of linear iterations to gain 6 digits. The results at $Re = 100$ with the Implicit Euler scheme are given in Table 6.14 As expected, the results are highly dependent on c but at the same time, linear convergence rates can always be improved by reducing Δt. Comparatively better results are obtained with the second order Crank-Nicolson approach, see table 6.15.

		GMRES					
		$c = 1$		$c = 10$		$c = 100$	
Δt	Grid points	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
0.1	289	22	18	165	88	348	159
	1089	28	24	263	162	819	409
	4225	31	29	411	297	1000	1000
	16641	32	31	656	524	1000	1000
	66049	37	34	776	726	1000	1000
0.01	289	7	6	48	27	250	99
	1089	8	7	63	40	438	179
	4225	9	8	87	63	904	437
	16641	9	9	115	94	1000	1000
	66049	10	10	127	124	1000	1000
0.001	289	3	3	14	8	84	29
	1089	4	3	15	10	117	43
	4225	4	4	17	13	212	91
	16641	4	4	20	16	357	174
	66049	5	4	21	19	510	304

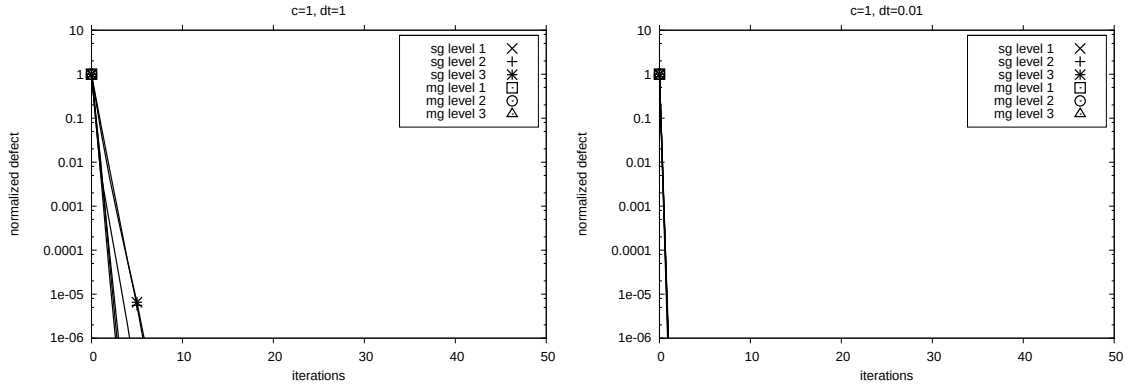
Table 6.14: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Implicit Euler, $Re = 100$

		GMRES					
		$c = 1$		$c = 10$		$c = 100$	
Δt	Grid points	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
0.1	289	15	14	119	61	334	147
	1089	18	17	178	112	736	330
	4225	21	20	272	193	1000	928
	16641	22	21	395	320	1000	1000
	66049	25	25	492	467	1000	1000
0.01	289	5	5	32	18	193	67
	1089	6	6	39	25	321	121
	4225	7	7	50	36	653	292
	16641	7	7	61	51	1000	630
	66049	8	8	72	69	1000	1000
0.001	289	3	3	11	8	59	19
	1089	4	4	11	9	72	26
	4225	4	4	12	10	123	51
	16641	4	4	13	11	187	91
	66049	4	4	14	13	256	158

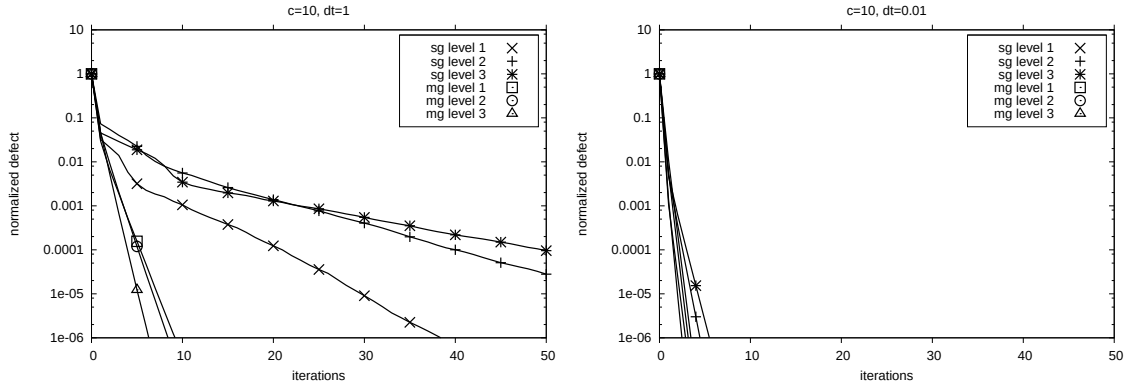
Table 6.15: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Crank Nicolson, $Re = 100$

6.3.3. Multigrid preconditioned GMRES

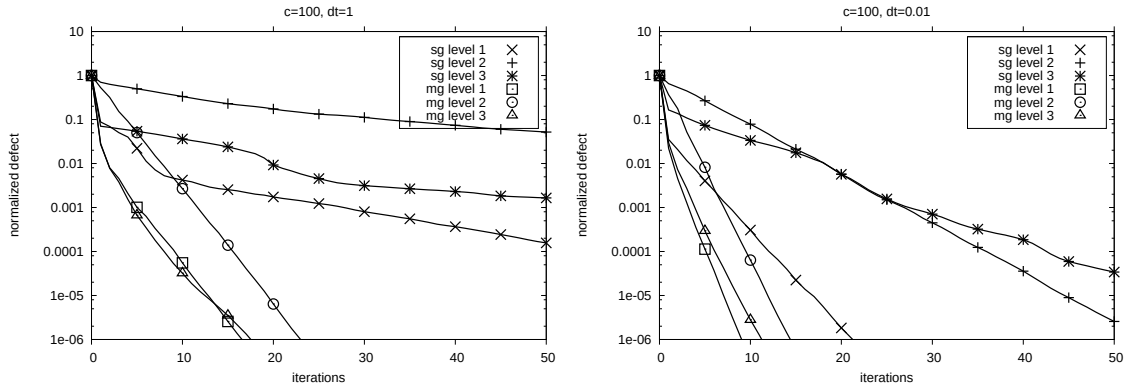
We also compare the results of plain GMRES with multigrid preconditioned GMRES by giving single grid and two-grid convergence in figure 6.1. We show the plots for the three highest levels and GEF only. For the case $\Delta t = 0.01$ on the right, the two variants both converge with few iterations, except for the case c=100, where we obtain good results with coarse grid correction, but we observe strong level-dependence of the single grid curves. For a large time step of $\Delta t = 1$ on the other hand, multigrid always shows significant improvements, with the exception of the case $c = 1$, which is by construction well suited for the GEF approach.



(a) Convergence for $c = 1$



(b) Convergence for $c = 10$



(c) Convergence for $c = 100$

Figure 6.1: Convergence of (normalized) linear defect with GMRES (single-grid and MG preconditioned), 4225 to 66049 grid points, $\Delta t = 1$ vs. $\Delta t = 0.01$

6.4. Numerical results for Driven Cavity at Re=5000

In this section we extend our solver tests to the case $Re = 5000$. From table 6.16 it could be seen that a large c is better for the non-linear rates and Newton is almost independent of the number of grid points while Fixed Point needs more iterations with growing number of degrees of freedom. The choice of bigger c positively influences the non-linear solvers, especially for the fixed point method. However, by looking at the results of linear solvers given in table 6.17 and 6.18, an optimal parameter range is known to be around $c = 10$ which is also confirmed by the theoretical and analytical results given in [35].

		$c = 10$		$c = 100$	
Δt	Grid points	Newton	Fixed Point	Newton	Fixed Point
0.1	289	4	12	3	6
	1089	4	12	3	7
	4225	4	13	3	8
	16641	5	18	4	8
	66049	8	34	6	26
0.01	289	2	10	2	5
	1089	3	11	2	5
	4225	4	12	2	5
	16641	4	12	3	6
	66049	6	13	3	10
0.001	289	2	7	2	5
	1089	2	7	2	5
	4225	3	9	2	5
	16641	3	10	3	6
	66049	5	11	3	8

Table 6.16: Driven Cavity: upw2, Result of non-linear solvers to gain 6 digits using Implicit Euler, $Re = 5000$, zero starting solution

		GMRES			
		$c = 10$		$c = 100$	
Δt	Grid Points	GEF	GEF(\)	GEF	GEF(\)
0.1	289	277	99	391	158
	1089	576	224	1000	467
	4225	1000	569	1000	1000
	16641	1000	1000	1000	1000
	66049	1000	1000	1000	1000
0.01	289	98	30	283	98
	1089	169	53	597	210
	4225	302	99	1000	473
	16641	553	179	1000	1000
	66049	1000	456	1000	1000
0.001	289	36	9	99	29
	1089	43	13	174	50
	4225	58	18	321	94
	16641	70	26	690	188
	66049	130	53	1000	321

Table 6.17: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Implicit Euler, $Re = 5000$

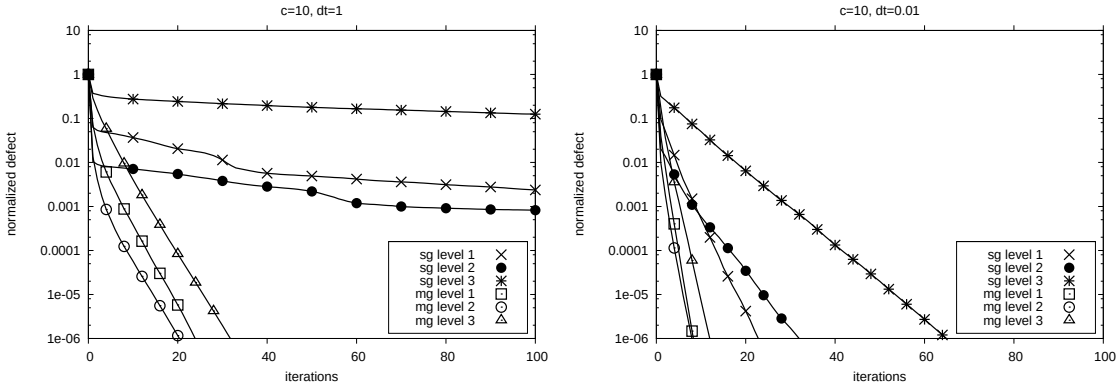
		GMRES			
		$c = 10$		$c = 100$	
Δt	Grid points	GEF	GEF(\)	GEF	GEF(\)
0.1	289	218	72	363	149
	1089	431	150	902	394
	4225	881	338	1000	978
	16641	1000	732	1000	1000
	66049	1000	1000	1000	1000
0.01	289	68	19	228	70
	1089	106	32	439	143
	4225	176	56	875	308
	16641	286	93	1000	763
	66049	585	236	1000	1000
0.001	289	28	10	71	19
	1089	32	11	111	30
	4225	40	13	189	53
	16641	45	16	378	101
	66049	71	30	642	235

Table 6.18: Driven Cavity:upw2, No. of averaged linear iterations to gain 6 digits, using Crank Nicolson, $Re = 5000$

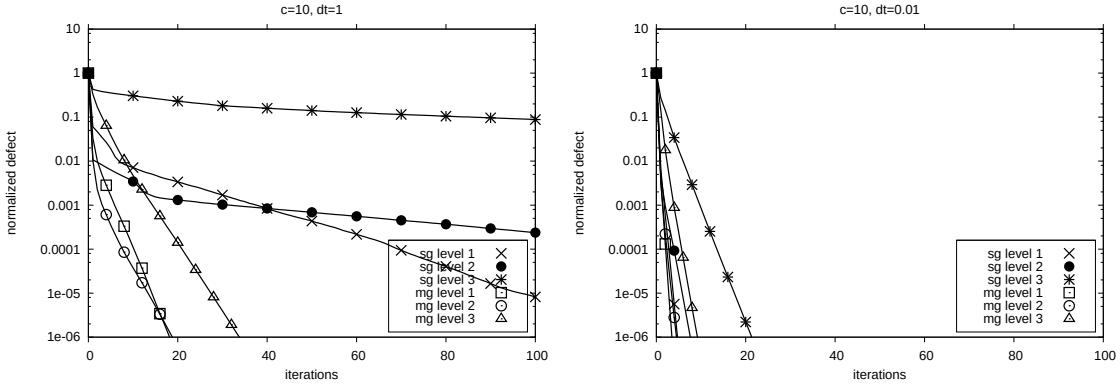
In Table 6.19 we compare the Newton and Fixed point schemes at $Re = 5000$ taking the $Re = 100$ solution as starting guess. It is observed that for large Δt and large number of unknowns, the non-linear convergence can fail and the FP method requires explicit damping (here $\sigma = 0.75$), while the Newton method performs remarkably well. For the Newton method results are almost grid independent especially for smaller values of time step size. In the same table, we compare the results for linear convergence rates for single grid and two grid application. We also show that the linear solver can deal with the high Reynolds number by use of multigrid and that the GEF(\) has a strong impact on the convergence rates. However, on the highest refinement level. using multigrid with the plain GEF smoothing steps is quite similar to smoothing with GEF(\), which is also visible in the linear convergence plots in figure 6.2. Again, we observe that the collision preconditioning takes stronger effect on the small levels and otherwise may be omitted to optimize the runtime.

		nonlinear		linear single grid		linear two-grid	
Δt	Grid points	Newton	Fixed Point	GEF	GEF(\)	GEF	GEF(\)
1	289	3	10	8.27E-1	3.56E-1	5.37E-1	2.33E-1
	1089	3	15	9.51E-1	7.24E-1	5.67E-1	4.27E-1
	4225	4	21	9.84E-1	9.19E-1	6.13E-1	5.45E-1
	16641	5	25	9.96E-1	9.76E-1	6.14E-1	5.95E-1
	66049	5	32	9.97E-1	9.91E-1	6.64E-1	7.55E-1
0.01	289	3	7	2.23E-1	1.29E-4	1.48E-1	1.15E-4
	1089	3	7	2.23E-1	1.29E-4	1.48E-1	1.15E-4
	4225	3	8	5.91E-1	9.48E-2	2.35E-1	3.76E-2
	16641	4	9	7.26E-1	2.82E-1	2.67E-1	1.05E-1
	66049	4	10	8.20E-1	5.41E-1	3.32E-1	2.38E-1
0.001	289	2	5	9.80E-7	4.93E-7	1.67E-7	5.18E-7
	1089	3	5	5.99E-7	1.42E-7	5.35E-7	1.59E-7
	4225	3	6	1.70E-7	1.51E-7	9.98E-7	1.73E-7
	16641	3	7	6.02E-7	5.80E-7	1.36E-7	7.74E-8
	66049	3	7	6.60E-5	2.50E-7	1.41E-5	1.90E-7

Table 6.19: Driven Cavity $Re = 5000$: Results of nonlinear/linear solvers to gain 6 digits, only case c=10, nonlinear results with $Re = 100$ as starting guess



(a) Convergence for $c = 10$, GEF(plain)



(b) Convergence for $c = 10$, GEF(\)

Figure 6.2: Driven Cavity $Re = 5000$: Convergence of linear defect with GMRES (single-grid and MG preconditioned), 4225 up to 66049 grid points, $\Delta t = 1$ vs. $\Delta t = 0.01$

6.5. Flow around cylinder: bench3

After having shown the solver's efficiency with the driven cavity problem, we now switch to check the accuracy of the applied time discretization schemes and the space discretization on highly adapted grids. For this purpose, the well known *flow around cylinder* [58] benchmark is considered. The geometry of this benchmark is indicated in figure 6.3. Here, the domain Ω consists of a channel of height $H = 0.41$ and length $L = 2.2$ having a circular cylinder located at $(0.2, 0.2)$ with diameter $D = 0.1$. The value of the kinematic viscosity is set to $\nu = 10^{-3}$. The Reynolds number Re determining the flow is defined as $Re = \frac{U_{mean}D}{\nu}$ in which $U_{mean} = \frac{2}{3}U_{max}$.

The drag and lift values are

$$F_D = \int_S (\rho\nu \frac{\partial u_t}{\partial n} n_y - p n_x) dS, \qquad F_L = -\int_S (\rho\nu \frac{\partial u_t}{\partial n} n_x + p n_y) dS$$

representing the total forces in the horizontal and vertical directions, respectively. Furthermore, the pressure drop between two points on the cylinder which is defined as

$$\Delta p = p_A - p_B,$$

where $A(0.15, 0.2)$ and $B(0.25, 0.2)$ are points on the boundary of the cylinder. We modify our configuration used in [33], using a time dependent inflow profile given by

$$U(0, y, t) = 4U_{max} y(H - y)(sin(t.\pi/8))/H^2, \qquad V(0, y, t) = 0.$$

with $U_{max} = 1.5$ and the time interval is $0 \leq t \leq 8s$. This gives a time varying Reynolds number between $0 \leq Re(t) \leq 100$ and we plot the forces acting on the cylinder for the whole 8 seconds of simulation time. The accuracy of the problem depends on the following benchmark quantities.

- the drag coefficient $C_D = \frac{2F_D}{\rho U_{mean}^2 D}$
- the lift coefficient $C_L = \frac{2F_L}{\rho U_{mean}^2 D}$
- pressure difference Δp between front and back of the cylinder

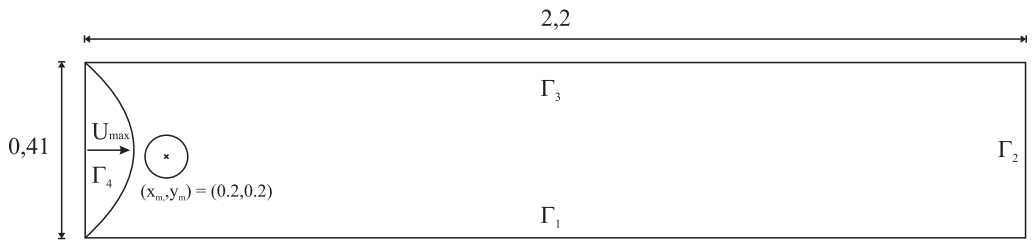


Figure 6.3: Geometry for the *flow around cylinder* configuration in 2D.

In Figure 6.5 we obtain excellent results for the pressure difference Δp between front and back of the cylinder, when we use extrapolation of the pressure on the boundary. However, as described in [33] this has no influence on the drag and lift when using the force

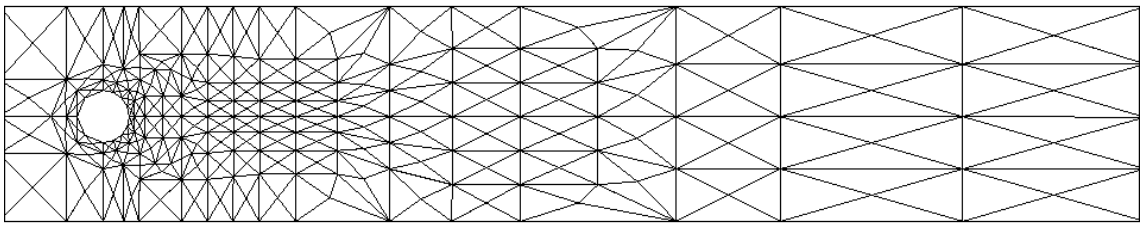


Figure 6.4: *Triangular grid for flow around cylinder*

evaluation method proposed in [43]. With grid refinement, we are successively increasing c and obtain a good approximation of the reference drag (from Featflow). The lift shows strong oscillations and is more difficult to reproduce, as shown in Figure 6.6. However, we get the right period on the highest refinement level and a good approximation of the amplitude when the Mach-number is sufficiently reduced. The accuracy using a large time step of $\Delta t = 1/100$ is compared to a semi-implicit simulation with $\Delta t = 1/4000$. The advanced performance in time becomes abundantly clear from the contiguous graphs over the whole simulation time. The results of drag and lift for two fixed levels (16k and 66k) but with increasing the value of c successively are given in figures 6.7–6.10 along with their zoomed versions. The results for drag and lift are strongly dependent on the Mach number.

Nevertheless, the results are not perfect and in a next step we will try to increase the order of the spatial discretization. The Discontinuous Galerkin method is a first extension to our method which allows to preserve the developed upwinding techniques, while basis functions of arbitrary order can be used, especially in crucial parts of the geometry. For further details and some initial results, see Appendix C.

	Mesh information			upw1	upw2
Level	Elements	Vertices	Midpoints	Total unknowns	Total unknowns
0	520	286	806	286	1 092
1	2 080	1 092	3 172	1 092	4 264
2	8 320	4 264	12 584	4 264	16 848
3	33 280	16 848	50 128	16 848	66 976
4	133 120	66 976	200 096	66 976	267 072

Table 6.20: *flow around cylinder* mesh information

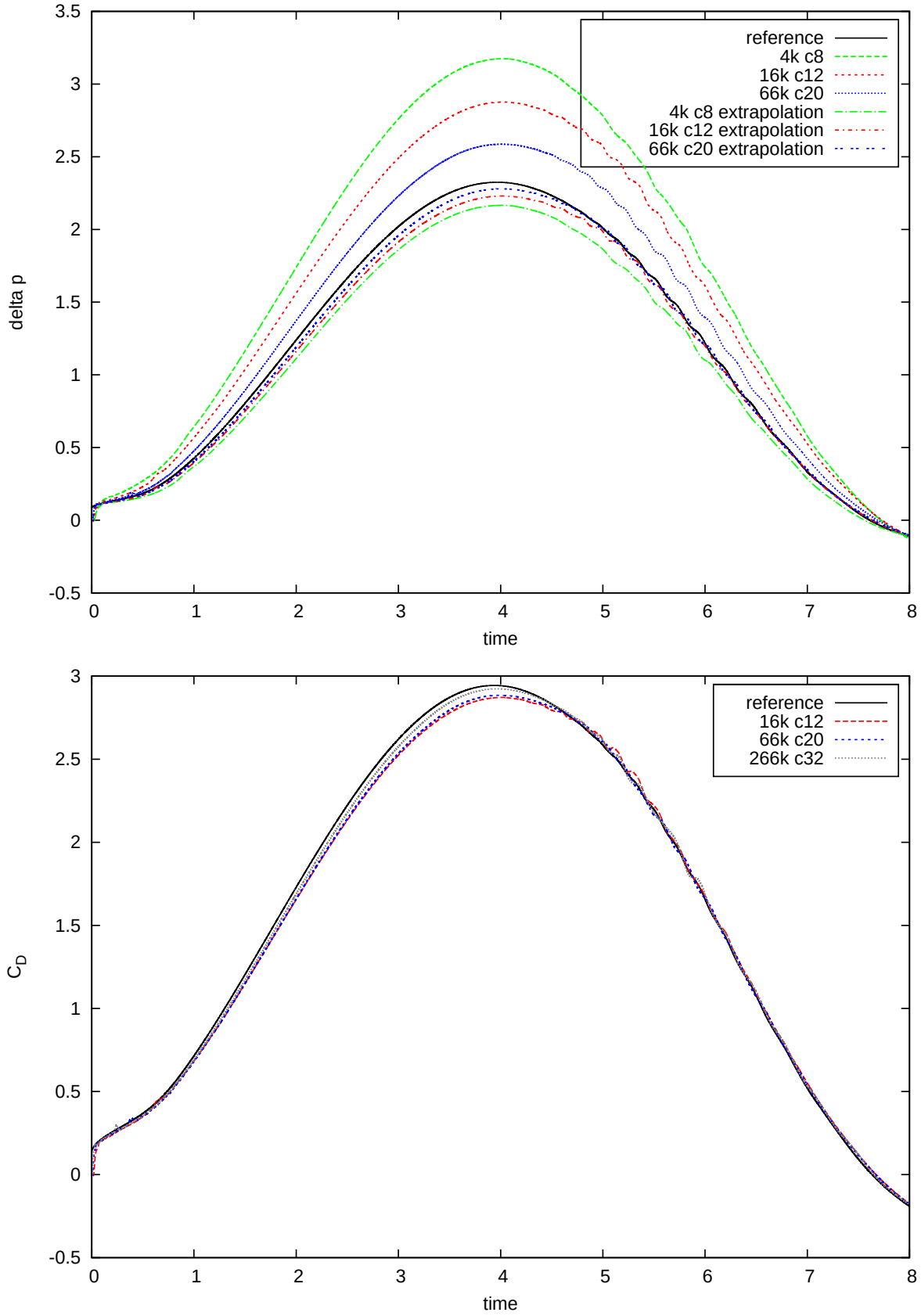


Figure 6.5: *Flow around cylinder* with 8 sec. inflow pulse: Δp and drag

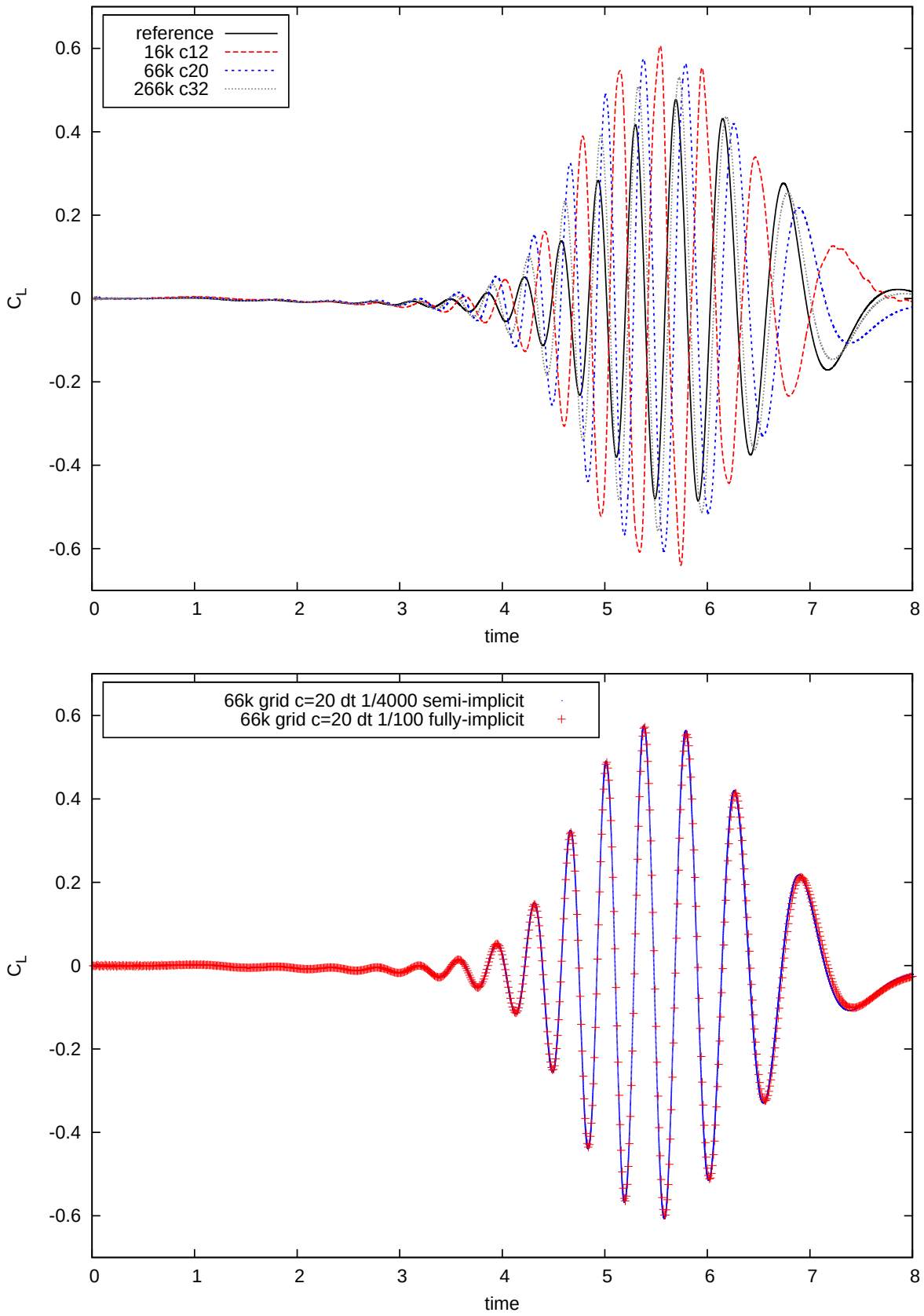


Figure 6.6: *Flow around cylinder* with 8 sec. inflow pulse: lift

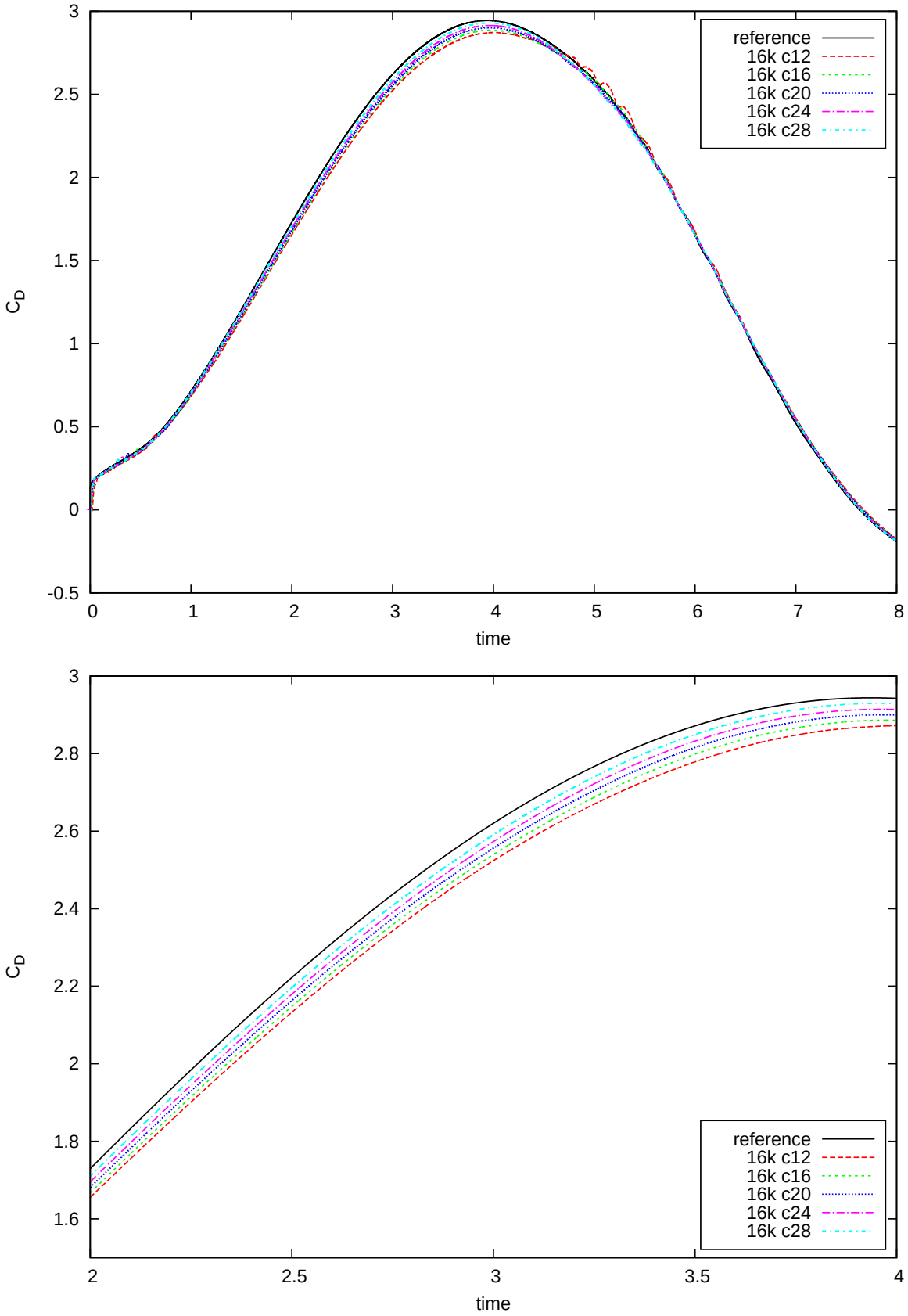


Figure 6.7: *bench3*: 16 848 grid points, drag

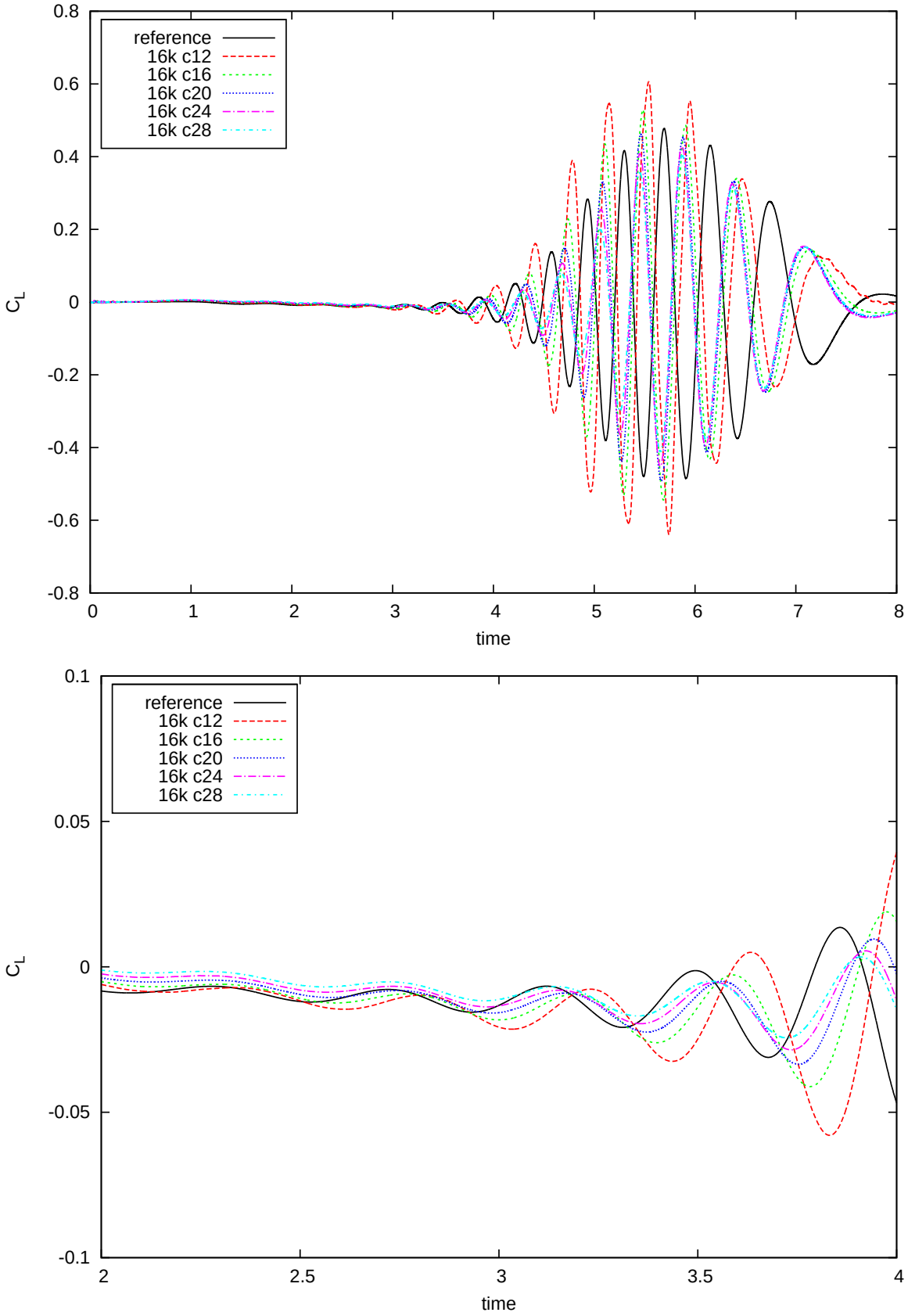


Figure 6.8: *bench3*: 16 848 grid points, lift

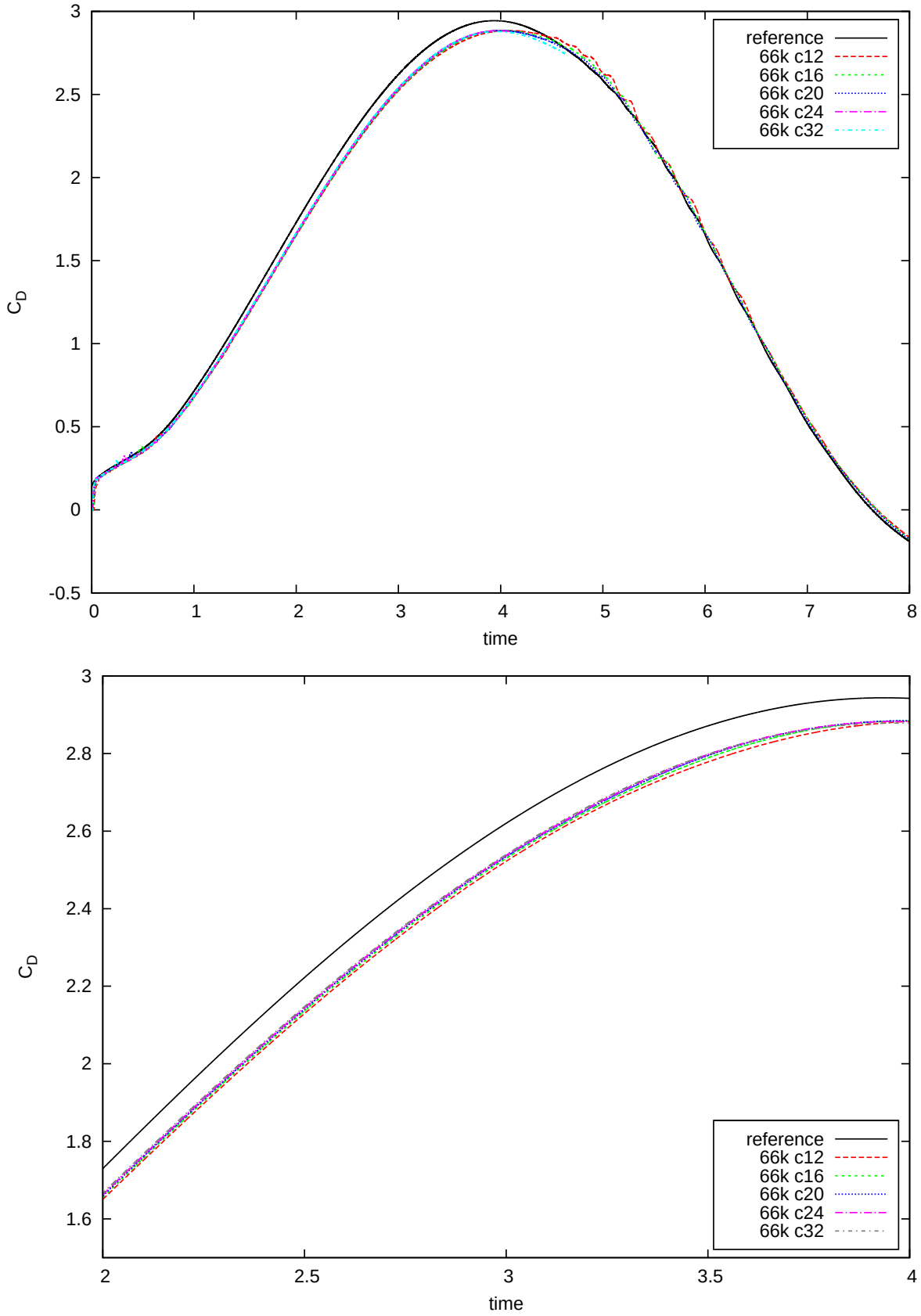


Figure 6.9: *bench3*: 66 976 grid points, drag

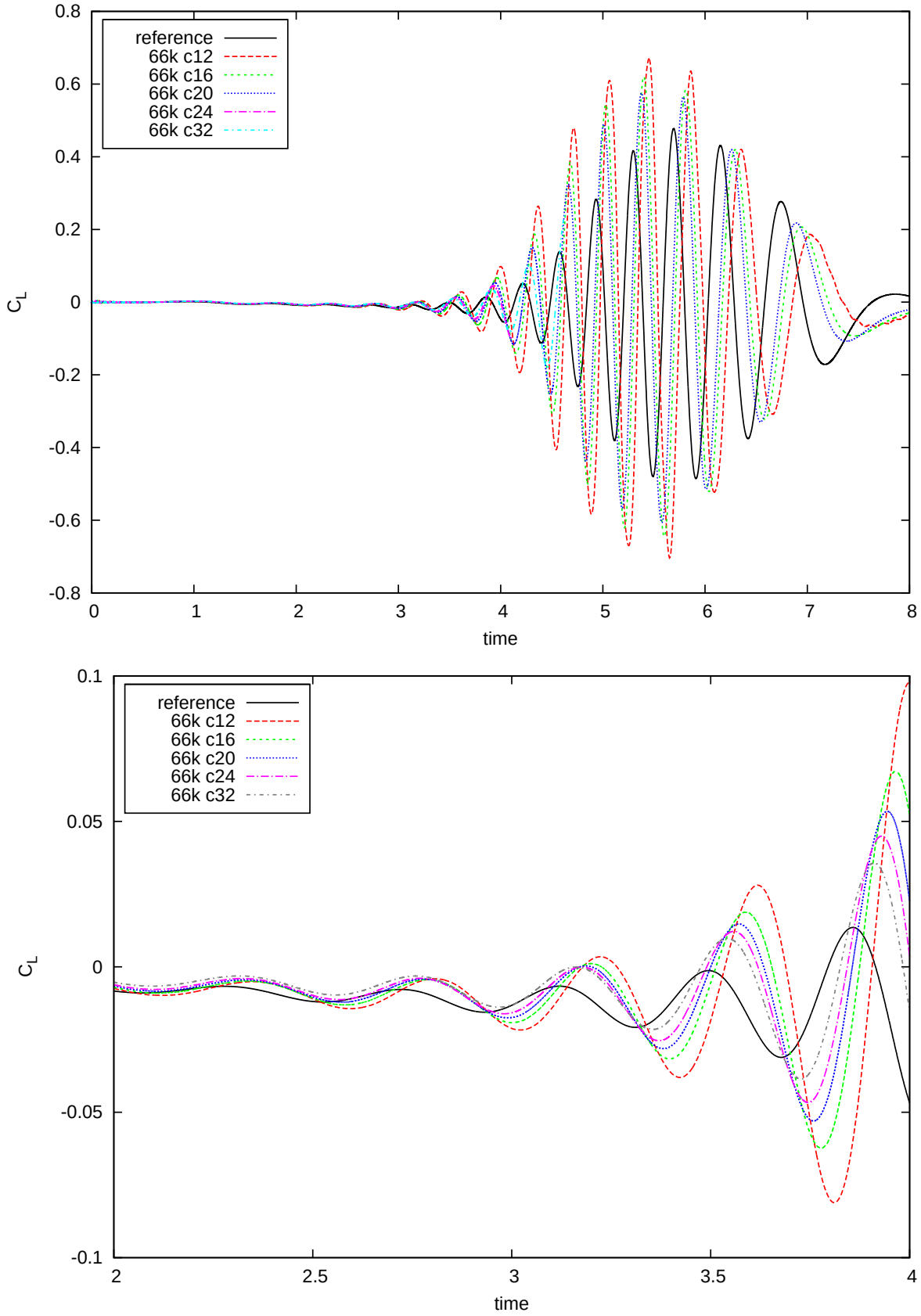


Figure 6.10: *bench3*: 66 976 grid points, lift

7

Non Newtonian Fluids

This chapter is devoted to the extension of our monolithic stationary approach for Newtonian fluids given in Chapter 3 towards the non-constant viscosity case. In general, the viscosity may depend on pressure, temperature and shear-rate. However, in this chapter we deal with the case of shear-dependent viscosity and we modify our non-linear solvers to tackle with the increased non-linearity. First, we describe some models in which the viscosity is a function of shear-rate and then we present the results based on these models for selected configurations.

7.1. Generalised Newtonian Fluids

Fluids that are not adequately described by a linear constitutive relation are usually referred to as 'non-Newtonian fluids'. Several constitutive relations have been proposed to describe these fluids. An important subclass of non-Newtonian fluids are 'generalized Newtonian fluids'. A significant progress is made in developing the mathematical theory of generalized Newtonian fluids during the last two decades. Examples of these models include the Power Law, Carreau-Yasuda, Eyring and Cross models.

7.1.1. Power Law Model

For the power law model, the viscosity is a function of shear rate and it is given by the following relation

$$\mu = \mu_0(\varepsilon + \dot{\gamma}^2)^{n/2-1}$$

where $\dot{\gamma}$ is the shear rate which is related to the second invariant of the strain rate tensor D (i.e. $\dot{\gamma} = \sqrt{D_{ij}D_{ij}}$), n is the power law exponent and ε is a regularisation parameter. If $n/2 < 1$, the model describes shear thinning behaviour while for the cases $n/2 > 1$, it describes shear-thickening behaviour.
Although the power-law model offers the simplest representation of shear thinning and shear thickening behaviour, it does have a number of shortcomings. Generally, it applies over only a limited range of shear rates and this model does not predict the zero and infinite shear viscosities.

7.1.2. Carreau-Yasuda Model

The Carreau-Yasuda model [5] is used to describe the shear thinning behaviour of many fluids and it incorporates two limiting viscosities μ_0 and μ_∞. In this model the viscosity μ is given by

$$\frac{\mu(\dot{\gamma}) - \mu_\infty}{\mu_0 - \mu_\infty} = (1 + (\lambda\dot{\gamma})^a)^{\frac{n-1}{a}}$$

where μ_0 is the zero-shear-rate viscosity ($\dot{\gamma} \to 0$), μ_∞ is the infinity-shear-rate viscosity ($\dot{\gamma} \to \infty$), λ is the time constant and $n < 1$ is the power-law exponent and a is a model constant. This five parameter$(\mu_0, \mu_\infty, \lambda, a, n)$ model is able to describe shear thinning behaviour over wide ranges of shear rates but only at the expense of the added complexity of some extra parameters. This model predicts Newtonian fluid behaviour when either $n = 1$ or $\lambda = 0$ or both.

7.1.3. Carreau Model

If we choose $a = 2$ in the Carreau-Yasuda model, then we obtain the Carreau model as a special case given by

$$\frac{\mu(\dot{\gamma}) - \mu_\infty}{\mu_0 - \mu_\infty} = (1 + (\lambda\dot{\gamma})^2)^{\frac{n-1}{2}}$$

where symbols have the same meaning as described above.

7.2. Derivation for the expression of norm D

First of all we derive an expression for the norm of D for the two dimensional case.

$$V = \langle u, v, 0 \rangle$$

$$\nabla V = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}$$

$$(\nabla V)^T = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{bmatrix}$$

$$\begin{aligned} \nabla V + (\nabla V)^T &= \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} + \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{bmatrix} \\ &= \begin{bmatrix} 2\frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & 2\frac{\partial v}{\partial y} \end{bmatrix} \end{aligned}$$

There are two formulations for D:

1. Gradient formulation
   For the Gradient formulation we take $D = \nabla V$ and in this case the norm D is given by

$$\|D(u)\|^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$$

2. Tensor formulation
   For the tensor formulation we take $\boldsymbol{D} = (\nabla V + (\nabla V)^T)/2$ and in this case the norm D is given by
$$\|D(u)\|^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2$$

For the Newtonian case both the gradient and tensor formulations are equivalent in case when Dirichlet data is imposed on boundaries but for non-Newtonian case, the two formulations are not equivalent. We will consider the tensor formulation in our tests.

7.3. Computation of Stress Tensor via LBM

The computation of the stress tensor for an incompressible fluid with pressure p is given by
$$\sigma_{\alpha\beta} = -p\delta_{\alpha\beta} + 2S_{\alpha\beta}.$$
As the standard way to evaluate the term $S_{\alpha\beta}$ is
$$S_{\alpha\beta} = \nu(\partial_\beta u_\alpha + \partial_\alpha u_\beta)$$
which involves the derivatives of the corresponding velocity field. This is usually done as a post processing step after having obtained the velocity field. This involves additional cost in conventional CFD methods, however in LB method we can directly calculate $S_{\alpha\beta}$ via the summation

$$-\sum_i \xi_{i,\alpha}\xi_{i,\beta}(f_i - f_i^{eq})$$

We get the summation-coeffcients by evaluating the product of the corresponding entries in $\boldsymbol{\xi}_i = c \cdot \mathbf{e}_i$ with
$$\mathbf{e}_i \in \left\{ \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}-1\\-1\end{pmatrix}, \begin{pmatrix}0\\-1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix} \right\}$$
which means that we get the three terms

$$S_{11} = -c^2(f_E^{neq} + f_{NE}^{neq} + f_{NW}^{neq} + f_W^{neq} + f_{SW}^{neq} + f_{SE}^{neq})$$

$$S_{22} = -c^2(f_{NE}^{neq} + f_N^{neq} + f_{NW}^{neq} + f_{SW}^{neq} + f_S^{neq} + f_{SE}^{neq})$$

$$S_{12} = -c^2(f_{NE}^{neq} - f_{NW}^{neq} + f_{SW}^{neq} - f_{SE}^{neq})$$

So the stress tensor components can be obtained without any additional computational cost. Also, computing the shear in this manner is efficient since it removes the need to calculate derivatives of the velocity.

7.4. The Modified Newton Method

In this section, we present the modified Newton scheme for the case of non-linear viscosity. As described in Chapter 1, for the shear dependent viscosity, the advection is treated fully implicitly while the viscosity is treated in a semi-implicit way. We modify the Algorithm 5.1 for the monolithic stationary approach and for non-linear viscosity and the new algorithm for this Modified Newton's method is given in Algorithm 7.1.

Algorithm 7.1 The Modified Newton Method for non-constant viscosity

1) The nonlinear system for the stationary monolithic approach for f_i^{n+1} is given by

$$\mathbf{T}_i f_i^{n+1} - \frac{1}{\tau}\sum_k \omega_{ik} f_k^{n+1} = \mathbf{0} \quad i = 0, ..., 8.$$

2) In each nonlinear iteration, perform a Newton iteration for $n = 1, 2, \ldots$ with the resulting linear block-system

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \left[\frac{\partial \mathcal{R}(\mathbf{x}^n)}{\partial \mathbf{x}}\right]^{-1} \mathcal{R}(\mathbf{x}^n)$$

Derivation of Jacobian (analytically)

$$df_i^{eq} = W_i \left[\rho + \rho_0 \left(\frac{3}{c^2}(\boldsymbol{\xi}_i \cdot \mathbf{u}) + \frac{9}{c^4}(\boldsymbol{\xi}_i \cdot \mathbf{u})(\boldsymbol{\xi}_i \cdot \tilde{\mathbf{u}}) - \frac{3}{c^2}\mathbf{u} \cdot \tilde{\mathbf{u}}\right)\right] =: \sum_k \bar{\omega}_{ik} f_k$$

Coefficients $\bar{\omega}_{ik}$ and ω_{ik} are only different by a factor of 2 in front of the quadratic terms.

3) Solve the linear system

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^n)}{\partial \mathbf{x}}\right] \Delta \mathbf{x} = -\mathcal{R}(\mathbf{x}^n)$$

with

$$\left[\frac{\partial \mathcal{R}(\mathbf{x}^n)}{\partial \mathbf{x}}\right] = \left(\begin{bmatrix} \mathbf{T}_0 & & & \\ & \mathbf{T}_1 & & \\ & & \ddots & \\ & & & \mathbf{T}_8 \end{bmatrix} - \frac{1}{\tau(\dot{\gamma})^{(n-1)}} \begin{bmatrix} \bar{\omega}_{00} & \bar{\omega}_{01} & \ldots & \bar{\omega}_{08} \\ \bar{\omega}_{10} & \bar{\omega}_{11} & & \vdots \\ \vdots & & \ddots & \vdots \\ \bar{\omega}_{80} & \ldots & \ldots & \bar{\omega}_{88} \end{bmatrix} \right).$$

4) Update the solution

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \sigma \Delta \mathbf{x}^{(n)}$$

where $\sigma > 0$ is the damping parameter.

7.5. Channel flow with Power law and Carreau models

We consider 2D stationary flow in a straight channel. This benchmark is the well known Poiseuille flow. The flow in the channel is either driven by a constant pressure gradient or by enforcing the velocity profile from the analytical solution of the flow at the inlet. We consider a bounded domain $\Omega \subset R^2$ with $\Omega = [0,1] \times [0,1]$ and we prescribe an inflow parabolic profile as

$$u = 4y(1-y), \quad v = 0 \tag{7.1}$$

We performed our simulations based on the Power law and Carreau models with different settings of respective parameters. For the case $n = 2$, the Power law model corresponds to the Newtonian case, for which we have analytical expressions for velocity and stress components. Therefore, we compute the L_2 error between the LB solution and exact solution (see Figures 7.1–7.2) using both the Ladd and Zou-He boundary schemes. As expected, the results of L_2 error are better with Zou-He method and also the range of optimal c is bigger as compared with Ladd scheme.

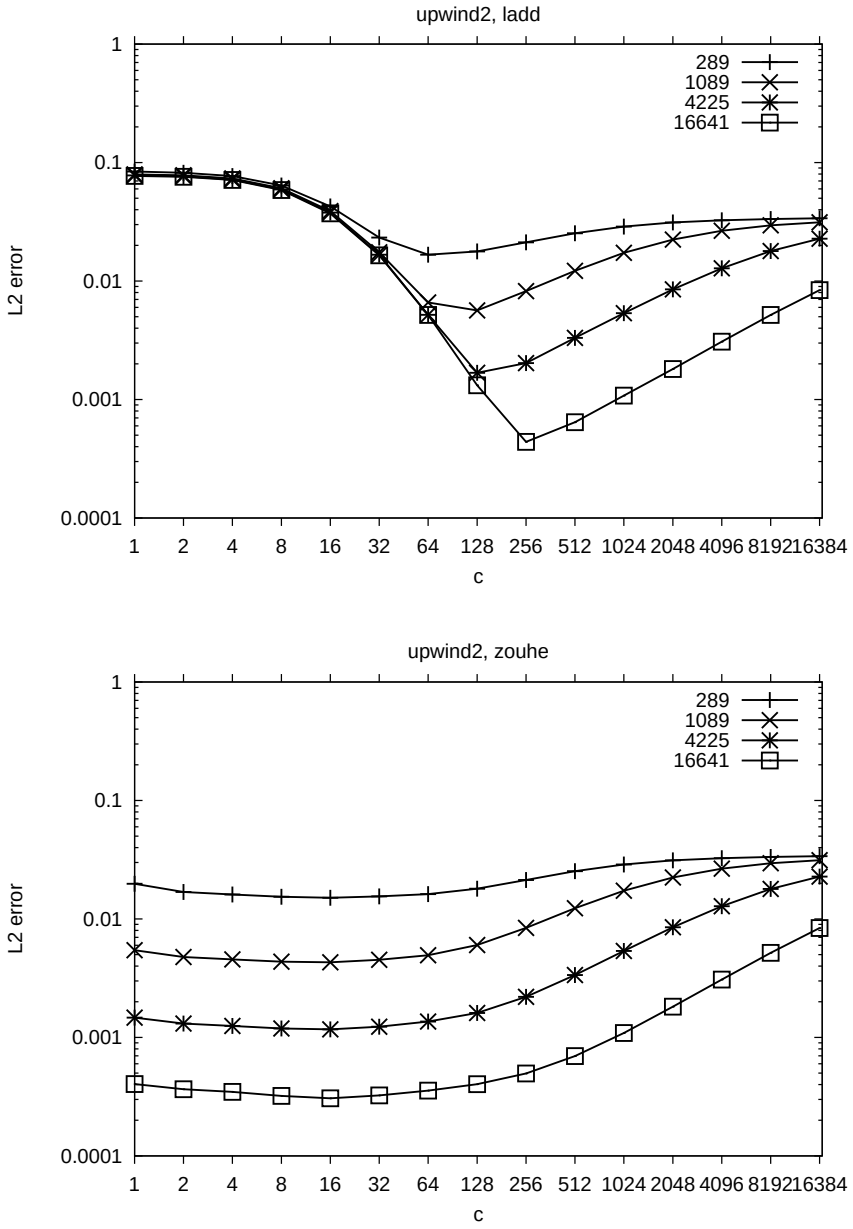


Figure 7.1: Poiseuille Flow:Comparison of L_2 error for boundary schemes, $\mu_0 = 1$

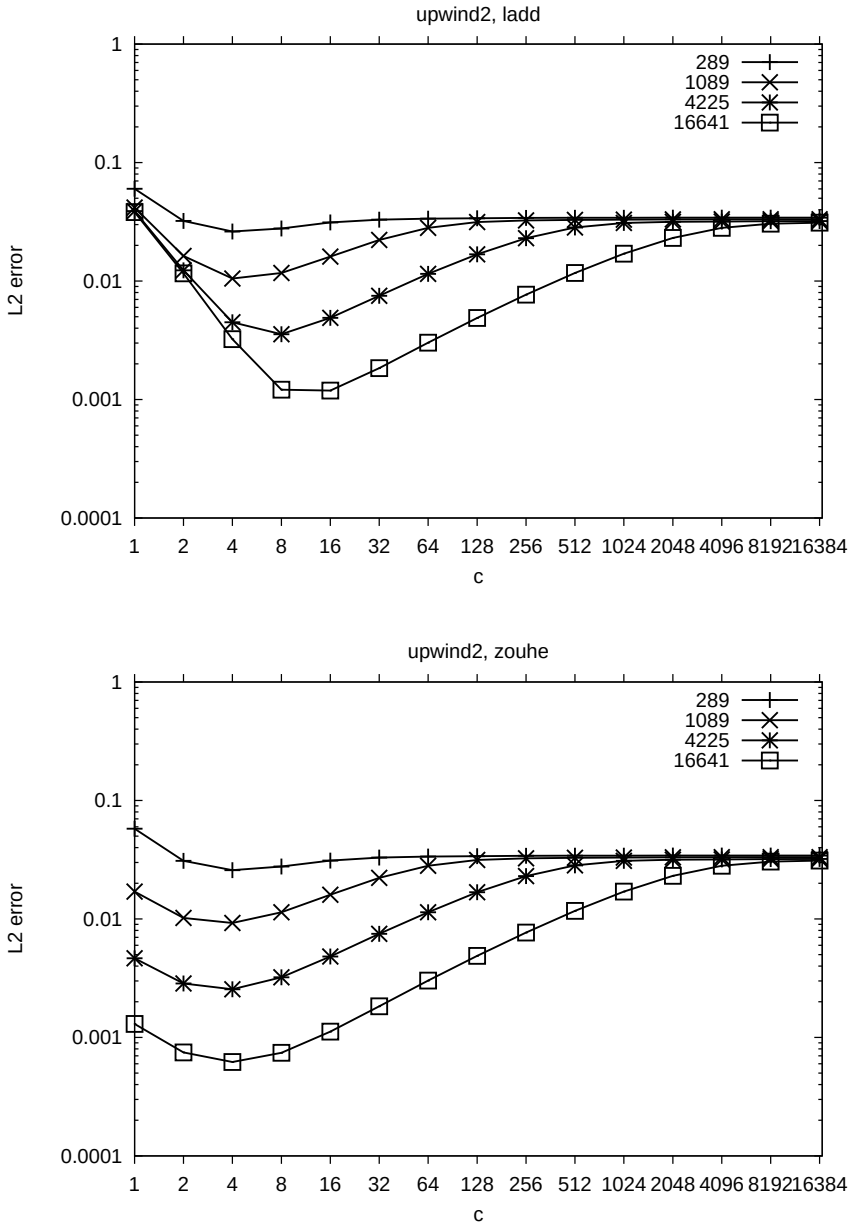


Figure 7.2: Poiseuille Flow: Comparison of L_2 error for boundary schemes, $\mu_0 = 0.01$

From figures 7.3–7.6, it is evident that the computation of stress components using LB approach is better than using the derivative based (DB) approach, just in the optimal area of c, and also on finer grids. The reason is that in stress computation via derivative based approach, one might calculate the derivatives of the obtained velocity field which leads to loss of accuracy while in LB approach we use non-equilibrium distributions to calculate the stress tensor (see section 7.3). Therefore, the stress tensor components can be obtained without almost any additional cost. We also see that the L_2 error starts to grow up for larger values of c due to Mach dependent space discretization error.

We also present results for the obtained velocity and viscosity profiles for the Power law and Carreau model cases. The value of sound parameter is set to $c = 10$ in these tests. Typical behaviour of viscosity for shear thinning and shear thickening cases for different values of the power law exponent n could be seen from the figures 7.7 and 7.8 for the cases $\mu_0 = 1$ and $\mu_0 = 0.1$ respectively. Also as the Carreau model generally used only for shear-thinning cases, so we show the solution only for the shear-thinning cases, see figures 7.9–7.12 for different settings of the parameters appearing in the Carreau model.

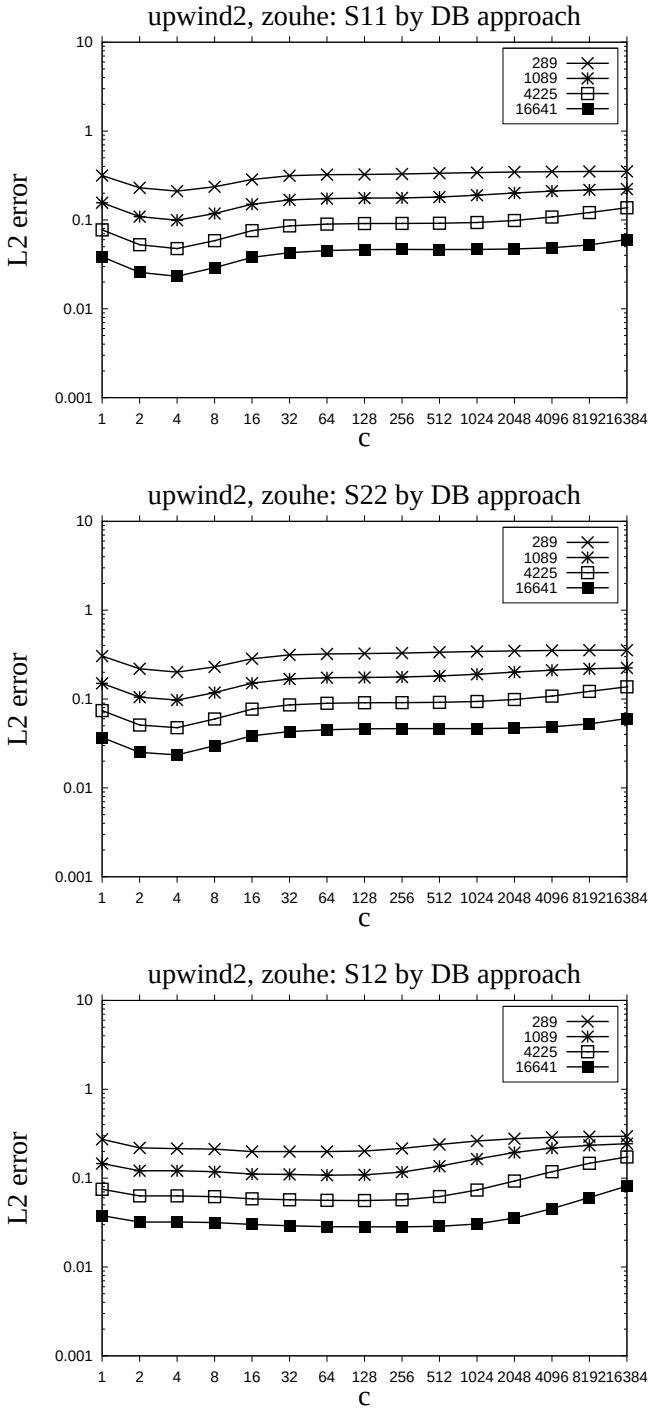


Figure 7.3: Poiseuille Flow: Stress components by FEM method, $UPW2$, $\mu_0 = 1$

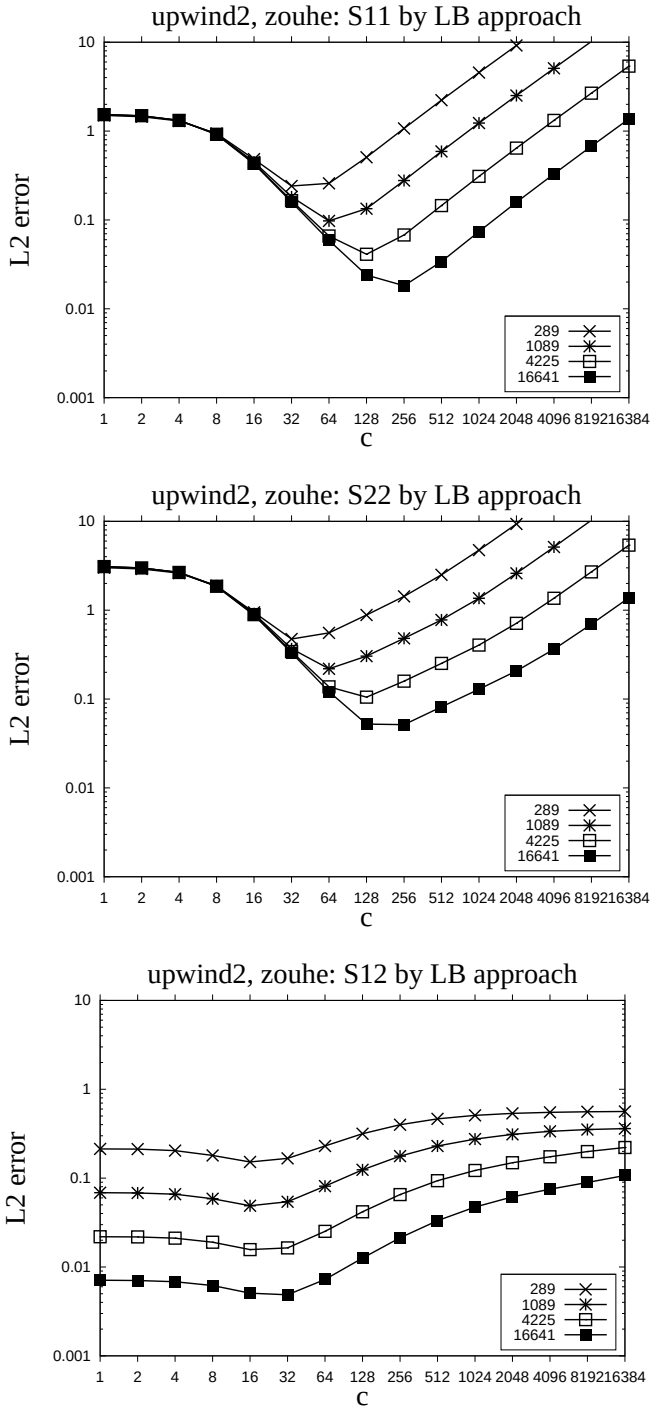


Figure 7.4: Poiseuille Flow: Stress components by LBM method, $UPW2$, $\mu_0 = 1$

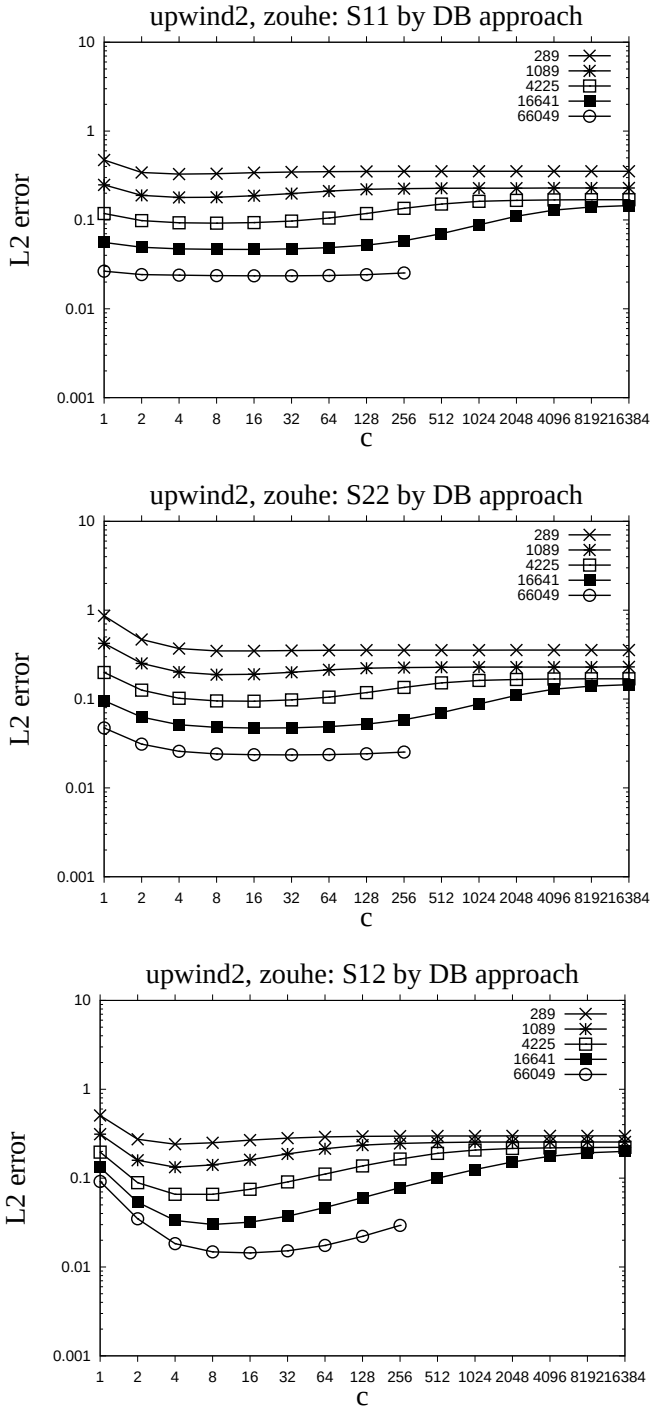


Figure 7.5: Poiseuille Flow: Stress components by FEM method, $UPW2$, $\mu_0 = 0.01$

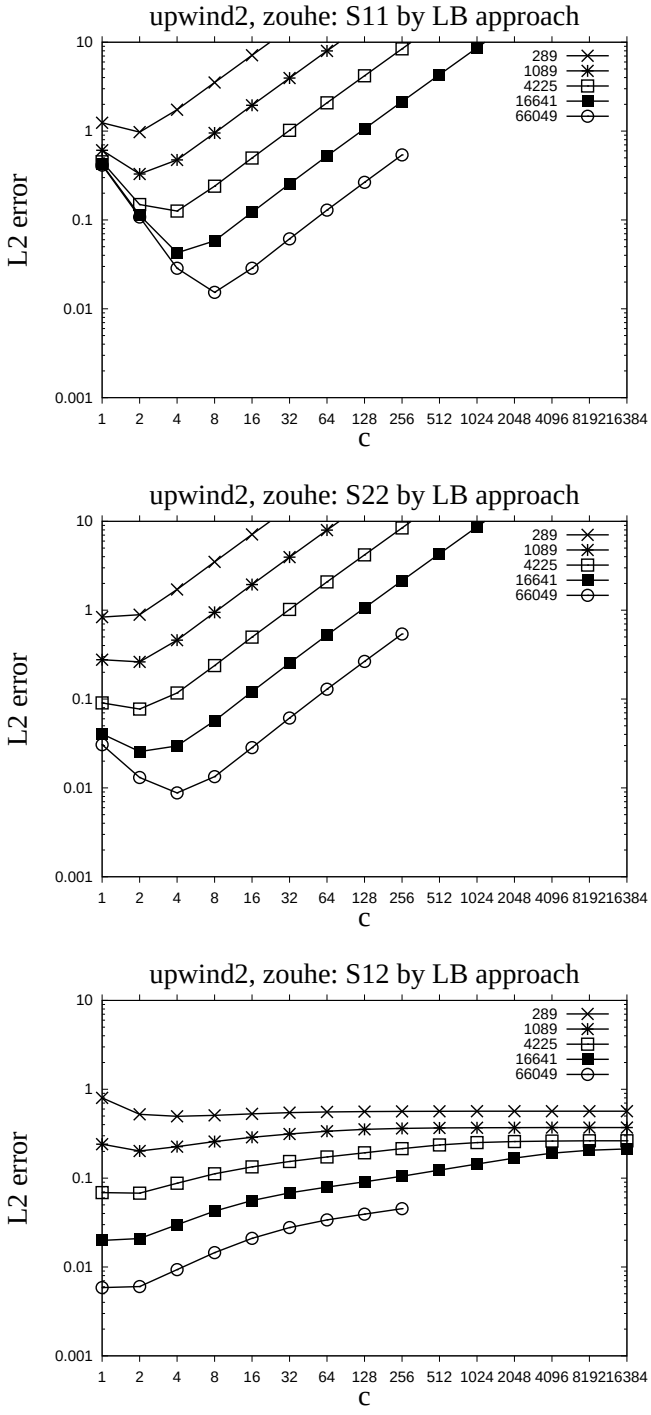


Figure 7.6: Poiseuille Flow: Stress components by LBM method, $UPW2$, $\mu_0 = 0.01$

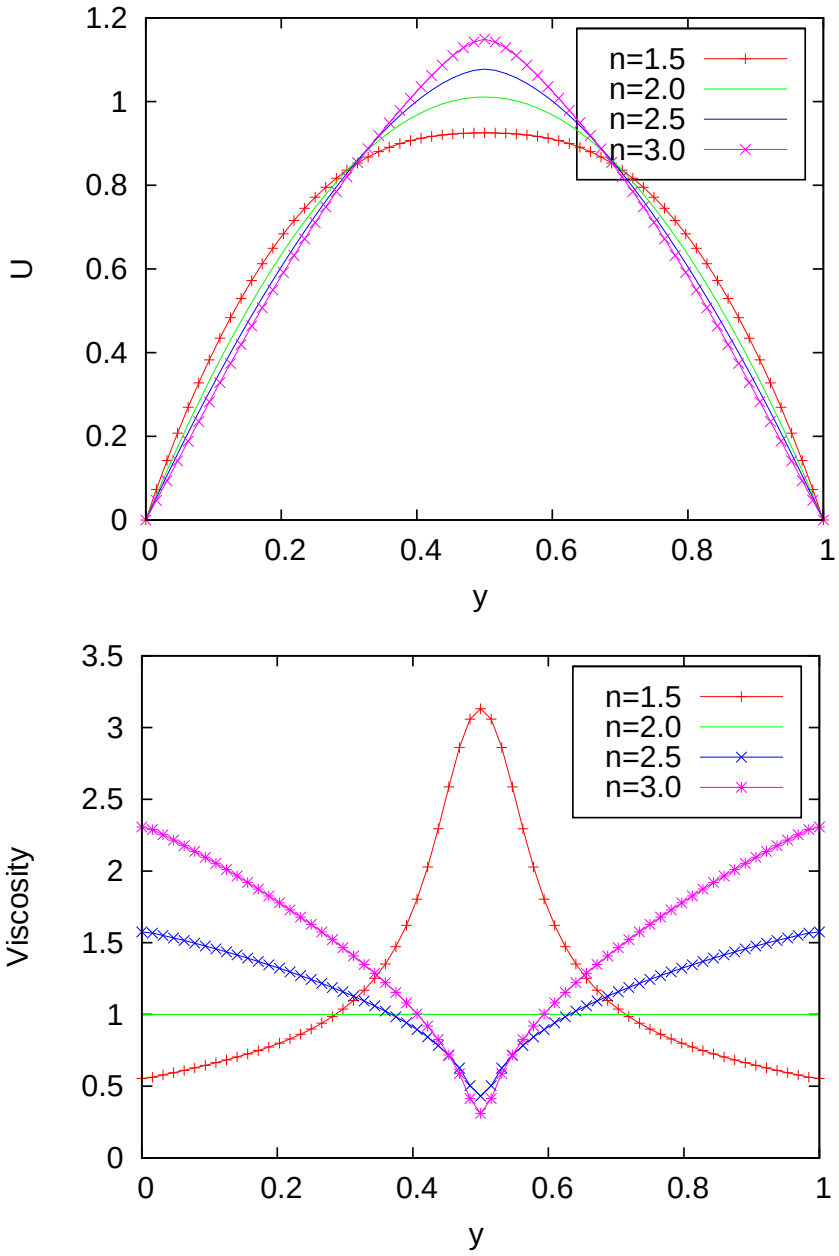


Figure 7.7: Power law: Comparison of U and viscosity, $\varepsilon = 0.01$ $\mu_0 = 1$

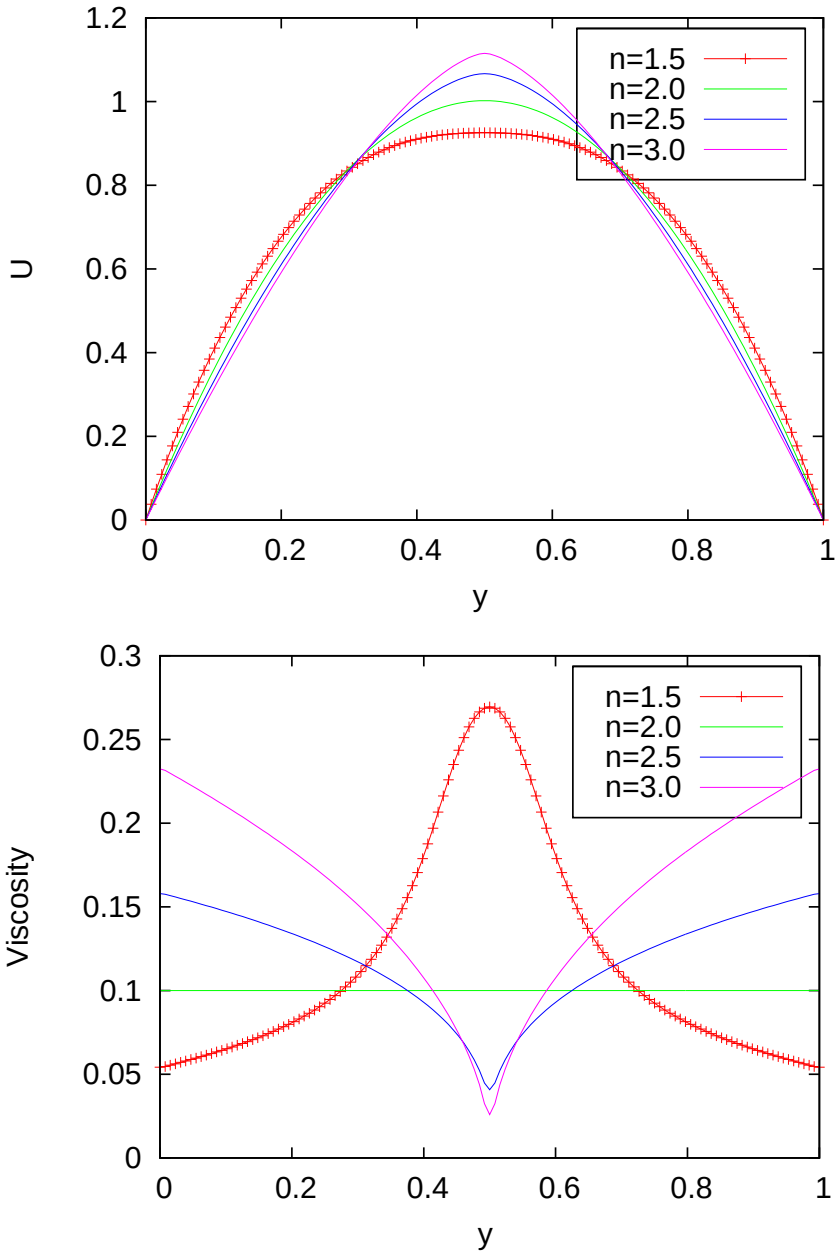


Figure 7.8: Power law: Comparison of U and viscosity, $\varepsilon = 0.01$ $\mu_0 = 0.1$

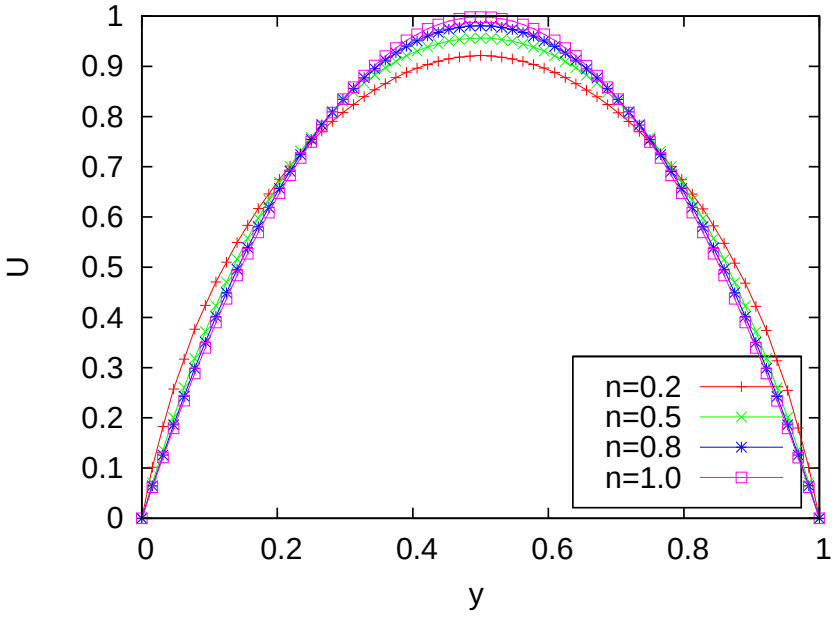


Figure 7.9: Carreau model: Comparison of U , $\lambda = 1$, $\mu_\infty = 0$, $\mu_0 = 1$, $c = 10$

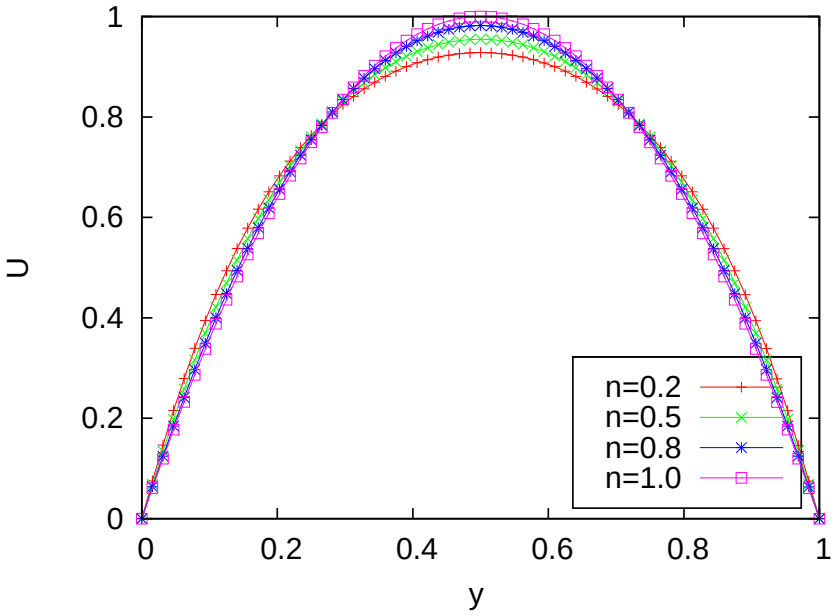


Figure 7.10: Carreau model: Comparison of U , $\lambda = 1$, $\mu_\infty = 0$, $\mu_0 = 1$, $c = 100$

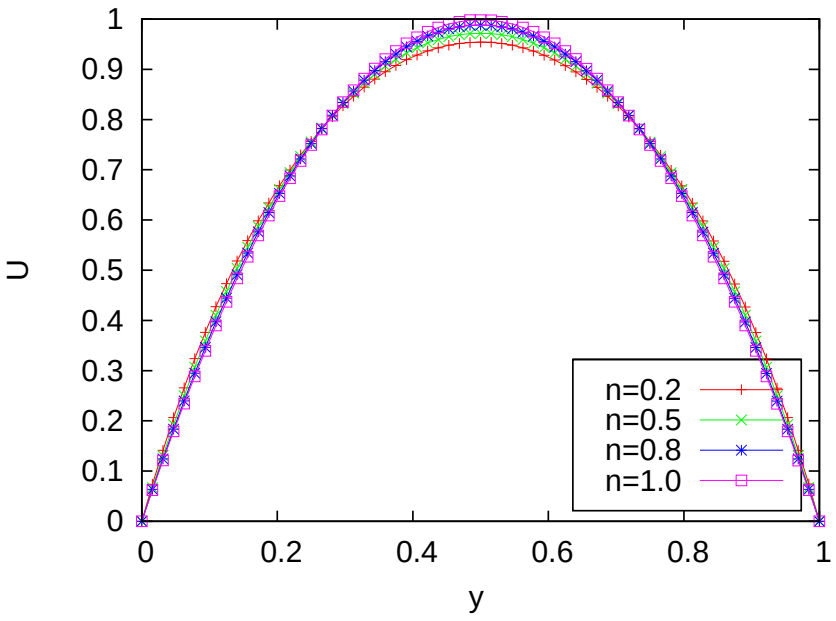


Figure 7.11: Carreau model: Comparison of U , $\lambda = 0.5$, $\mu_\infty = 0$, $\mu_0 = 1$, $c = 10$

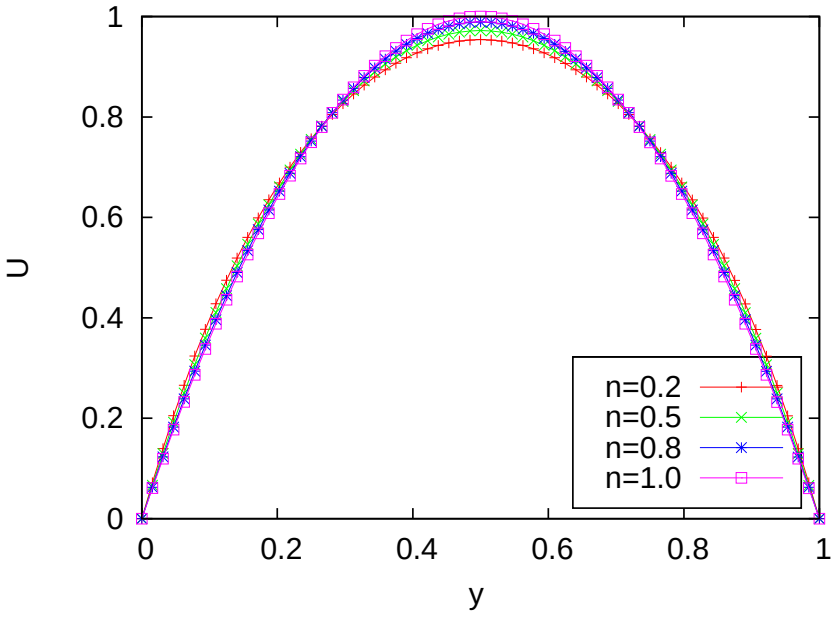


Figure 7.12: Carreau model: Comparison of U , $\lambda = 0.5$, $\mu_\infty = 0$, $\mu_0 = 1$, $c = 100$

7.6. Driven Cavity

As a second test case for non-Newtonian flows, we consider the driven cavity problem. The computational domain is a unit square. The left, right and bottom walls have no slip conditions and the flow is driven by a uniform translation of the upper lid. We use the Ladd boundary scheme to implement the macroscopic boundary conditions. Simulations are conducted using the Power Law model at $\mu_0 = 0.1$ and $\mu_0 = 0.01$ and for different values of the power law exponent n. Predicted horizontal and vertical velocities along the horizontal and vertical wall bisectors are shown in Figures 7.13–7.16. Since we do not have a closed form solution for the driven cavity problem, therefore, the simulated results are compared with the finite element reference solutions obtained by Featflow [67] and the agreements are quite satisfactory. The effect of the power law exponent n on horizontal and vertical components of the velocity is also given in figure 7.17.

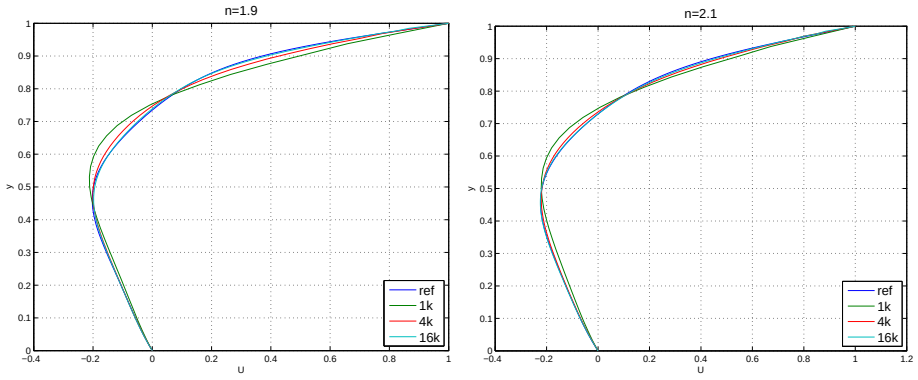


Figure 7.13: Predicted horizontal velocities of driven cavity flows, $\mu_0 = 0.01$, $c = 10$

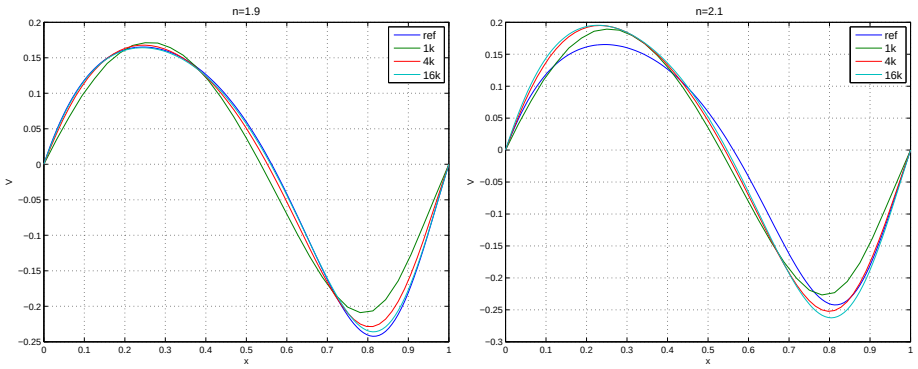


Figure 7.14: Predicted vertical velocities of driven cavity flows, $\mu_0 = 0.01$, $c = 10$

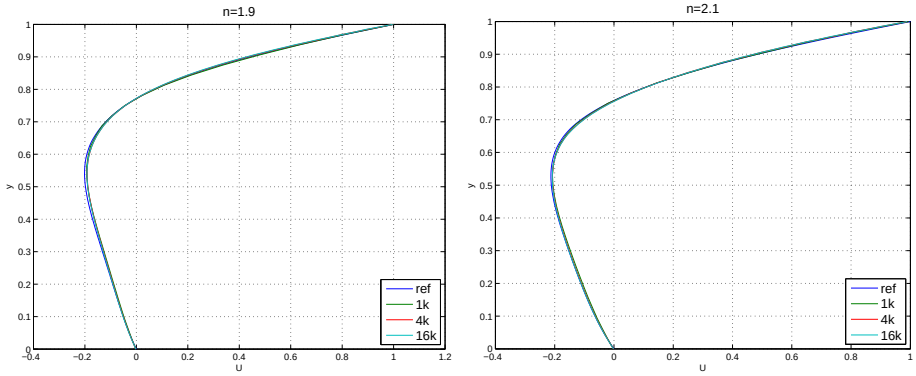


Figure 7.15: Predicted horizontal velocities of driven cavity flows, $\mu_0 = 0.1$, $c = 10$

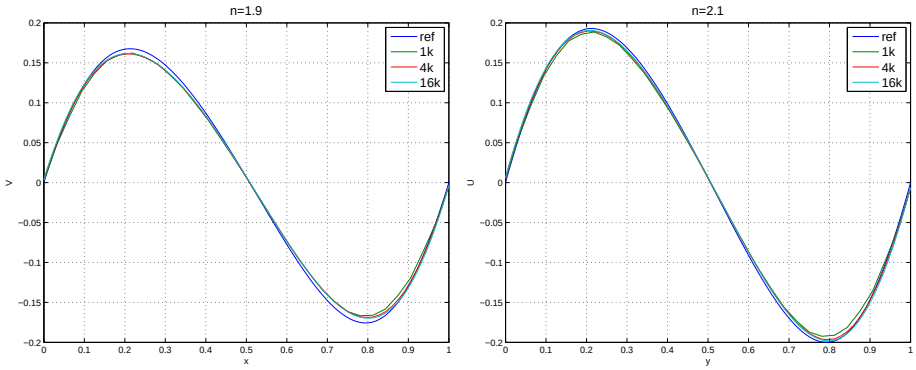


Figure 7.16: Predicted vertical velocities of driven cavity flows, $\mu_0 = 0.1$, $c = 10$

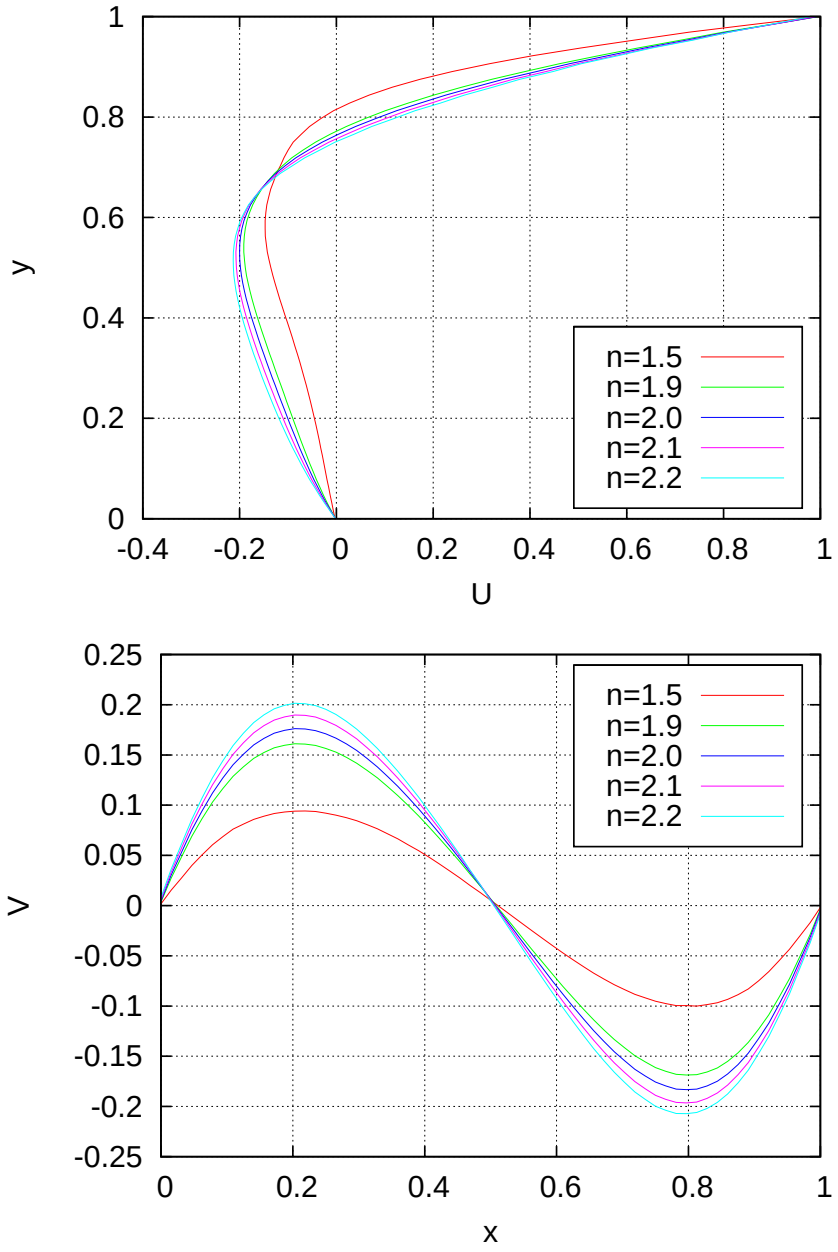


Figure 7.17: Dimensionless velocity profiles along vertical and horizontal bisectors showing the effect of the power law exponent 'n' at $\mu_0 = 0.1$, $c = 10$

7.7. Non-Linear Solver analysis

In this section, we compare the non-linear solvers modified for the non-constant viscosity case. Here, we have additional non-linearity due to different non-linear viscosity functions described in section 7.1. As described in Chapter 1, for the non-linear viscosity, we can not treat the viscosity in a fully implicit way, therefore we do not use a full Newton method for this case, instead we use a modified Newton approach, see Algorithm 7.1, in which we treat the advection fully implicitly, however, we use the viscosity in an explicit way by taking its value from the previous non-linear iteration. As it could be seen from the tables 7.1–7.4 that even this modified Newton method still gave far better convergence rates as compared with the fixed point iteration. Also, it is clear that larger c is better for the non-linear solvers but increasing the number of grid points increases Fixed Point iterations while our (semi) Newton scheme meets the criteria in a few iterations. The deviation from $n = 2$ means departure from the Newtonian case. In case of shear-thickening $n = 2.5$ we need more iterations as compared with shear-thinning case $n = 1.5$ means shear-thickening is more hard to tackle.

	shear thinning $n = 1.5$		Newtonian $n = 2.0$		shear thickening $n = 2.5$	
Grid points	Newton	Fixed Point	Newton	Fixed Point	Newton	Fixed Point
81	4	4	3	5	5	6
289	5	5	2	5	5	7
1089	6	7	2	5	6	9
4225	8	9	2	4	9	13
16641	9	10	2	4	12	20

Table 7.1: Driven Cavity:Power Law, No. of non-linear iterations to reduce the non-linear defect by 10^{-6}, UPW=1, $\varepsilon = 0.01$, $\mu_0 = 0.01$, $c = 10$

	shear thinning $n = 1.5$		Newtonian $n = 2.0$		shear thickening $n = 2.5$	
Grid points	Newton	Fixed Point	Newton	Fixed Point	Newton	Fixed Point
81	3	4	2	3	4	5
289	4	4	2	3	4	5
1089	4	4	2	3	5	6
4225	5	5	2	3	5	7
16641	6	6	2	3	6	8

Table 7.2: Driven Cavity:Power Law, No. of non-linear iterations to reduce the non-linear defect by 10^{-6}, UPW=1, $\varepsilon = 0.01$, $\mu_0 = 0.01$, $c = 100$

	shear thinning $n = 1.5$		Newtonian $n = 2.0$		shear thickening $n = 2.5$	
Grid points	Newton	Fixed Point	Newton	Fixed Point	Newton	Fixed Point
81	5	6	4	7	6	6
289	6	8	4	8	7	11
1089	9	13	5	12	11	28
4225	12	18	5	17	20	85
16641	14	24	5	20	47	131

Table 7.3: Driven Cavity: Power Law, No. of non-linear iterations to reduce the non-linear defect by 10^{-6}, UPW=2, $\varepsilon = 0.01$, $\mu_0 = 0.01$, $c = 10$

	shear thinning $n = 1.5$		Newtonian $n = 2.0$		shear thickening $n = 2.5$	
Grid points	Newton	Fixed Point	Newton	Fixed Point	Newton	Fixed Point
81	4	4	2	4	6	6
289	4	5	3	5	6	6
1089	6	7	3	7	7	10
4225	8	13	3	11	10	25
16641	11	20	3	18	24	104

Table 7.4: Driven Cavity: Power Law, No. of non-linear iterations to reduce the non-linear defect by 10^{-6}, UPW=2,$\varepsilon = 0.01$, $\mu_0 = 0.01$, $c = 100$

7.8. Linear Solvers with Modified Preconditioners

We also modified our preconditioners in order to be able to use iterative solvers in an efficient manner for the case of a non-linear viscosity function. In Tables 7.5–7.7, we give the results using the transport preconditioner and block-Jacobi preconditioner in the GMRES solver. It can be seen that the transport preconditioner is suitable for transport dominated cases i.e. for smaller values of c. However, it is not suitable for large values of c; especially the deviations from the Newtonian case are very large for higher values of c. On the other hand the block-Jacobi performs well for collision dominated cases i.e. for larger values of c, at least for small number of unknowns.

Although the Power-Law model is the simplest representation of shear thinning and shear thickening behaviour, it has certain shortcomings. Generally, it applies over only a limited range of shear rates and it does not predict the zero and infinite shear rate viscosities, therefore, in the Power-Law model, the viscosity might change dramatically. So in order to have a better understanding of our preconditioners, we also performed simulations using Carreau model. The corresponding results are given in tables 7.11–7.13. Here the case $n = 1$ corresponds to the Newtonian case. Since the Carreau model is generally used to describe the shear-thinning behaviour, so we chose the values of n describing the shear-thinning effect only. In this model when we deviate from $n = 1$, the average number of linear iterations does not change too much in contrast to the behaviour shown by the Power Law model.

	tr-pre			bl-jac		
Grid points	thinning n=1.5	Newtonian n=2.0	thickening n=2.5	thinning n=1.5	Newtonian n=2.0	thickening n=2.5
81	52	44	59	189	153	197
289	66	56	71	453	352	519
1089	90	71	99	1000	778	1000
4225	148	92	161	1000	1000	1000

Table 7.5: Driven Cavity: Power Law, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.1$, $c = 1$

	tr-pre			bl-jac		
Grid points	thinning n=1.5	Newtonian n=2.0	thickening n=2.5	thinning n=1.5	Newtonian n=2.0	thickening n=2.5
81	149	118	158	88	83	102
289	271	155	362	198	192	203
1089	502	184	564	456	448	479
4225	715	221	834	983	951	1000

Table 7.6: Driven Cavity: Power Law, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.1$, $c = 10$

	tr-pre			bl-jac		
Grid points	thinning n=1.5	Newtonian n=2.0	thickening n=2.5	thinning n=1.5	Newtonian n=2.0	thickening n=2.5
81	287	238	311	82	87	108
289	546	431	685	268	208	287
1089	843	647	1000	552	525	594
4225	1000	875	1000	1000	1000	1000

Table 7.7: Driven Cavity: Power Law, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.1$, $c = 100$

	tr-pre			bl-jac		
Grid points	thining n=1.5	Newtonian n=2.0	thickening n=2.5	thining n=1.5	Newtonian n=2.0	thickening n=2.5
81	114	102	126	89	84	92
289	182	133	192	205	187	221
1089	283	158	298	464	439	475
4225	415	179	443	1000	1000	1000

Table 7.8: Driven Cavity: Power Law, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.01$, $c = 1$

	tr-pre			bl-jac		
Grid points	thining n=1.5	Newtonian n=2.0	thickening n=2.5	thining n=1.5	Newtonian n=2.0	thickening n=2.5
81	253	226	264	74	66	79
289	573	401	589	196	172	202
1089	1000	617	1000	569	521	585
4225	1000	858	1000	1000	1000	1000

Table 7.9: Driven Cavity: Power Law, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.01$, $c = 10$

	tr-pre			bl-jac		
Grid points	thining n=1.5	Newtonian n=2.0	thickening n=2.5	thining n=1.5	Newtonian n=2.0	thickening n=2.5
81	319	305	342	70	64	73
289	777	731	796	174	153	181
1089	1000	1000	1000	519	492	576
4225	1000	1000	1000	1000	1000	1000

Table 7.10: Driven Cavity: Power Law, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.01$, $c = 100$

	tr-pre			bl-jac		
Grid points	n=0.5	n=0.8	n=1.0	n=0.5	n=0.8	n=1.0
81	47	44	42	177	162	153
289	60	54	51	394	373	352
1089	82	75	71	788	758	727
4225	113	99	92	1000	1000	1000

Table 7.11: Driven Cavity: Carreau Model, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.1$, $\mu_\infty = 0$, $\lambda = 1$, $c = 1$

	tr-pre			bl-jac		
Grid points	n=0.5	n=0.8	n=1.0	n=0.5	n=0.8	n=1.0
81	154	132	114	96	91	87
289	256	221	148	287	275	319
1089	409	301	182	682	664	651
4225	576	367	224	1000	1000	1000

Table 7.12: Driven Cavity: Carreau Model, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.1$, $\mu_\infty = 0$, $\lambda = 1$, $c = 10$

	tr-pre			bl-jac		
Grid points	n=0.5	n=0.8	n=1.0	n=0.5	n=0.8	n=1.0
81	253	241	232	89	87	83
289	556	532	421	233	221	215
1089	1000	1000	645	622	599	530
4225	1000	1000	864	1000	1000	1000

Table 7.13: Driven Cavity: Carreau Model, No. of averaged linear iterations to gain 6 digits, using GMRES, UPW=2, $\mu_0 = 0.1$, $\mu_\infty = 0$, $\lambda = 1$, $c = 100$

7.9. Remarks

In this Chapter, we have presented the results for the nonlinear shear-dependent viscosity using a monolithic stationary framework. As a final remark we expect to get better convergence results for non-linear and linear solvers in a non-stationary framework which has been very successful as shown by the results given in Chapter 6 for the case of constant viscosity case. The time stepping schemes would damp the non-linearities for smaller time step sizes giving improved convergence rates for Newton and Fixed Point solvers. At the same time, linear convergence rates in iterative schemes can always be improved by reducing Δt.

8

Conclusion and Outlook

In this thesis we have used new algorithms for the Lattice Boltzmann equation based on modern techniques from Numerics for PDEs. These algorithms eliminate the drawbacks faced by standard Lattice Boltzmann method and they work for constant and non-constant viscosity. To give a final summary, we revisit our achievements and set our future plans as follows.

Alternative approach to standard LBM

We have treated the Lattice Boltzmann equation in a more general framework to increase the applicability of the lattice Boltzmann approach for incompressible flow problems. A new collision/advection implicit off-lattice discretization is used as an alternative to the standard Lattice Boltzmann method, first motivated by Hübner in [33]. With this monolithic approach we are able to remove the limitations faced by standard LBM such as the use of structured grids, the restrictions due to the CFL condition and the stability problems.

Implicit time discretization

In this thesis, we have used implicit time stepping schemes for accurate and robust numerical simulations of nonstationary flow problems. The accuracy of these schemes is verified by the flow around cylinder benchmark , with a special focus on stable simulations using large Δt. We have also discussed possible variants of semi-implicit schemes all of which lead to nonsymmetric linear systems. The stability limits of the semi-implicit scheme and a comparison with fully implicit schemes is also given.

Higher order upwind schemes for space discretization

Independently of the time stepping, a spatial discretization based on short characteristic upwinding approach is used for the advection term. This special finite difference discretization led to lower triangular transport matrices after the application of a special sorting algorithm in each direction. Corresponding schemes of first and second order upwinding are successfully implemented.

Sorting Algorithm

Based on graph theory we have used a very useful sorting algorithm. On one hand, we obtained high numerical efficiency due to this sorting technique which yielded lower triangular matrices for the transport step and on the other hand this sorting algorithm proved to be the basis for a powerful preconditioner especially in transport dominated configurations.

Non linear and linear solvers

In our monolithic approach the collision operator has to be treated implicitly, resulting in a nonlinear system. For the common incompressible model it was easy to use a continuous Newton method, obtaining throughout excellent convergence independent of mesh refinement.

Efficient Preconditioners

We have used efficient preconditioners which can deal with transport and collision dominated cases. Our numerical results showed that the proposed preconditioners improve the condition number , depending upon the range of c and h. Convection dominated cases have been solved very efficiently using the lower triangular transport blocks. Also a new GEF reformulation of the LBE is derived which can combine the efficiency of direct transport solvers on one hand with special preconditioning for collision dominated cases on the other hand leading to a very efficient and robust monolithic solver. A sophisticated application of the GEF resulted in a prototypical algorithm where multigrid is used as a preconditioner in Krylov-space methods and achieves very good convergence rates for all configurations.

Treatment of non-constant viscosity

After the numerical tests for Newtonian cases, we also implemented our approach towards non-linear viscosity. We modified our basic non-linear solvers to work for non-linear viscosity case. In this case, the implementation of full Newton method was difficult so instead of using full Newton method, we implement a Quasi Newton method, in which the advection is treated fully implicitly while the viscosity is handled in an explicit fashion. Nevertheless this modified approach for Newton gave improved results as compared with fixed point method. We have also modified the basic preconditioners for the linear solvers, in order to use them for nonlinear viscosity functions.

Coupling of LB with Discontinuous Galerkin Method

Regarding future developments, a promising way to combine the upwinding approach with finite element methods can be realized using upwind Discontinuous Galerkin (DG) schemes. These schemes provide a practical framework for the development of high-order accurate methods using unstructured grids. The good thing is that we can use our sorting technique in the same style but now instead the sorting could be done elementwise. Corresponding research is ongoing and we expect advantages using higher order DG approximations especially compared to the finite difference upwinding which naturally has to use first order differences at the boundary.

Appendix

A

From Lattice Boltzmann to Navier-Stokes Equation

The macroscopic flow behavior governed by the Navier-Stokes equations can be recovered from lattice Boltzmann equation by the application of Chapman-Enskog procedure. We start with the LBGK model with SRT given as

$$f_\alpha(\mathbf{x}+\mathbf{e}_\alpha\delta t, t+\delta t) = f_\alpha(\mathbf{x},t) - \frac{1}{\tau}[f_\alpha(\mathbf{x},t) - f_\alpha^{(eq)}(\mathbf{x},t)] \tag{A.1}$$

Performing a Taylor series expansion in time and space with $\delta x = \delta t$ to the second order, we obtain

$$\delta t\frac{\partial f_\alpha}{\partial t} + \delta t e_{\alpha k}\frac{\partial f_\alpha}{\partial x_k} + \frac{(\delta t)^2}{2}\Big[\frac{\partial f_\alpha}{\partial t} + 2e_{\alpha k}\frac{\partial^2 f_\alpha}{\partial t \partial x_k} + e_{\alpha k}e_{\alpha n}\frac{\partial^2 f_\alpha}{\partial x_k \partial x_n}\Big] + \frac{1}{\tau}(f_\alpha - f_\alpha^{(eq)}) = 0 \tag{A.2}$$

In order to relate the LB equation with macroscopic Navier Stokes equations, it is necessary to formally separate different time scales. In this way, physical phenomena occurring at different time scales are discussed separately and contribute individually to the final equations of motion. To obtain this, the time and space derivatives are expanded using the Chapmann-Enskog expansion [10, 20], which in essence is a standard multi-scale expansion, as

$$\frac{\partial}{\partial t} = \varepsilon\frac{\partial}{\partial t_1} + \varepsilon^2\frac{\partial}{\partial t_2} \tag{A.3}$$

$$\frac{\partial}{\partial x} = \varepsilon\frac{\partial}{\partial x_1} \tag{A.4}$$

Here the expansion parameter $\varepsilon << 1$ can be identified as the Knudsen number. The Knudsen number has to be much smaller than one, in order for the treatment of the fluid as a continuous system to be valid.

The time expansion (A.3) has a simple justification as the advection and diffusion phenomena happen at different time scales and the idea behind this expansion is that all involved terms ($\partial t_1$and ∂t_2) are of the same order of magnitude, and the smallness is introduced via the parameter ε. As advection is faster than the diffusion therefore t_1 describes advection and t_2 describes the diffusion scale.

The distribution function f_α is also expanded using the expansion parameter ε as follows

$$f_\alpha = f_\alpha^{(0)} + \varepsilon f_\alpha^{(1)} + \varepsilon^2 f_\alpha^{(2)} + \mathcal{O}(\varepsilon^3) \tag{A.5}$$

Inserting Eqs. (A.3)–(A.5) into Eq. (A.2), we obtain:

$$\varepsilon^2\Big[\delta t\Big\{\frac{\partial f_\alpha^{(1)}}{\partial t_1}+\frac{\partial f_\alpha^{(0)}}{\partial t_2}+e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}\Big\}+\frac{(\delta t)^2}{2}\Big\{\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1^2}+2e_{\alpha k}\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1\partial x_{1k}}+e_{\alpha k}e_{\alpha n}\frac{\partial^2 f_\alpha^{(0)}}{\partial x_{1k}\partial x_{1n}}\Big\}$$
$$+\frac{1}{\tau}f_\alpha^{(2)}\Big]+\varepsilon\Big[\delta t\frac{\partial f_\alpha^{(0)}}{\partial t_1}+\delta t e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}+\frac{1}{\tau}f_\alpha^{(1)}\Big]=-\frac{1}{\tau}\Big[f_\alpha^{(0)}-f_\alpha^{(eq)}\Big] \tag{A.6}$$

Collecting terms in the ascending order of ε;

$$\mathcal{O}(\varepsilon^0): \qquad f_\alpha^{(0)}=f_\alpha^{(eq)} \tag{A.7}$$

$$\mathcal{O}(\varepsilon^1): \qquad f_\alpha^{(1)}=-\tau\delta t\Big[\frac{\partial f_\alpha^{(0)}}{\partial t_1}+e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}\Big] \tag{A.8}$$

$$\mathcal{O}(\varepsilon^2): \qquad f_\alpha^{(2)}=-\tau\delta t\Big[\frac{\partial f_\alpha^{(1)}}{\partial t_1}+\frac{\partial f_\alpha^{(0)}}{\partial t_2}+e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}\Big] \tag{A.9}$$
$$-\tau\frac{(\delta t)^2}{2}\Big[\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1^2}+2e_{\alpha k}\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1\partial x_{1k}}+e_{\alpha k}e_{\alpha n}\frac{\partial^2 f_\alpha^{(0)}}{\partial x_{1k}\partial x_{1n}}\Big]$$

Using Eq. (A.8) into (A.9) and performing some algebra, we get;

$$\mathcal{O}(\varepsilon^2): \qquad f_\alpha^{(2)}=-\tau\delta t\frac{\partial f_\alpha^{(0)}}{\partial t_2}+(\frac{1}{2}-\tau)\delta t\Big[\frac{\partial f_\alpha^{(0)}}{\partial t_1}+e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}\Big] \tag{A.10}$$

From Eq.(A.7), we see that f_α is not far from the equilibrium state, so we re-write Eq. (A.5) as

$$f_\alpha=f_\alpha^{(eq)}+\varepsilon f_\alpha^{(1)}+\varepsilon^2 f_\alpha^{(2)}+\mathcal{O}(\varepsilon^3) \tag{A.11}$$

The equilibrium distribution function satisfies the following constraints

$$\rho=\sum_\alpha f_\alpha^{eq} \quad , \quad \rho\mathbf{u}=\sum_\alpha \mathbf{e}_\alpha f_\alpha^{(eq)} \tag{A.12}$$

which leads to the following constraints

$$\sum_\alpha f_\alpha^k=0 \quad , \quad \sum_\alpha \mathbf{e}_\alpha f_\alpha^{(k)}=0 \quad k=1,2,\ldots \tag{A.13}$$

Multiplying Eqs. (A.8) and (A.10) by $e_{\alpha n}$ and summing over all α and using the constraints given by Eqs. (A.12) and (A.13) we obtain the following macroscopic equations:

$$\mathcal{O}(\varepsilon^1): \qquad \frac{\partial\rho}{\partial t_1}+\frac{\partial\rho u_k}{\partial x_{1k}}=0 \tag{A.14}$$

$$\mathcal{O}(\varepsilon^1): \qquad \frac{\partial\rho u_n}{\partial t_1}+\frac{\partial}{\partial x_{1k}}\sum_\alpha e_{\alpha n}e_{\alpha k}f_\alpha^{(k)}\frac{\partial f_\alpha^{(eq)}}{\partial x_{1k}}=0 \tag{A.15}$$

$$\mathcal{O}(\varepsilon^2): \qquad \frac{\partial\rho}{\partial t_2}=0 \tag{A.16}$$

$$\mathcal{O}(\varepsilon^2): \qquad \frac{\partial\rho u_n}{\partial t_2}+(1-\frac{1}{2\tau})\frac{\partial}{\partial x_{1k}}\sum_\alpha e_{\alpha n}e_{\alpha k}f_\alpha^{(k)}\frac{\partial f_\alpha^{(1)}}{\partial x_{1k}}=0 \tag{A.17}$$

Combining Eq. (A.14) and (A.16), the continuity equation is recovered to the second order of ε:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_k}{\partial x_k} = 0 \tag{A.18}$$

Combining Eq. (A.15) and (A.17), the momentum equation is recovered to the second order of ε:

$$\frac{\partial \rho u_n}{\partial t} + \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} f_\alpha^{(k)} \frac{\partial f_\alpha^{(eq)}}{\partial x_{1k}} + (1 - \frac{1}{2\tau}) \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} f_\alpha^{(k)} \frac{\partial f_\alpha^{(1)}}{\partial x_{1k}} = 0 \tag{A.19}$$

Substituting Eq. (A.18) into (A.19), the momentum equation becomes

$$\frac{\partial \rho u_n}{\partial t} + \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} \frac{\partial f_\alpha^{(eq)}}{\partial x_{1k}} + (\tau - \frac{1}{2}) \sum_\alpha \Big[e_{\alpha n} e_{\alpha k} \frac{\partial^2 f_\alpha^{(eq)}}{\partial t_1 \partial x_{1k}} + e_{\alpha n} e_{\alpha k} e_{\alpha m} \frac{\partial^2 f_\alpha^{(eq)}}{\partial x_{1k} \partial x_{1m}} \Big] = 0 \tag{A.20}$$

From eq. (A.19), we can identify the momentum flux tensor

$$\Pi_{\alpha\beta} = \Pi_{\alpha\beta}^{(0)} + \Pi_{\alpha\beta}^{(1)} = \sum_i e_{i\alpha} e_{i\beta} \Big[f_i^{(eq)} + (1 - \frac{1}{2\tau}) f_i^{(1)} \Big] = 0 \tag{A.21}$$

with the zero order momentum tensor and first order momentum tensor given by the following equations

$$\Pi_{\alpha\beta}^{(0)} = \sum_i e_{i\alpha} e_{i\beta} f_i^{(eq)} \tag{A.22}$$

$$\Pi_{\alpha\beta}^{(1)} = \sum_i e_{i\alpha} e_{i\beta} (1 - \frac{1}{2\tau}) f_i^{(1)} \tag{A.23}$$

To specify the detailed form of $\Pi_{\alpha\beta}$, the lattice structure and corresponding equilibrium distributions have to be specified. We consider $D2Q9$ model here since this is the widely used model. Derivation for other lattice structures can be obtained in a similar fashion. For the $D2Q9$ model, the following identities are observed [16].

$$\sum_\alpha w_\alpha e_{\alpha m} e_{\alpha n} e_{\alpha k} e_{\alpha j} = \frac{c^4}{9} \Big(\delta_{mn} \delta_{kj} + \delta_{mk} \delta_{nj} + \delta_{mj} \delta_{nk} \Big) \tag{A.24}$$

$$\frac{\partial}{\partial x_{1m}} \frac{\partial}{\partial x_{1k}} \sum_\alpha w_\alpha e_{\alpha m} e_{\alpha n} e_{\alpha k} e_{\alpha j} \rho u_j = \frac{c^4}{9} \Big[\frac{\partial^2 (\rho u_n)}{\partial x_{1m} \partial x_{1m}} + 2 \frac{\partial^2 (\rho u_n)}{\partial x_{1m} \partial x_{1n}} \Big] \tag{A.25}$$

Substituting f_α^{eq} into (A.20) and using the identity (A.25), we find

$$\sum_\alpha \Big[e_{\alpha n} e_{\alpha k} \frac{\partial^2 f_\alpha^{(eq)}}{\partial t_1 \partial x_{1k}} + e_{\alpha n} e_{\alpha k} e_{\alpha m} \frac{\partial^2 f_\alpha^{(eq)}}{\partial x_{1k} \partial x_{1m}} \Big] = \frac{c^2}{3} \Big[\frac{\partial^2 (\rho u_n)}{\partial x_{1m} \partial x_{1m}} + 2 \frac{\partial^2 (\rho u_n)}{\partial x_{1m} \partial x_{1n}} \Big] \tag{A.26}$$

Substituting eq.(A.26) back into eq. (A.20), the momentum equation becomes

$$\frac{\partial \rho u_n}{\partial t} + \frac{\partial \rho u_k u_n}{\partial x_{1k}} = -\frac{\partial p}{\partial x_{1n}} + (\tau - \frac{1}{2})\frac{c^2}{3}\left[\frac{\partial^2(\rho u_n)}{\partial x_{1m}\partial x_{1m}} + 2\frac{\partial^2(\rho u_n)}{\partial x_{1m}\partial x_{1n}}\right] \tag{A.27}$$

where the pressure p ans kinematic viscosity ν are given by:

$$p = c_s^2 \rho = \frac{\rho}{3} \tag{A.28}$$

$$\nu = c_s^2(\tau - \frac{1}{2}) \tag{A.29}$$

Therefore, the LBE recovers the Navier-Stokes equations in the incompressible flow limit

$$\frac{\partial u_\alpha}{\partial x_\alpha} = 0 \tag{A.30}$$

$$\frac{\partial u_\beta}{\partial t} + u_\alpha \frac{\partial u_\beta}{\partial x_\alpha} = -\frac{1}{\rho}\frac{\partial p}{\partial x_\beta} + \nu \nabla^2 u_\beta \tag{A.31}$$

B

Newton Method

In this appendix, we present the widely used method to solve the system of nonlinear algebraic equations, namely, the Newton method which is well-known for its quadratic convergence. We consider a system of nonlinear algebraic equations of the form

$$\mathcal{F}(\mathbf{x}) = 0, \tag{B.1}$$

where

$$\mathcal{F}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix}, \tag{B.2}$$

for each $f_i : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $i = 1, \ldots, n$. To apply the Newton method for approximating the solution of problem (B.1), we define the following Jacobian matrix

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \ldots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \ldots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \ldots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{bmatrix}. \tag{B.3}$$

Then, the Newton method can be expressed as a sequence of approximations

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \sigma \Big[J(\mathbf{x}^{(k)}) \Big]^{-1} \mathcal{F}(\mathbf{x}^{(k)}) \tag{B.4}$$

where $\sigma > 0$ is damping parameter. The Newton's method is generally expected to converge quadratically, if the initial approximation $\mathbf{x}^{(0)}$ is sufficiently close the exact solution $\mathbf{x}^\star$ and the inverse of the Jacobian matrix $J(\mathbf{x})$ exits. In practice, we do not actually compute the inverse of the Jacobian matrix, but instead solve a linear system of equations

$$J(\mathbf{x}^k)\Delta\mathbf{x}^{(k)} = -\mathcal{F}(\mathbf{x}^{(k)}). \tag{B.5}$$

The step by step procedure in the Newton's implementation is given as follows:

1. Given an initial guess $\mathbf{x}^{(0)}$, set $k = 0$
2. Solve the linear subproblem
$$J(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = -\mathcal{F}(\mathbf{x}^{(k)}).$$
3. Update the iterate to obtain
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \sigma\Delta\mathbf{x}^{(k)}$$
4. Set $k = k+1$, Go to step 2.

This procedure should be repeated until the norm of $\mathcal{F}(\mathbf{x}^{(k)})$ is below a certain threshold.

Inexact Newton

In order to apply pure Newton method it is required to solve the linear system given in B.5 exactly. The techniques based on Gaussian elimination or another type of factorization of the coefficient matrix can be expensive when the number of variables is large. Instead, if the linear system B.5 is solved by using an iterative method, then the Newton method is called inexact Newton since we are not solving the linear problems exactly. The schemes are identified as Newton-Jacobi, Newton-SOR or Newton-Krylov methods, according to the iterative process that is used for the solution of linear systems.

C

Discontinuous Galerkin formulation for LBE

Regarding our future research, we would like to extend our Finite Difference upwind approach presented in this thesis using higher order finite elements. Our aim is the development of similar monolithic, off lattice discretization schemes but with higher accuracy. One choice is to use standard high order Finite Volume schemes on general grids but they are not suitable due to extended stencils. Therefore, we chose Discontinuous Galerkin approach.

- Consider a general continuity equation of type

$$\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{F}(q) = 0$$

- Boltzmann-Transport $\mathbf{F}(q) = \boldsymbol{\xi}_i \cdot q$
  - Simplest physical flux with constant characteristic
- Continue with standard FEM
  - Multiply with test function ω
  - Integrate over control volume Ω
  - Apply Gaussian Formula (partial integration)

$$\int_\Omega \omega \frac{\partial q}{\partial t}\, d\mathbf{x} - \int_\Omega \nabla\omega \cdot \mathbf{F}(q)\, d\mathbf{x} + \int_{\partial\Omega} \omega \mathbf{F}(q) \cdot \mathbf{n}\, d\mathbf{s} = 0$$

- Standard high order Finite Volume $\Rightarrow$ exploding stencil
- Therefore, apply Discontinuous Galerkin
- Discontinuity $\Rightarrow$ no uniqueness of solution along the edge
- Introduce mumerical flux $\hat{F}(q^-, q^+, \mathbf{n})$

$$\int_\Omega \omega \frac{\partial q}{\partial t}\, d\mathbf{x} - \int_\Omega \nabla\omega \cdot \mathbf{F}(q)\, d\mathbf{x} + \int_{\partial\Omega} \omega \hat{F}(q^-, q^+, \mathbf{n})\, d\mathbf{s} = 0$$

- For example central flux $\Rightarrow$ not stable

- Lax-Wendroff, Godurov etc. for our problem yield all upwind flux

$$\hat{F}(q^-, q^+, \mathbf{n}) = \begin{cases} (\mathbf{v}\cdot\mathbf{n})q^- & \mathbf{v}\cdot\mathbf{n} \geq 0 \\ (\mathbf{v}\cdot\mathbf{n})q^+ & \mathbf{v}\cdot\mathbf{n} < 0 \end{cases}$$

  Also, in this case information flows in one direction between neighbouring elements so we can apply the same topological sorting algorithm, but instead of sorting the nodes we sort elementwise which gives lower triangular block matrices for transport in every direction.

- Choice of basis functions $T0$ or $Q0$ comparable to constant Finite Volumes
- Q1, Q2 bi-linear, bi-quadratic
- Alternative Taylor-basis (compare Taylor expansion)
- Number of unknowns per quad-element:
  - Q1 – 4 (corners)
  - Q2 – 9 (corners and midpoints)
  - T1 – 3 (all in midpoint)
  - T2 – 6 (all in midpoint)
- Direct solution of transport for any basis function
- Example, mesh for flow around cylinder, before and after sorting for $\mathbf{e}_2 = (1,1)^T$

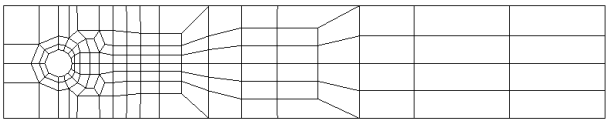


Figure C.1: mesh for the flow around cylinder benchmark

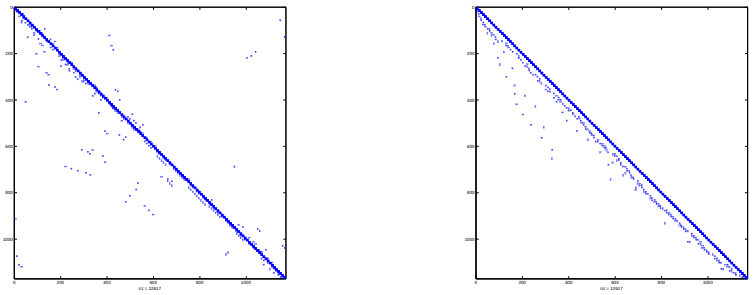


Figure C.2: affect of sorting algorithm

- Easily applicable p-adaptivity, local solutions, no steady dependence over edges

Results for Poiseuille flow on unit square, Ladd vs. Zou-He boundary conditions

Exact solution (up to machnie accuracy) for Zou-He bc's with Q2 on a single element (and higher levels, same for T2):

Q2	c=1	c=2	c=4	c=8
level 1	5.4E-09	6.9E-09	1.1E-08	8.2E-09

Table C.1: Poiseuille flow: L2-error with Zou-He boundary scheme

Results with Ladd bc's (only error of $O(\mathrm{Ma}^2)$, not h):

Q2	c=1	c=2	c=4	c=8	c=16	c=32	c=64	c=128
level 1	3.5E-04	1.1E-04	3.0E-05	7.8E-06	2.0E-06	5.1E-07	1.3E-07	3.0E-08
T2	c=1	c=2	c=4	c=8	c=16	c=32	c=64	c=128
level 1	2.9E-04	7.8E-05	2.0E-05	5.1E-06	1.3E-06	3.2E-07	9.2E-08	2.3E-08

Table C.2: Poiseuille flow: L2-error with Ladd boundary scheme

Again better Zou-He bc's, now basis Q1 (only influence of h, same for T1):

Q1	c=1	c=2	c=4	c=8
level 1	7.5E-02	7.5E-02	7.5E-02	7.5E-02
level 2	2.3E-02	2.4E-02	2.4E-02	2.5E-02
level 3	5.6E-03	5.7E-03	5.8E-03	5.9E-03
level 4	1.3E-03	1.4E-03	1.4E-03	1.4E-03
level 5	3.3E-04			
level 6	8.1E-05			
level 7	2.0E-05			
level 8	5.0E-06			

Table C.3: Poiseuille flow: L2-error with Zou-He boundary scheme, influence of h

Results for flow around cylinder: stationary benchmark with Re=20

The first results are given for the flow around cylinder benchmark [58] for the stationary case at $Re = 20$. The geometry of this benchmark is indicated in figure C.1. Here, the domain Ω consists of a channel of height $H = 0.41$ and length $L = 2.2$ having a circular cylinder located at $(0.2, 0.2)$ with diameter $D = 0.1$. The Reynolds number Re determining the flow is defined as $Re = \frac{U_{mean}D}{\nu}$ in which $U_{mean} = \frac{2}{3}U_{max}$. The value of the kinematic viscosity is set to $\nu = 10^{-3}$ and $U_{max} = 0.3$ giving a value of $Re = 20$.

		c=1		c=2		c=4	
	# EL	*Drag*	*Lift*	*Drag*	*Lift*	*Drag*	*Lift*
T0	130	22.04	5.552E-01	42.01	7.640E-01	81.86	1.171E+00
	520	17.92	3.178E-01	31.97	4.854E-01	59.91	7.581E-01
	1080	12.21	2.176E-01	19.78	3.461E-01	34.82	4.960E-01
	4320	8.871	9.288E-02	12.68	1.601E-01	20.30	2.303E-01
T2	130	5.851	1.695E-02	6.219	2.184E-02	6.867	3.854E-02
	520	5.539	9.567E-03	5.617	9.558E-03	5.697	9.748E-03
	1080	5.514	1.055E-02	5.552	1.037E-02	5.559	1.059E-02
	4320	5.522	1.060E-02	5.559	1.048E-02	5.564	1.038E-02

Table C.4: Drag and Lift for Benchmark, Reference $c_D = 5.5795$, $c_L = 1.06$E-02

Outlook for DG implementation

- Monolithic steady approach using DG
- Extension to 3D flow problems
- Higher order basis functions
- p-adaptivity
- Limiter

Bibliography

[1] U. Aaltosalmi. *Fluid Flow in Porous Media with the Lattice-Boltzmann Method*. PhD thesis, University of Jyväskylä, Finland, 2005.

[2] E. Aharonov and D. H. Rothman. Non-Newtonian flow through porous media: a Lattice Boltzmann method. *Geophysical Research Letters*, 20:679–682, 1993.

[3] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[4] P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases I. Small amplitude processes in charged and neutral one-component system. *Physical Review*, 94:511–525, 1954.

[5] R. B. Bird, R. C. Armstrong, and O. Hassager. *Dynamics of Polymer Liquids*, volume 1. Prentice Hall International, 2nd edition, 1987.

[6] J. Boyd, J. Buick, and S. Green. A second order accurate lattice Boltzmann non-Newtonian flow model. *Journal of Physics A 39*, 46:14241, 2006.

[7] D. Braess. *Finite Elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge ; New York : Cambridge University Press, 3rd edition, 2007.

[8] A. Caiazzo. Analysis of lattice Boltzmann initialization routines. *J. Ststist.*, 121:37, 2005.

[9] A. Cancelliere, C. Chang, E. Foti, D. Rothman, and S. Succi. The permeability of a random medium: Comparison of simulation with theory. *Phys. Fluids A*, 2(12): 2085, 1990.

[10] S. Chapman. On the law of distribution of molecular velocities, and on the theory of viscosity and thermal conduction, in a non-uniform simple monatomic gas. *Phil. Trans. Roy. Soc.*, A216:279–348, 1916.

[11] S. Chapman and T. G. Cowling. *The Mathematical Theory of non uniform Gases*. Cambridge University Press, 1970.

[12] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech*, 30:329–364, 1998.

[13] S. Chen, H. Chen, D. Martínez, and W. Matthaeus. Lattice Boltzman model for simulation of magnetohydrodynamics. *Phys. Rev. Lett.*, 67(27):3776, 1991.

[14] Y. Chen, H. Ohashi, and M. Akiyama. Heat transfer in lattice BGK modeled fluid. *J. Statist. Phys.*, 81:71, 1995.

[15] S. S. Chikatamarla, C. E. Frouzakis, I. V. Karlin, A. G. Tomboulides, and K. B. Boulouchos. Lattice Boltzmann method for direct numerical simulation of turbulent flows. *J. Fluid. Mech.*, 256:298–308, 2010.

[16] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, 1998.

[17] C. Cuvelier, A. Segal, and A. A. van Steenhoven. *Introduction to the Finite Element Method*. D. Reidel Publishing Company, 1986.

[18] N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice Hall International, 1974.

[19] A. Düster, L. Demkowicz, and E. Rank. High-order finite elements applied to the discrete Boltzmann equation. *Int. J. of Num. Meth. in Eng.*, 67:1094–1121, 2006.

[20] D. Enskog. *Kinetische Theorie der Vorgänge in mässig verdünnten Gasen*. PhD thesis, Uppsala, 1917.

[21] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice Gas Automata for the Navier-Stokes equation. *Physical Review Letters*, 56:1505–1508, 1986.

[22] U. Frisch, D. d'Humières, B. Hasslacher, and P. Lallemand. Lattice gas hydrodynamics in two or three dimensions. *Complex Systems*, 1:75–136, 1987.

[23] S. Gabbenelli, G. Drazer, and J. Koplik. Lattice Boltzmann method for non-newtonian (power law) fluids. *Physical Review E*, 72:046312, 2005.

[24] M. Geveler, D. Ribbrock, S. Mallach, D. Göddeke, and S. Turek. A Simulation Suite for Lattice Boltzmann based on Real-Time CFD Applications exploiting Multi-level Parallelism on Modern Multi-and Many-Core Architectures. *Journal of Computational Science*, 2:113–123, 2011.

[25] D. Grunau, S. Chen, and K. Eggert. A Lattice Boltzmann model for multiphase fluid flows. *Phys. Fluids A5*, 10:2557, 1993.

[26] A. Gunstensen and D. Rothmann. Lattice Boltzmann model of immiscible fluids. *Phys. Rev. A*, 8:4320, 1991.

[27] A. Gunstensen and D. Rothmann. Lattice Boltzmann studies of immiscible two-phase flow through porous media. *J. Geophys. Res.*, 98:6431, 1993.

[28] Z. Guo and T. S. Zhao. Lattice Boltzmann model for incompressible flows through porous media. *Physical Review E*, 66:036304, 2002.

[29] X. He and L. S. Luo. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *J. of Stat. Phys.*, 88:927–944, 1997.

[30] X. He and L. S. Luo. A priori derivation of the lattice Boltzmann equation. *Phys. Rev. E*, 55:R6333–R6336, 1997.

[31] R. Herbin and O. Labergerie. Finite volume schemes for elliptic and elliptic-hyperbolic problems on triangular meshes. *Comp. Meth. Appl. Mech. Engin.*, 147: 85–103, 1997.

[32] T. Hübner. Spezielle diskretisierungs- und lösungsmethoden für integro-differentialgleichungen am beispiel der strahlungstransportgleichung, Diploma thesis, May 2005.

[33] T. Hübner. *A Monolithic, Off-Lattice Boltzmann approach to the Discrete Boltzmann Equation with Fast and Accurate Numerical Methods*. PhD thesis, Universität Dortmund, 2011.

[34] T. Hübner and S. Turek. An efficient and accurate short-characterisctics solver for radiative transfer problems. *Computing*, 81:281–296, 2007.

[35] T. Hübner and S. Turek. Efficient monolithic simulation techniques for the stationary lattice Boltzmann equation on general meshes. *Computing and Visualization in Science*, 13(3):129–143, 2010.

[36] W. S. Jiaung, J. R. Ho, and C. P. Kuo. Lattice Boltzmann method for heat conduction problem with phase change. *Numerical Heat Transfer Part B*, 39:167–187, 2001.

[37] D. Kehrwald. Lattice Boltzmann simulation of shear thining fluids. *Journal of Statistical Physics*, 121:223, 2005.

[38] A. Ladd. Numerical simulations of particulate suspensions via a discretised Boltzmann equation. *Journal of Fluid Mechanics*, 271:285–309, 1994.

[39] P. Lallemand and L. S. Luo. Theory of the lattice boltzmann method:dispersion, dissipation, isotropy, galilean invarience, and stability. *Phys. Rev. E*, 61(6):6546–6562, 2000.

[40] R. J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[41] Y. Li, E. LeBoeuf, and P. K. Basu. Least-squares finite element scheme for the lattice Boltzmann method on an unstructured mesh. *Phys. Rev. E*, 72:046711, 2005.

[42] O. Malaspinas, G. Courbebaisse, and M. O. Deville. Simulation of generalized Newtonian fluids with the lattice Boltzmann method. *Int. J. of Mod. Phys. C*, 18: 1939–1949, 2007.

[43] R. Mei, D. Yu, W. Shyy, and L. S. Luo. Force evaluation in the lattice boltzmann method involving curved geometry. *Physical Review E*, 65:041203, 2002.

[44] R. Mei, L. S. Luo, P. Lallemand, and D. d'Humières. Consistent initial conditions for lattice Boltzmann simulations. *Computer and Fluids*, 35:855, 2006.

[45] A. R. Mitchell and D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations*. Wiley, 1980.

[46] A. A. Mohammad, M. El-Ganoui, and R. Bennacer. Lattice Boltzmann simulation of natural convection in an open ended cavity. *International Journal of Thermal Sciences*, 48:1870–1875, 2009.

[47] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations, An Introduction*. Cambridge University Press, 2005.

[48] G. Peng, H. Xi, and C. Duncan. Lattice Boltzmann method for fluid flows. *Phys. Rev. E*, 58:R4124, 1998.

[49] T. Pohl. *High Performance Simulation of Free Surface Flows Using the Lattice Boltzmann Method*. PhD thesis, Universität Erlangen -Nürenberg, 2008.

[50] Y. H. Qian, D. d'Humières, and P. Lallemand. Lattice BGK models for Navier-Stokes equation. *Europhys. Letters*, 17:479–484, 1992.

[51] S. S. Rao. *The Finite Element in Engineering*. Elsevier Science and Technology Books, 4th edition, 2004.

[52] M. Reider and J. Sterling. Accuracy of discrete velocity BGK models for the solution of the incompressible Navier-Stokes equations. *Computer and Fluids*, 24:459–467, 1995.

[53] D. Ribbrock, M. Geveler, D. Göddeke, and S. Turek. Performance and accuracy of Lattice-Boltzmann kernels on multi- and manycore architectures. *Journal of Computational Science*, 1:239–247, 2010.

[54] R. D. Richtmyer and K. W. Morton. *Difference Methods for Initial-value Problems*. Wiley-Interscience, 1967.

[55] D. H. Rothman and S. Zaleski. *Lattice-Gas Cellular Automata: Simple Models of Complex Hydrodynamics*. Cambridge University Press, 1997.

[56] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. ISBN 0898715342.

[57] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[58] M. Schäfer and S. Turek. Benchmark computations of laminar flow around cylinder. *Notes on Numerical Fluid Mechanics*, 52:547–566, 1996.

[59] P. A. Skordos. Initial and boundary conditions for the lattice Boltzmann method. *Phys. Rev. E*, 48:4823–4842, 1993.

[60] G. Strang. *Linear Algebra and Its Applications*. Brooks Cole, 3 edition, February 1988. ISBN 0155510053.

[61] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*. Prentice Hall, 1973.

[62] S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, 2001.

[63] N. Thürey, K. Iglberger, and U. Rüde. Free surface flows with moving and deforming objects for LBM. *Proceedings of Vision, Modelling and Visualization*, pages 193–200, 2006.

[64] J. Tölke and M. Krafczyk. TeraFLOP computing on a desktop pc with gpus for 3d cfd. *International Journal of Computational Fluid Dynamics*, 22:443–456, 2008.

[65] J. Tölke, M. Krafczyk, and E. Rank. A multigrid-solver for the discrete boltzmann-equation. *J. Stat. Phys.*, 107(1/2):573–591, 2002.

[66] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997. ISBN 0898713617.

[67] S. Turek. FEATFLOW: Finite Element software for the incompressible Navier-Stokes equations: User manual (www.featflow.de). *University of Dortmund*, 2000.

[68] S. Turek, L. Rivkind, J. Hron, and R. Glowinski. Numerical study of a modified time-stepping θ-scheme for incompressible flow simulations. *Computing and Visualization in Science*, 28:533–547, 2006.

[69] S. Ubertini and S. Succi. Recent advances of lattice Boltzmann techniques on unstructured grids. *J. of Stat. Phys.*, 88:927–944, 1997.

[70] H. A. van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13: 631–644, March 1992. ISSN 0196-5204.

[71] Peng Yuan. *Thermal Lattice Boltzmann Two Phase Flow Model for Fluid Dynamics*. PhD thesis, University of Pittsburgh, 2005.

[72] J. G. Zhou. A lattice Boltzmann model for the shallow water equations with turbulence modelling. *International Journal of Modern Physics C*, 13:1135–1150, 2002.

[73] J. G. Zhou. *Lattice Boltzmann Methods for Shallow Water Flows*. Springer-Verlag, 2004.

[74] Q. Zou and X. He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598, 2007.