A Monolithic, Off-Lattice Approach to the Discrete Boltzmann Equation with Fast and Accurate Numerical Methods

---

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Thomas Hübner

A Monolithic, Off-Lattice Approach to the Discrete Boltzmann Equation with Fast and Accurate Numerical Methods

Thomas Hübner

Dissertation eingereicht am: 05.10.2010
Tag der mündlichen Prüfung: 09.03.2011

Mitglieder der Prüfungskommission

Prof. Dr. Stefan Turek (1. Gutachter, Betreuer)
Prof. Dr. Heribert Blum (2. Gutachter)
Prof. Dr. Rudolf Scharlau
Prof. Dr. Christoph Buchheim
Dr. Hilmar Wobker

Acknowledgements

First and foremost I feel grateful to Professor Stefan Turek for his guidance which began early in my studies. After supervising my diploma thesis he introduced me to the field of the Boltzmann equation. Although the outcome was unpredictable, he entrusted me with this interesting but challenging task. Professor Turek was always ready to share his invaluable experience and we had many fruitful discussions. Nevertheless, his intention was to inspire independent research, asking essential questions and giving helpful hints in time. He is providing an enjoyable working atmosphere, taking care of professional and social aspects even beyond his tight time schedule.

I would also like to thank Professor Heribert Blum for accepting to review my work and also for accompanying my studies in the background with his friendly nature.

I thank the members of the Institute of Applied Mathematics, who are without exception always helpful and together provide a vast pool of knowledge and support. Mentioning here only few individuals would inevitably mean disregarding the remaining ones.

I am thankful for the support by the DFG (Deutsche Forschungsgemeinschaft; TU102/16-1) and the BMBF (Bundesministerium für Bildung und Forschung; 01IH08003D / SKALB).

Finally, I would like to express my deep gratitude to my family who supported me and showed much patience in quite difficult moments of my life. I hope I will be able to fully repay their kindness and understanding.

Dortmund, October 5, 2010

Thomas Hübner

Contents

1

Introduction

The following thesis is, in general, concerned with numerical techniques for the Boltzmann equation

$$\frac{\partial f(t,\mathbf{x},\boldsymbol{\xi})}{\partial t} + \boldsymbol{\xi}\cdot\nabla_{\mathbf{x}} f(t,\mathbf{x},\boldsymbol{\xi}) = \Omega(f). \tag{1.1}$$

Above differential equation is continuous in time, space and microscopic velocity-space and describes transport of distributions f that undergo certain collisions $\Omega(f)$ modeled by a complicated integral term over the so-called velocities $\boldsymbol{\xi}$. For a numerical treatment of (1.1), at first Ω is approximated by a special kernel, then a finite set of velocities $\boldsymbol{\xi}_i,\ i = 0,\ldots,N-1$ is used, resulting in the discrete velocity model (DVM):

$$\frac{\partial f_i(t,\mathbf{x})}{\partial t} + \boldsymbol{\xi}_i\cdot\nabla f_i(t,\mathbf{x}) = -\frac{1}{\tau}(f_i(t,\mathbf{x}) - f_i^{eq}(t,\mathbf{x})) \qquad i = 0,\ldots,N-1 \tag{1.2}$$

This semi-discrete form of the Boltzman equation consists, on the one hand, of a differential (transport) term in space along N constant characteristics — in two dimensions usually a special set of 9 velocity directions is used, denoted as D2Q9 model. On the other hand, the above Bhatnagar-Gross-Krook (BGK) approximation of Ω describes single-time relaxation of the distributions f_i towards equilibrium on a typical timescale τ. Other collision models are possible, in general they introduce a local, nonlinear coupling of the specific distributions in velocity-space.
When discussing the Boltzmann equation, commonly the Lattice Boltzmann method (LBM) is assumed foremost, because of its recent huge impact on the CFD community. However, the LBM is just a standard discretisation of (1.2), wherein the total derivative $D_t = \partial_t + \boldsymbol{\xi}\cdot\nabla$ is discretised by explicitly coupling space and time derivatives. The algorithm is described by two characteristic, decoupled sub-steps:

$$f_i^{\star}(t+\Delta t,\mathbf{x}) = f_i(t,\mathbf{x}) - \frac{\Delta t}{\tau}(f_i(t,\mathbf{x}) - f_i^{eq}(t,\mathbf{x})) \tag{1.3}$$

$$f_i(t+\Delta t,\mathbf{x}+\Delta t\boldsymbol{\xi}_i) = f_i^{\star}(t+\Delta t,\mathbf{x}) \tag{1.4}$$

After local collisions in step (1.3), the new distributions are translated to neighbouring nodes in step (1.4). The method can be regarded as an operator-splitting of Marchuk-Yanenko type which can be carried out with high computational efficiency on uniform grids. However, the explicit nature of this method causes tight restrictions concerning the standard Courant Friedrichs Levy (CFL) condition and stability, as for instance discussed in [28]. The motivation of our work is to overcome these drawbacks using an implicit approach which allows to choose larger timesteps and to focus on accuracy while not struggling with stability. Also, we want to avoid the gilded cage of the on-lattice algorithm and be allowed to use arbitrary discretisations and grids. Obviously, (1.2) is a system of partial differential equations (PDE), and talking about modern Numerics for PDEs one has to be aware of recent advances w.r.t. implicit time discretisations, unstructured,

locally adapted grids and fast iterative solvers: In combination with high order methods in space and time, they can provide more accurate results with few grid points and less time steps. A fully implicit, monolithic approach even allows direct stationary solvers in an efficient and robust way which is the focus of this thesis.

The LBM is extensively discussed in the literature concerning, for example, stability [28], model extensions [9], boundary treatment [27], [51] or its connection to the Navier-Stokes equations [17], [36], [37]. The method's potential was demonstrated by benchmark computations for laminar flows in [15], especially in comparison with the Finite Element method. On the other hand, an implicit treatment of the DVM appears in only very few places. In [43] Tölke has applied an upwind finite difference discretisation of second order on structured grids and used multigrid to solve the resulting linear equation systems. In [31] Mavriplis also took advantage of the multigrid method after trying various linear iterative solvers and encountering problems due to the high stiffness of the system and bad conditioning for realistic configurations. In [34] Noble introduced an implicit discretisation of the LBM and successfully treated the resulting nonlinearities with the Newton method. For the resulting sparse banded matrices he used a solver from LAPACK and demonstrated for his scheme to be superior to the explicit method, with superlinear scaling of CPU time and memory demands of the Newton/LAPACK combination. It is obvious that the best direct linear solvers can perform only up to a certain memory limit, then, at the latest, iterative schemes have to take over. However, most of the mentioned authors concentrated on the structured framework, thereby staying close to the LBM.
Different authors tried to overcome the restriction of the LBM to Cartesian meshes, using Finite Element and Finite Volume methods with more or less success. In part, severe problems due to numerical instability and lack of efficiency were encountered (see [41] and the work cited therein). An interesting approach was introduced recently in [11], where a discontinuous Galerkin formulation was used, obtaining accurate results with high order ansatz polynomials. In a discussion with Düster it was stated that the idea to merge high order discretisations on unstructured meshes with implicit time integration is very appealing, but the combination has a higher complexity than the individual techniques.

The author's work originated in radiative transfer equations (RTE, see [19]), introducing a special sorting technique for higher order finite difference upwind discretisations which leads to lower triangular transport matrices. This technique yielded an efficient linear solver, using iterative Krylov-space methods preconditioned by the transport part. Even for high absorption/scattering rates, level-independent convergence rates were obtained by a *generalized mean intensity* (GMI) approach with additional preconditioning. The DVM is treated here similarly as a special (semi-discretised) integro-differential equation which consists of (linear) partial differential operators of transport-reaction type with constant characteristics. Our new approach is to convey the same FD scheme and algorithm to the DVM, taking advantage of the accurate and efficient treatment on unstructured grids but extend the discretisation for implicit time stepping schemes. Moreover, with a monolithic solver as a cornerstone of the work, our new method is supposed to play out its advantages in directly obtaining steady state solutions, in contrast to the LBM approaching it with tedious pseudo-timestepping.

In this thesis, we limit ourselves to the two dimensional DVM and apply mainly the D2Q9 model. In view of given steady state problems, the considered set of equations is independent of time and (1.2) is equivalent to the stationary form

$$\boldsymbol{\xi}_i \cdot \nabla f_i + \frac{1}{\tau}(f_i - f_i^{eq}) = 0 \qquad i = 0, \ldots, 8. \tag{1.5}$$

However, the microscopic velocities $\boldsymbol{\xi}_i = c\mathbf{e}_i$ correspond to the lattice vectors

$$\mathbf{e}_i \in \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\},$$

scaled by a parameter c which determines the speed of sound, $c_s = c/\sqrt{3}$, of the system. Obviously, this lattice is specially chosen to coincide with a Cartesian mesh — the standard configuration space of the LBM — but it has also some necessary conservation and symmetry properties (see Section 2.3). First of all, summation (quadrature) of the distributions has to yield accurately macroscopic moments of density and momentum:

$$\rho = \sum_i f_i \quad , \quad \rho\mathbf{u} = \sum_i \boldsymbol{\xi}_i f_i \tag{1.6}$$

The aforementioned equilibrium term f_i^{eq} is given in these terms as

$$f_i^{eq}(\rho, \mathbf{u}) = \omega_i \rho \left(1 + \frac{3}{c^2}(\boldsymbol{\xi}_i \cdot \mathbf{u}) + \frac{9}{2c^4}(\boldsymbol{\xi}_i \cdot \mathbf{u})^2 - \frac{3}{2c^2}\mathbf{u}^2 \right) \tag{1.7}$$

with quadrature weights ω_i equal to $\frac{4}{9}$ for the zero velocity, $\frac{1}{9}$ for the orthogonal, resp., $\frac{1}{36}$ for the diagonal velocities. A simpler, so-called 'incompressible model' that was applied in this thesis reduces the above term to a quadratic polynomial in the primary simulation variables (see Sec. 2.3.1). However, f_i^{eq} results from a truncated small velocity expansion of the collision term which introduces an error of order $O(\mathrm{Ma}^2)$ (see [12], [37]) in the Mach number $\mathrm{Ma} = U/c_s$. This is consistent with the overall compressibility error resulting from an approximation of the the incompressible Navier-Stokes equations by means of a Boltzmann discretization what is usually shown in a Chapman-Enskog analysis (see [36]). Velocity and pressure in the NSE are closely connected to the moments (1.6) while the dynamic viscosity ν is related to the relaxation time of the DVM by

$$\nu = c_s^2 \tau. \tag{1.8}$$

In the LBM above relation is modified as $\nu = c_s^2(\tau - \frac{\Delta t}{2})$ to cancel artificial viscosity produced by the method.

Applying an implicit time-discretization on nonuniform grids to Eq. (1.2) results in a nonlinear system of algebraic equations because an implicit treatment of the collision operator cannot be avoided as in the LBM. We want to emphasize that the parameter c appears as a linear scaling factor to the differential operator, and quadratic in the collision term through $\frac{1}{\tau}$. It follows that for small c the equation is dominated by convection operators, so for the linear sub-problems we expect throughout good results using iterative solvers with a direct transport preconditioning. On the other hand, for high Reynolds numbers due to low viscosity ν and especially for large c the system is collision dominated which requires additional preconditioning, motivating a new algebraic reformulation of the discretised Boltzman equation. In total, we discovered that c influences the approximation as well as the solution efficiency to a high degree, and discuss in Sec. 3.3.1 the interplay between c and h which is significant for the asympotic behaviour of the model. Up to now, few publicacions can be found that analyse the contributions of discretisation and modeling error seperately or give numerical results that cover a wide range of Mach numbers.

In the following, our thesis consists of three main parts: **Part I** deals with the theory of the Boltzmann equation and provides necessary foundations for the final discretisation methods. Chapter 2 shows step-by-step the derivation of physical models based on simplifications and small velocity

approximations of the continuous Boltzmann equation. At the same time, the somewhat confusing naming convention used in the Boltzmann community and the connection with the Navier-Stokes equations are enlightened. We show that the latter is obtained in the incompressible limit (due to low Mach number) of the Boltzman equation. A representative account of the underlying Chapman-Enskog theory is given in Section 2.4, trying to outline the advanced asymptotical analysis as simply as possible. The Chapman-Enskog ansatz using a small parameter expansion of the distribution function proved to be a basic tool in the thesis.
On a different level of abstraction, in Chapter 3 we deal with the discrete Boltzmann equation and discuss explicit/implicit discretisations on structured or unstructured grids. Dependent on the chosen approach the nature and efficiency of the resulting numerical algorithms is significantly affected. As main examples are presented two opposing approaches, for one the popular Lattice Boltzmann method, confronted with our monolithic approach which is classified as an off-lattice, collision/advection implicit scheme. The expected asymptotical behaviour (being a central theme in this thesis) of the intended finite difference discretisation is discussed in detail with implications that have to bear up against numerical verification.

Part II gives a full account of the Numerics for PDE that we applied to the discrete velocity model, from discretisation to solver aspects. In the central Chapter 4 we describe implicit time-discretisation techniques and various aspects of the constant characteristic upwinding. This space discretisation scheme is closely connected to our special sorting technique introduced in [19] to solve efficiently transport problems. We present also the boundary treatment for the Boltzmann equation and finally give the main result of our work, namely the *generalized equilibrium formulation* (GEF). This algebraic reformulation of the DVM implicitly contains the inverse transport. In dealing with stiff systems due to low Mach number, the new approach allows additional preconditioning, either by the collision part of the system matrix or by a sophisticated application of multigrid schemes. We give an overview of diverse linear and non-linear solvers we applied to the obtained discrete system in Chapters 5 and 6.

The last **Part III** includes a rigorous numerical analysis of our proposed methods. First, we validate the consistency of our Boltzmann discretisation with the Navier-Stokes equations in Chapter 7, discussing spatial accuracy using several CFD testcases. Especially, we try to comprehend the asymptotical behaviour due to the Mach number and its connection with the order of discretisation applied to the differential term. Moreover, Chapter 8 adds the aspect of temporal accuracy to our numerical analysis by presenting multiple results for nonstationary flow around cylinder at Reynolds number $\mathrm{Re} = 100$. Finally, in Chapter 9 we analyse the convergence behaviour of our linear/nonlinear solvers, with a special focus on the multigrid preconditioned GEF for the monolithic approach. We give a concluding summary of our work in Chapter 10 and look out on future work planned to extend the new methods developed in this thesis. In the Appendix we give details about numerical testcases used in this thesis. Moreover, we present separate numerical methods for the use in Boltzmann simulations, advanced force-evaluation and initial-conditions schemes which are based on the Chapman-Enskog theory. We extend our approach to the alternative D2Q7 model including numerical results for this case with reduced number of discrete velocities.

Part I

Basic Principles of the Boltzmann Equation

2

Comprehensive derivation of Boltzmann physical models

In this chapter we will describe step by step the derivation of physical models based on the continuous, time-dependent Boltzmann equation (BE). For an outline of the consecutive steps see the upper part of the derivation chart given in Figure 2.1, down to the discrete velocity model. The lower part denoted as 'Total discretisation' represents the content of the next chapter, which will treat the derivation of numerical algorithms using feasible discretisations in space and time.
In Section 2.1, we start with formal definitions of the BE, with the parameters of time, space and velocity-space. The equation is composed of transport of the distribution functions and a collision operator given by a complicated integral. Moreover, we introduce how the microscopic distributions are connected to macroscopic moments. The next step of the derivation introduces a simpler operator for the collision integral, presented in Section 2.2. Closely connected is a further simplification, giving a special equilibrium term which results from a small velocity expansion truncated in second order. In Section 2.3 we shift from the continuous to the discrete Boltzmann equation, discretising one of the given dimensions, that of the velocity-space. We obtain the so-called Discrete-Velocity-Model which actually appears as a multitude of models in 2D and 3D, this thesis is limited to two dimensions in space, though. Next, we introduce the incompressible model which is a common simplification, before we present the Chapman-Enskog analysis for the 9-velocity DVM. By summarizing this important part of theory, we try to give a comprehensible account how to derive Navier-Stokes hydrodynamics by means of the discrete Boltzmann equation, however, only in the incompressible limit of a small Mach number.
In Section 2.5, for the sake of completeness, we introduce the advanced multiple-relaxation-time model (MRT). Latter appears little straightforward, instead of a uniform rate the MRT model uses different relaxation times for the (physical) moments that are included inside the Boltzmann equation.

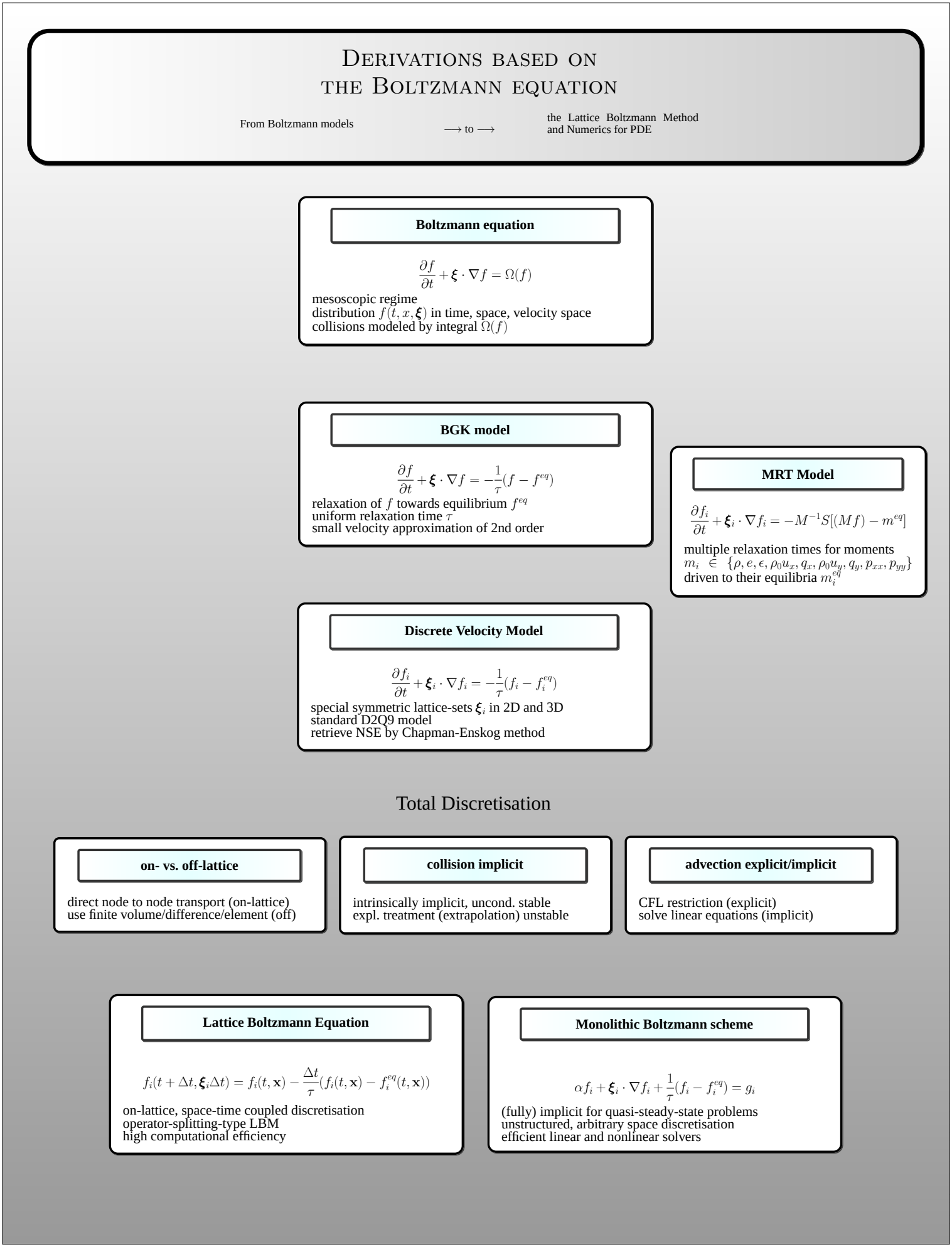


Figure 2.1: Derivation chart

2.1. Continuous Boltzmann equation

At the beginning of the derivation stands the continuous Boltzmann equation

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = \Omega(f) \tag{2.1}$$

as a kinetic model describing the behaviour of microscopic particles by means of a function

$$f(t, \mathbf{x}, \boldsymbol{\xi}).$$

But, instead of providing information about every possible particle as in population models, f describes the distribution — similar to a probability-distribution — of particles at a time t and position x moving in direction $\boldsymbol{\xi}$. The term on the left hand side of Eq. (2.1) is the total derivative $D_t = \partial_t + \boldsymbol{\xi} \cdot \nabla$ along the characteristic $\boldsymbol{\xi}$ and describes transport of the particles, on the right hand side are modeled collisions, i.e. the local interaction between distributions f with varying $\boldsymbol{\xi}$. These collisions are expressed through the quite complicated integral

$$\Omega(f) = \int \varphi(\omega)(f(\boldsymbol{\xi}')f(\boldsymbol{\xi}_1') - f(\boldsymbol{\xi})f(\boldsymbol{\xi}_1))d\omega d\boldsymbol{\xi}_1 \tag{2.2}$$

for two 'particles' with speeds $\boldsymbol{\xi}$, $\boldsymbol{\xi_1}$ before and $\boldsymbol{\xi}'$, $\boldsymbol{\xi_1}'$ after collision, while $\varphi(\omega)$ is a function of the enclosed angle. Above integral term has some specific properties, the main aspect is that several invariants are given by the equation

$$0 = \int \psi \Omega(f) d\boldsymbol{\xi} \tag{2.3}$$

which is satisfied for $\psi \in \{1, \xi_\alpha, \boldsymbol{\xi}^2\}$ with ξ_α denoting components of the velocity $\boldsymbol{\xi}$. By integration of the distribution function itself over the phase space we obtain the zeroth moment of the density

$$\rho = \int_{-\infty}^{\infty} f d\boldsymbol{\xi}, \tag{2.4}$$

resp., the first order momentum

$$\rho \mathbf{u} = \int_{-\infty}^{\infty} \boldsymbol{\xi} f d\boldsymbol{\xi} \tag{2.5}$$

with the macroscopic velocity $\mathbf{u}$. To express the equations of Navier-Stokes, higher order moments are needed. We will use mainly the second-order momentum flux tensor

$$\Pi_{\alpha\beta} \quad = \quad \int_{-\infty}^{\infty} \xi_\alpha \xi_\beta f d\boldsymbol{\xi}.$$

Taking advantage of property (2.3), resp., integrating the Boltzmann equation with the specific collision invariants ψ

$$\int_{-\infty}^{\infty} \psi \left(\frac{\partial f}{\partial t} + \xi_\beta \frac{\partial f}{\partial x_\beta} - \Omega(f) \right) d\boldsymbol{\xi} = 0$$

now yields the crucial macroscopic continuity equations:
For $\psi = 1$ we have mass conservation

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_\alpha}{\partial x_\alpha} = 0. \tag{2.6}$$

With $\psi = \boldsymbol{\xi}$ we obtain momentum conservation

$$\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta} \Pi_{\alpha\beta} \quad = \quad 0. \tag{2.7}$$

We use here and later the standard summation convention for greek indices only, mainly for the 2D case summing up two components

$$\xi_\alpha \frac{\partial f}{\partial x_\alpha} := \sum_{\alpha=1}^{2} \xi_\alpha \frac{\partial f}{\partial x_\alpha}.$$

Using normalized pressure $p = \frac{c_s^2 \rho}{\rho_0}$ in (2.6), the equation approximates divergence free flow in order $O(\mathrm{Ma}^2)$ (see [17]). However, we still need to evaluate the tensor $\Pi_{\alpha\beta}$ to obtain a closed form of the momentum equation. This last step is accomplished by the Chapman-Enskog method which is quite laborious, introducing a small parameter expansion of the distributions (see Sec. 2.4). The resulting tensors $\Pi_{\alpha\beta}^{(k)}$ in consecutive order yield the Euler equations for $k = 0$, while evaluating the tensor for $k = 1$ we can compare with the stress tensor $S_{\alpha\beta}$ in the Navier-Stokes equations.
We summarize that in its original form, the Boltzmann equation seems to be quite an abstract partial differential equation compared to the NSE, nevertheless reproducing important hydrodynamic moments by 'averaging' of the distributions. Similarly, conservative equations are obtained quite easily. Instead of working directly on the macroscopic level with the moments velocity and pressure, the BE solves for microscopic distributions in a velocity-space, therefore having an additional dimension to be considered by the discretisation.

2.2. Single Relaxation Time BGK model

Previously, we described the continuous Boltzmann equation which, in its original form, is hardly suited for an efficient numerical treatment. Therefore, we substitute the complicated collision operator Ω and then present the first approximation which introduces an error in the small Mach number limit. In the end we obtain a simpler model which is consistent with the original one and preserves important properties.

For uniform gases with $D_t f = 0$ the Boltzmann equation falls back to the integral (2.2). The reduced equation is then solved by the so-called Maxwellian (local) equilibrium distribution [4]

$$f^{(0)} = \frac{\rho}{(2\pi c_s^2)^{D/2}} \exp\left(-\frac{(\boldsymbol{\xi}-\mathbf{u})^2}{2c_s^2}\right) \tag{2.8}$$

which is a function of the microscopic velocity $\boldsymbol{\xi}$, the density ρ and velocity $\mathbf{u}$. On the way towards a simplified operator one is satisfied with reproducing the summed up behaviour of the collisions rather than microscopic details. Therefore, Bhatnagar, Gross and Krook (see [2]) introduced an operator that is simply based on the relaxation of the distributions towards the Maxwellian

$$\Omega(f) = -\frac{1}{\tau}(f - f^{(0)}).$$

The relaxation rate τ is here a small paramter, driving the distributions rapidly towards equilibrium, in context of the Navier-Stokes equations it is proportional to the dynamic viscosity ν, we will give details about the physical meaning in the next section. Compared to the complicated original form (2.2), this approach has the same invariants which is important for the derivation of continuity equations. We obtain then the so-called single relaxation time BGK model (SRTBGK)

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi}\cdot\nabla f = -\frac{1}{\tau}(f - f^{(0)}). \tag{2.9}$$

A small velocity expansion of the term (2.8) yields the equilibrium term

$$f^{eq} = \frac{\rho}{(2\pi c_s^2)^{D/2}} \exp^{\left(-\frac{\boldsymbol{\xi}^2}{2c_s^2}\right)} \left(1 + \frac{(\boldsymbol{\xi}\cdot\mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}\cdot\mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2}\right). \tag{2.10}$$

An equivalent expression for the lattice gas equation is given in [12]. Expansions with terms of higher order than $O(\mathbf{u}^2)$ are possible. For an explicit treatment of the collisions little changes would be required, but for a fully implicit treatment of Eq. (2.9) the additional terms would have to be considered in the nonlinear solver.
However, the given approximation (2.10) is consistent with the low Mach number expansion in the Chapman Enskog method for deriving the Navier Stokes equations (see Sec. 2.4). Actually, the approximation order $O(\mathrm{Ma}^2)$ will be used several times in the following steps and presents an essential limit for the derivation of Boltzmann schemes in this chapter.

2.3. Discrete Boltzmann equation

Before we give the important result of deriving the full system of macroscopic continuity equations, we describe the discretisation of the velocity-space (or phase space) which is also preceding the upcoming discretisation in time and space. We need to replace the continuous argument $\boldsymbol{\xi}$ by a discrete set $\boldsymbol{\xi}_i$ for a numerical treatment of the BGK model (2.9). The requirement for the discrete velocity set to be coupled to configuration space is understandable in view of the nature of the Lattice Boltzmann method, but it is not a must for general Boltzmann schemes. More important, though, is the aspect of accuracy: The corresponding quadrature (to approximate the integral $\int d\boldsymbol{\xi}$) must exactly preserve moments (2.4),(2.5) and conservative equations considered in the Chapman-Enskog derivation of the NSE (see [18]) . Furthermore, the chosen set of velocities must possess symmetry properties to retain the isotropy of the Navier-Stokes equations, i.e. the physical system must behave the same regardless of the orientation in space. An obvious final requirement is that for any $\boldsymbol{\xi}_i$ the opposite $-\boldsymbol{\xi}_i$ is contained in the set. Given a velocity set that conforms to all requirements, we finally obtain the discrete Boltzmann equation

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \nabla f_i = -\frac{1}{\tau}(f_i - f_i^{eq}) \quad , \quad i = 0, \ldots, N-1 \tag{2.11}$$

which is also known as the discrete velocity model (DVM). A discrete form of the equilibrium term is given by

$$f_i^{eq} = \omega_i \rho \left(1 + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2} \right)$$

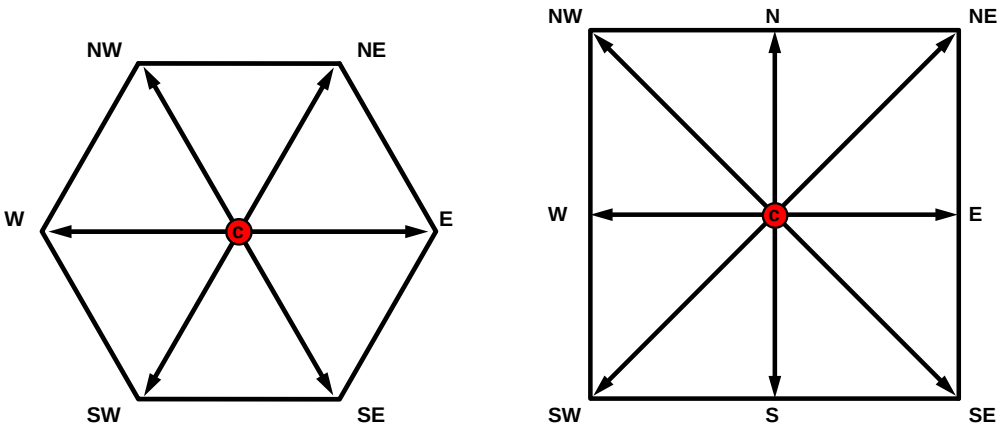
with weights ω_i resulting from quadrature restraints (for an in-depth look we refer to the literature, see [18]). Finally, the moments in discrete velocity space are obtained by the quadratures

$$\rho = \sum_i f_i \quad , \quad \rho \mathbf{u} = \sum_i \boldsymbol{\xi}_i f_i \quad , \quad \Pi_{\alpha\beta} = \sum_i \xi_{i,\alpha} \xi_{i,\beta} f_i .$$

The minimal lattice in 2D is a 6 speed model — corresponding to a uniform triangular mesh with angles of 60 as seen in Fig. 2.2a — the D2Q7 is able to approximate the Navier-Stokes equations, in contrast to the D2Q6 model which contains no rest particle. The coupling of discretisation space and configuration space will be discussed further in Sec. 3.1.1 in the context of on-lattice Boltzmann schemes. The basic model used in the LBM community is the D2Q9 model (see Fig. 2.2b), which corresponds to a Cartesian mesh. For convenience, and to step onto a common ground we applied this set in our thesis, nevertheless we implemented without difficulty also the D2Q7 model, as shown in the Appendix A3. In 3D, models are usually constructed only on a cubic square lattice. There, the number of lattice vectors is growing fast, we have D3Q13, D3Q15, D3Q19, D3Q27 models, depending which subsets of lattice vectors are chosen. The subsets are characterised by the vector lenghts ($|\boldsymbol{e}| \in \{0,1,2,3\}$), resp., the distance to the neighbouring nodes: Subset 0 is only the center. Subset 1 are the nearest neighbours situated on the 6 wall midpoints. Subset 2 are the next nearest neighbours on the 12 cube's edge midpoints. Finally, subset 3 are the vectors pointing to the 8 cube-corners.

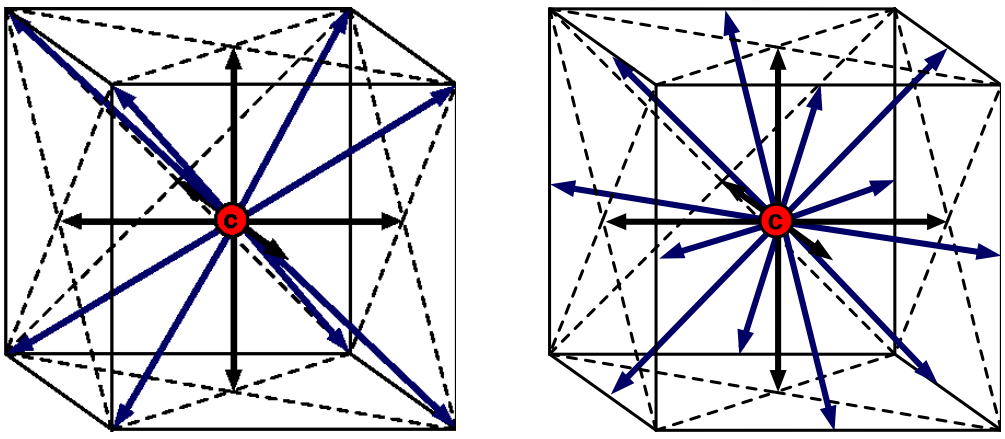
$$\boldsymbol{e}_i = \begin{cases} (0,0,0) & , \quad \text{i=0} & \textbf{subset 0} \\ (\pm 1,0,0),(0,\pm 1,0),(0,0,\pm 1) & , \quad \text{i=1,}\ldots\text{,6} & \textbf{subset 1} \\ (\pm 1,\pm 1,0),(\pm 1,0,\pm 1),(0,\pm 1,\pm 1) & , \quad \text{i=7,}\ldots\text{,19} & \textbf{subset 2} \\ (\pm 1,\pm 1,\pm 1) & , \quad \text{i=20,}\ldots\text{,27} & \textbf{subset 3} \end{cases}$$

We choose the discrete Boltzmann equation to present the ideas of the Chapman Enskog analysis, although basically the same methods can be applied to the continuous form. Moreover, we give in the following another modification of the DVM which is quite common in practice. It concerns mainly the equilibrium term but retains the overall consistency of the model.



(a) reduced D2Q7 for hexmesh (b) D2Q9 for square mesh

Figure 2.2: 2D Lattice sets



(a) D3Q15 with subsets 0, 1 and 3 (b) D3Q19 with subsets 0, 1 and 2

Figure 2.3: 3D Lattice sets

2.3.1. Incompressible model

Although it is impossible to obtain a constant density in discrete Boltzmann schemes for the simulation of incompressible flow, the changes in density are in practice very low. With the incompressible limit being defined by the Mach number going to zero, the fluctuations $\delta\rho$ of density are of second order $O(\varepsilon^2)$ in the small Knudsen number. In [17] it was introduced the ansatz to substitue $\rho = \rho_0 + \delta\rho$ for the density in the equilibrium term. Then, all expressions in terms of $\delta\rho(\mathbf{u}/c)$ and $\delta\rho(\mathbf{u}/c)^2$ are omitted, being of second or higher order in the Mach number. The new 'incompressible model' defines the macroscopic moments as

$$\rho = \sum_i f_i \quad , \quad \rho_0 \mathbf{u} = \sum_i \boldsymbol{\xi}_i f_i \tag{2.12}$$

which means a simpler momentum and an equilibrium term of the following form:

$$f_i^{eq} = \omega_i \left(\rho + \rho_0 \left(\frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2} \right) \right), \tag{2.13}$$

with the constant density usually being set to one. Especially in view of a numerical treatment of the DVM, this expression is significantly easier, nevertheless it is consistent with the small velocity expansion of the Maxwellian equilibrium. Also, the overall compressibility limit of $O(\mathrm{Ma}^2)$ will not be violated during the following Chapman-Enskog method in deriving the Navier-Stokes equations.

2.4. Chapman-Enskog expansion

This section presents a cornerstone of the theory concerning the Boltzmann equation and in the current Chapter 2. It is important in order to basically understand how the numerical solution of first oder PDEs can give the same results as obtained from CFD for the Navier-Stokes equations. The first ideas given in Section 2.1 were quite simple, we could express macroscopic quantities and basic continuity equations like mass conservation by building the zeroth and first moments defined as integrals over the microscopic velocity. However, to obtain higher order moments, necessary to approximate the Navier-Stokes equations, is more difficult and there are several approaches presented in the literature. For example, in [24] it was shown that the Lattice Boltzmann equation can be regarded as a finite difference representation of the NSE, a numerical method for macroscopic equations with stencils obtained from central differences of second order for the differential terms like divergence, gradient and Laplacian. In his work, Junk mainly uses the diffusive scaling (see [25]) for the so-called Chapman-Enskog expansion. His analysis using the relation $\Delta t = O(\Delta x^2)$ leads directly to the incompressible Navier-Stokes equations. However, the work presents advanced theory of asymptotical analysis, which needs time and study to be fully understood. We will present a simplified Chapman-Enskog method, showing the small parameter expansion of the primary distributions, but restricting ourselves to a straightforward derivation of the main macroscopic equations.

In the following we will carry out the Chapman-Enskog analysis for the 9-velocity BGK model (see also [17]). The integration over the phase space will now be replaced by quadrature which is constructed to preserve the crucial physical quantities like conservative moments and symmetries. The same holds for the equilibrium term of the incompressible model described in the previous section. The Chapman-Enskog ansatz is an expansion of the distribution function f_i with small parameter ε (in general $O(\varepsilon)$ is equivalent to $O(\mathrm{Ma})$), written as

$$f_i = f_i^{(0)} + \varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)} + \dots \tag{2.14}$$

The equilibrium $f_i^{(0)}$ will be accordingly represented by the simpler term f_i^{eq} from (2.13) in the following calculations. The first two moments obtained from summation of the series (2.14) are conservative, it means they are given fully by the equilibrium term:

$$\rho = \sum_i \left(f_i^{(0)} + \varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)} + \dots \right) = \sum_i f_i^{(0)}$$

$$\rho_0 u_\alpha = \sum_i \xi_{i,\alpha} \left(f_i^{(0)} + \varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)} + \dots \right) = \sum_i \xi_\alpha f_i^{(0)}$$

while the momentum flux tensor is nonconservative, depending therefore on the nonequilibrium functions

$$\begin{aligned} \Pi_{\alpha\beta} &= \sum_i \xi_{i,\alpha}\xi_{i,\beta} \left(f_i^{(0)} + \varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)} + \dots \right) \\ &= \sum_i \xi_{i,\alpha}\xi_{i,\beta} f_i^{(0)} + \varepsilon \sum_i \xi_{i,\alpha}\xi_{i,\beta} f_i^{(1)} + \varepsilon^2 \sum_i \xi_{i,\alpha}\xi_{i,\beta} f_i^{(2)} + \dots \\ &= \Pi_{\alpha\beta}^{(0)} + \varepsilon \Pi_{\alpha\beta}^{(1)} + \varepsilon^2 \Pi_{\alpha\beta}^{(0)} + \dots \end{aligned}$$

Starting with the zeroth order tensor, direct evaluation of $\Pi^{(0)}_{\alpha\beta}$ using f_i^{eq} results in the second and third order terms

$$\sum_i \xi_{i,\alpha}\xi_{i,\beta}f_i^{(0)} = \frac{1}{3}c^2\rho\delta_{\alpha\beta} + \rho_0 u_\alpha u_\beta \tag{2.15}$$

$$\sum_i \xi_{i,\alpha}\xi_{i,\beta}\xi_{i,\gamma}f_i^{(0)} = \frac{1}{3}c^2\rho_0(\delta_{\alpha\beta}u_\gamma + \delta_{\alpha\gamma}u_\beta + \delta_{\beta\gamma}u_\alpha). \tag{2.16}$$

Inserting the second order term (2.15) in the momentum equation (2.7) results in the Euler equations

$$\begin{aligned} \frac{\partial\rho_0 u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta}\Pi^{(0)}_{\alpha\beta} &= \\ \frac{\partial\rho_0 u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta}(c_s^2\rho\delta_{\alpha\beta} + \rho_0 u_\alpha u_\beta) &= 0. \end{aligned}$$

By extension to the first order tensor $\Pi^{(1)}_{\alpha\beta}$ one obtains the equation

$$\begin{aligned} \frac{\partial\rho_0 u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta}(\Pi^{(0)}_{\alpha\beta} + \varepsilon\Pi^{(1)}_{\alpha\beta}) &= \\ \frac{\partial\rho_0 u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta}(\rho_0 u_\alpha u_\beta + c_s^2\rho\delta_{\alpha\beta} + \varepsilon\Pi^{(1)}_{\alpha\beta}) &= 0. \end{aligned} \tag{2.17}$$

Given the stress tensor in the Navier-Stokes equations with the dynamic viscosity ν, i.e.

$$S_{\alpha\beta} = \nu\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right),$$

equation (2.17) is an approximation of the NSE of second order in the Mach number if it is possible to identify the stress tensor with

$$S_{\alpha\beta} = -\varepsilon\Pi^{(1)}_{\alpha\beta} + O(\varepsilon^2). \tag{2.18}$$

That is why it is necessary to obtain a closed form for $f^{(1)}$. As starting point of the actual analysis we take the dimensionless form of the Boltzmann equation with the index i temporarily omitted:

$$\frac{\partial\hat{f}}{\partial\hat{t}} + \hat{\boldsymbol{\xi}}\cdot\frac{\partial}{\partial\hat{x}}\hat{f} = -\frac{1}{\varepsilon}(\hat{f} - \hat{f}^{(0)})$$

The relations

$$\hat{f} = \frac{f}{n_r} \quad , \quad \hat{\boldsymbol{\xi}} = \frac{\boldsymbol{\xi}}{c_r} \quad , \quad \hat{x} = \frac{x}{L_r} \quad , \quad \hat{t} = t\frac{c_r}{L_r}$$

are determined by a typical distribution n_r, characteristic length L_r and typical microscopic velocity c_r. In general, the small parameter in the expansion is defined as the Knudsen-number $\varepsilon = \frac{c_r\tau}{L_r}$, giving the ratio of free mean path $l_r = c_r\tau$ between collisions and characteristic length L_r. Inserting the Chapman-Enskog expansion (2.14) yields

$$\begin{aligned} \frac{\partial}{\partial\hat{t}}(\hat{f}^{(0)} + \varepsilon\hat{f}^{(1)} + \varepsilon^2\hat{f}^{(2)}\ldots) + \hat{\boldsymbol{\xi}}\cdot\frac{\partial}{\partial\hat{x}}(\hat{f}^{(0)} + \varepsilon\hat{f}^{(1)} + \varepsilon^2\hat{f}^{(2)}\ldots) = \\ -\frac{1}{\varepsilon}(\hat{f}^{(0)} + \varepsilon\hat{f}^{(1)} + \varepsilon^2\hat{f}^{(2)}\ldots - \hat{f}^{(0)}). \end{aligned}$$

The next step is usually sorting above equation by terms in ε^{-1}, ε^0, ε^1, etc. This results for ε^0 in

$$\hat{f}^{(1)} = -(\frac{\partial\hat{f}^{(0)}}{\partial\hat{t}} + \hat{\boldsymbol{\xi}}\cdot\frac{\partial\hat{f}^{(0)}}{\partial\hat{x}}).$$

We obtain a closed form for $f^{(1)}$ by reverting from the dimensionless form:

$$\begin{aligned} f^{(1)} &= -(\frac{\partial f^{(0)}}{\partial t}\frac{\partial t}{\partial \hat{t}} + \frac{\boldsymbol{\xi}}{c_r}\cdot\frac{\partial f^{(0)}}{\partial \hat{x}}\frac{\partial x}{\partial \hat{x}}) \\ &= -\frac{L_r}{c_r}(\frac{\partial f^{(0)}}{\partial t} + \xi_\alpha\frac{\partial f^{(0)}}{\partial x_\alpha}) \end{aligned}$$

Now we can insert this identity in terms of $f^{(0)}$ into the first order tensor, being able to resolve equation (2.18) as

$$\begin{aligned} S_{\alpha\beta} &= -\varepsilon\Pi^{(1)}_{\alpha\beta} + O(\varepsilon^2) \\ &= \frac{L_r}{c_r}\varepsilon\sum_i \xi_{i,\alpha}\xi_{i,\beta}(\frac{\partial f_i^{(0)}}{\partial t} + \xi_{i,\gamma}\frac{\partial f_i^{(0)}}{\partial x_\gamma}) + O(\varepsilon^2) \\ &= \tau(\frac{\partial \Pi^{(0)}_{\alpha\beta}}{\partial t} + \sum_i \xi_{i,\alpha}\xi_{i,\beta}\xi_{i,\gamma}\frac{\partial f_i^{(0)}}{\partial x_\gamma}) + O(\varepsilon^2). \end{aligned}$$

We have two remaining terms on the right side, a time derivative of $\Pi^{(0)}_{\alpha\beta}$ and a space derivative of the third order moment given by (2.16), resulting in

$$\begin{aligned} S_{\alpha\beta} &= \tau(\frac{\partial \Pi^{(0)}}{\partial t} + \frac{\partial}{\partial x_\gamma}\sum_i \xi_{i,\alpha}\xi_{i,\beta}\xi_{i,\gamma}f_i^{(0)}) + O(\varepsilon^2) \\ &= \tau(\frac{\partial \Pi^{(0)}}{\partial t} + \frac{\partial}{\partial x_\gamma}c_s^2\rho_0(\delta_{\alpha\beta}u_\gamma + \delta_{\alpha\gamma}u_\beta + \delta_{\beta\gamma}u_\alpha)) + O(\varepsilon^2) \\ &= \tau(\frac{\partial \Pi^{(0)}}{\partial t} + c_s^2\rho_0(\delta_{\alpha\beta}\frac{\partial u_\gamma}{\partial x_\gamma} + \frac{\partial u_\beta}{\partial x_\alpha} + \frac{\partial u_\alpha}{\partial x_\beta})) + O(\varepsilon^2). \end{aligned}$$

The divergence $\frac{\partial u_\gamma}{\partial x_\gamma} = \nabla\cdot\mathbf{u}$, although it is of order $O(\mathrm{Ma}^2)$ due to the low compressibility limit, cannot be neglected as it is scaled by c_s^2. Instead, we will show that it corresponds up to a certain degree to the time-derivative of the zeroth-order momentum flux tensor. With the previously given definition and using a first approximation

$$\begin{aligned} \frac{\partial \Pi^{(0)}_{\alpha\beta}}{\partial t} &= \frac{\partial}{\partial t}(c_s^2\rho\delta_{\alpha\beta} + \rho_0 u_\alpha u_\beta) \\ &= c_s^2(\frac{\partial \rho}{\partial t})\delta_{\alpha\beta} + O(\mathrm{Ma}^2), \end{aligned}$$

we replace the time-derivative of density using mass conservation (2.6)

$$\begin{aligned} \frac{\partial \Pi^{(0)}_{\alpha\beta}}{\partial t} &= c_s^2(-\frac{\partial \rho u_\gamma}{\partial x_\gamma})\delta_{\alpha\beta} + O(\mathrm{Ma}^2) \\ &= c_s^2\rho_0(-\frac{\partial u_\gamma}{\partial x_\gamma})\delta_{\alpha\beta} + O(\mathrm{Ma}^2). \end{aligned}$$

This term with negative sign cancels out the divergence in second order of the Mach number and we obtain in the same order the viscous Stress tensor

$$S_{\alpha\beta} = \tau c_s^2\rho_0(\frac{\partial u_\beta}{\partial x_\alpha} + \frac{\partial u_\alpha}{\partial x_\beta}) + O(\mathrm{Ma}^2).$$

Overall, we succeeded in approximating the Navier-Stokes equations with the consistency order of $O(\mathrm{Ma}^2)$with the important practical result of identifying the dynamic viscosity as $\nu = \tau c_s^2 \rho_0$ in our simulation. The constant density ρ_0 is usually set to one. In [17], the stress tensor is obtained differently in the last steps, also the authors give a better approximation of $O(\mathrm{Ma}^3)$ in the NS momentum equation, but we find the present derivation sufficient and more clear, at the same time giving the main ideas. Besides, the compressibility error of order $O(\mathrm{Ma}^2)$ is dominating in practice.

2.5. Multiple Relaxation Time model

The previously decribed SRT model applies a uniform relaxation time which is a basic treatment in discrete Boltzmann schemes. But, instead of relaxing all distributions towards equilibrium using the same τ, a diversified look at the moments $\mathbf{m} = (m_0, \ldots, m_8)^t$ appearing inside the kinetic system is possible (as seen in [8], [9]). In the D2Q9 model the separate moments m_i are taken from

$$\{\rho, e, \varepsilon, \rho_0 u_x, q_x, \rho_0 u_y, q_y, p_{xx}, p_{xy}\} \tag{2.19}$$

with the following interpretation (see [8]):

$$\begin{array}{rl} \rho & \text{density} \\ e & \text{energy} \\ \varepsilon & \text{energy square} \\ \rho_0 u_x & \text{momentum (x-component)} \\ q_x & \text{energy flux (x-component)} \\ \rho_0 u_y & \text{momentum (y-component)} \\ q_y & \text{energy flux (y-component)} \\ p_{xx} & \text{stress tensor (diagonal entry)} \\ p_{xy} & \text{stress tensor (off-diagonal entry)} \end{array} \tag{2.20}$$

Actually, only 6 moments are necessary in the kinetic system of the Navier-Stokes equations in two dimensions, taken from the density, velocity and symmetric momentum flux tensor. The linear, regular transformation matrix M (with $\mathbf{m} = M\mathbf{f}$) consists of orthogonal eigenmodes and is given by

$$M = \begin{bmatrix} 1\cdot & (1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) \\ c^2\cdot & (-4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2) \\ c^4\cdot & (4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1) \\ c\cdot & (0 & 1 & -0 & -1 & 0 & 1 & -1 & -1 & 1) \\ c^3\cdot & (0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1) \\ c\cdot & (0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1) \\ c^3\cdot & (0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1) \\ c^2\cdot & (0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0) \\ c^2\cdot & (0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) \end{bmatrix}$$

Then, the collision step, presented exemplarily as part of the explicit two-step Lattice Boltzmann method with MRT, is carried out as follows: After transforming the f_i into the moments m_i, they are relaxed using variable collision-rates s_i towards the equilibrium moments m_i^{eq} given by

$$\begin{array}{rcl} m_0^{eq} & = & \rho \\ m_1^{eq} & = & e^{eq} = -2c^2\rho + 3\rho_0\mathbf{u}^2 \\ m_2^{eq} & = & \varepsilon^{eq} = c^4\rho + 3\rho_0\mathbf{u}^2 \\ m_3^{eq} & = & \rho_0 u_x \\ m_4^{eq} & = & q_x^{eq} = -c^2\rho_0 u_x \\ m_5^{eq} & = & \rho_0 u_y \\ m_6^{eq} & = & q_y^{eq} = -c^2\rho_0 u_y \\ m_7^{eq} & = & p_{xx}^{eq} = \rho_0(u_x^2 - u_y^2) \\ m_8^{eq} & = & p_{xy}^{eq} = \rho_0 u_x u_y. \end{array} \tag{2.21}$$

Afterwards, the relaxed values are transformed back and the resulting collisions are given by

$$\Omega = -M^{-1}S[(Mf) - m^{eq}], \tag{2.22}$$

with the relaxation rates included in $S = diag(s_0, \ldots, s_8)$. As seen from Eq. (2.21), the rates $s_0,\ s_3,\ s_5$ belong to conserved moments which cancel out anyway after collision. On the other hand, rates $s_7,\ s_8$ determine the relaxation time of the tensor Π and have to be choosen equal to $1/\tau$ in order to obtain the viscous stress of the Navier Stokes system. We can only influence the system by means of the remaining moments which are not fixed and do not cancel out.The rates $s_1,\ s_2,\ s_4,\ s_6$ can be used to improve conditioning and stability, in order to solve the system more efficiently. In case of the LBM with stability problems for τ close to $\frac{\Delta t}{2}$ (in practice $\Delta t = 1$), so for $s_7,\ s_8 \sim 2$ the adjustable parameters are set close to 1 (see [28]). However, in practice it is not trivial to find 'optimal' values for different Reynolds numbers and to search the whole parameter space is expensive (see [9]). In the framework of an implicit, resp., monolithic discretisation of the Boltzmann equation, the described step-by-step collision process has to be replaced by a system of equations. We need to calculate the alternative collision matrix $M^{-1}SM$, which, however, remains a local operator coupling all 9 local distributions $f_i(\mathbf{x}), i = 1, \ldots, 9$. We still obtain a nonlinear system, as the equilibrium term f^{eq} is replaced by the moments m^{eq}, which are quadratic polynomials in terms of the velocity for $m_1^{eq}, m_2^{eq}, m_7^{eq}, m_8^{eq}$. In our numerical tests, we will analyse at first the influence of relaxation rates smaller than $1/\tau$, but we will also give results for values bigger than $1/\tau$ and some interesting observations in Sec. 7.2.

3

Derivation of numerical Boltzmann methods

After presenting the theoretical background concerning Boltzmann models and the approximation of the Navier-Stokes equations in the previous chapter, here we will discuss practical aspects concerning efficient numerical methods. We will show that the popular Lattice Boltzmann method is just one elegant discretisation scheme for the Boltzmann equation on structured meshes, while other, more variable treatments are possible. In the following sections, several approaches for alternative discretisations are presented and our work — modern numerics for PDEs applied to the BE — is placed in the respective context. Refering to different authors, we define categories mainly based on the explicit, resp., implicit treatment of advection and collision, while structured/unstructured grids play another important role. One of the derived schemes is the actual Lattice Boltzmann equation, a special discretisation coupling space and time and thereby being an intrinsically *collision implicit* scheme. High computational efficiency is achieved by the corresponding Lattice Boltzmann method, an on-lattice algorithm suited for structured grids. We show that quite similarly are obtained off-lattice Boltzmann schemes, without difficulty avoiding the structured framework.
Another approach is characterized by implicit treatment of the transport term, yielding *collision/advection implicit* schemes. Only this class — leading to a system of coupled equations — enables to finally overcome the CFL condition. In our approach we pick up the thread given in the literature, applying special finite difference schemes on unstructured grids in combination with implicit time-discretisation which even allows a fully stationary, monolithic approach.

3.1. Collision implicit, advection explicit Lattice-Boltzmann schemes

In the last chapter, the derivation arrived at the DVM with the two variants of single and multi relaxation time models. In view of the following discretization, the models are quite similar, only the resulting local 'collision matrix' has different coefficients. Without loss of generality, we will use the SRT represantation for Ω in the following.
The next step in the derivation process is a discretisation in space and time, to this purpose we come back to the discrete (in microscopic velocity space) Boltzmann equation, which can be written using the total derivative as

$$D_t f_i = \Omega_i(f),$$

Usually, one performs time integration on the interval $[0, \Delta t]$:

$$f_i(t+\Delta t, \mathbf{x}+\Delta t\boldsymbol{\xi}_i) - f_i(t,\mathbf{x}) = \int_0^{\Delta t} \Omega_i(f(t+t', \mathbf{x}+t'\boldsymbol{\xi}_i))dt' \tag{3.1}$$

The integral on the right hand side is then approximated by

$$\Delta t[\theta\Omega_i(f(t+\Delta t, \mathbf{x}+\Delta t\boldsymbol{\xi}_i)) + (1-\theta)\Omega_i(f(t,\mathbf{x}))].$$

For the special case of discretisation on a lattice, we can use distribution functions $\tilde{f}$ *on the characteristic*. We substitute the new variables, denoted $\tilde{f}_i^n = f(t_n, \mathbf{x})$ and $\tilde{f}_i^{n+1} = f(t_n + \Delta t, \mathbf{x} + \Delta t \boldsymbol{\xi}_i)$ and obtain

$$\tilde{f}_i^{n+1} - \tilde{f}_i^n = \Delta t[\theta \tilde{\Omega}_i^{n+1} + (1-\theta)\tilde{\Omega}_i^n]. \tag{3.2}$$

The parameter $0 \leq \theta \leq 1$ must be chosen $\theta = 1/2$ to obtain second order accuracy. However, the crucial point is that one strongly wants to avoid implicit treatment of $\tilde{\Omega}_i^{n+1} = -\frac{1}{\tau}(\tilde{f}_i^{n+1} - \tilde{f}_i^{eq,n+1})$ because of the nonlinearity in the equilibrium term. One possibility is to use extrapolation as in [30] which has serious stability problems. Fortunately, He et al. (see [16]) succeded in removing the implicitness for $\theta = 1/2$, and the extended procedure by Guo et.al. (see [14]) for any choice of θ is easily accomplished by introducing a modified distribution function

$$g_i(t, \mathbf{x}) = f_i(t, \mathbf{x}) - \Delta t \theta \Omega_i(f(t, \mathbf{x})). \tag{3.3}$$

Due to invariance of Ω under quadrature, we have

$$\sum_i g_i = \sum_i f_i = \rho \quad \sum_i \boldsymbol{\xi}_i g_i = \sum_i \boldsymbol{\xi}_i f_i = \rho \mathbf{u}$$

which implies $g_i^{eq} = f_i^{eq}$. Following this approach, we can replace all terms containing θ in above Eq. (3.2) with the new variable g_i (one at time t^n and one at time t^{n+1}):

$$\tilde{g}_i^{n+1} = \tilde{g}_i^n + \Delta t \tilde{\Omega}_i^n(f)$$

The remaining, original f_i can be removed using the resorted identity (3.3), i.e.

$$f_i + \Delta t \theta \frac{1}{\tau} f_i = g_i + \Delta t \theta \frac{1}{\tau} f_i^{eq}$$

$$f_i = \frac{1}{1 + \Delta t \theta \frac{1}{\tau}} g_i + \frac{\Delta t \theta \frac{1}{\tau}}{1 + \Delta t \theta \frac{1}{\tau}} f_i^{eq}.$$

By further evolving this form, we can finally replace $\tilde{\Omega}_i^n(f)$ with the equation

$$\begin{aligned}
\frac{1}{\tau} f_i - \frac{1}{\tau} f_i^{eq} &= \frac{1}{\tau + \Delta t \theta} g_i + \frac{\Delta t \theta \frac{1}{\tau}}{\tau + \Delta t \theta} f_i^{eq} - \frac{1}{\tau} f_i^{eq} \\
&= \frac{1}{\tau + \Delta t \theta} g_i + \frac{\Delta t \theta \frac{1}{\tau}}{\tau + \Delta t \theta} f_i^{eq} - \frac{1 + \Delta t \theta \frac{1}{\tau}}{\tau + \Delta t \theta} f_i^{eq} \\
&= \frac{1}{\tau + \Delta t \theta}(g_i - f_i^{eq}) \\
&= \frac{1}{\tau + \Delta t \theta}(g_i - g_i^{eq})
\end{aligned}$$

and obtain

$$\tilde{g}_i^{n+1} = \tilde{g}_i^n - \frac{\Delta t}{\lambda}(\tilde{g}_i^n - \tilde{g}_i^{eq,n}).$$

This final form is actually the expression known as the Lattice Boltzmann equation, with the modified relaxation time $\lambda = \tau + \Delta t \theta$. But instead of obtaining λ by a tedious Chapman-Enskog analysis for the discrete equations (see Section 3.1.1), we showed more clearly the implicit character of the LBM. The collision implicit approach results in a scheme which is unconditionally stable, with the only limitation on the time step size — due to the explicit treatment of the advection term — remaining from the standard CFL condition.

3.1.1. Lattice Boltzmann Equation

Apart from the (still) uncommon ansatz using a modified distribution function, there is a classic way to obtain the standard LBE. It can be derived from the discrete Boltzmann equation using a special FD approach coupling space and time on a structured grid, the implicitness of the collisions is also omitted, but less obviously than in the previous section. Nevertheless, we want to present this approach due to its historical meaning, as it is closely connected to lattice gas cellular automata (see [12]). Remarkably, the Lattice Boltzmann equation was first derived from LGCA, before it was shown that it corresponds to the special finite difference discretisation. Another motivation for this section is to show why the viscosity in the LBM is given by

$$\nu = c_s^2(\tau - \frac{\Delta t}{2}) \tag{3.4}$$

which is different from the relation we use in our thesis and might be confusing. Naturally, the question might arise how to use relation (3.4) in a monolithic stationary approach without the simulation parameter Δt. To forestall the answer, we will show that Eq. (3.4) is valid only for the following special discretisation using finite differences in space and time. However, the applied first order schemes introduce significant numerical viscosity and loss of accuracy compared to the original DVM. We show how it can be identified using the Chapman-Enskog expansion (see Section 2.4) and finally eliminated resulting in a second order accurate scheme.

Starting from the DVM (2.11), the time derivative is discretised using a forward Euler difference quotient

$$\frac{\partial f_i(t,\mathbf{x})}{\partial t} \sim \frac{f_i(t+\Delta t,\mathbf{x}) - f_i(t,\mathbf{x})}{\Delta t}.$$

The transport is discretised at time $t+\Delta t$ by an upwind scheme on a spatial grid spanned by the lattice vectors and scaled by the particle speed

$$\boldsymbol{\xi}_i \cdot \nabla f_i(t+\Delta t,\mathbf{x}) \sim c\frac{f_i(t+\Delta t,\mathbf{x}+\Delta x\mathbf{e}_i) - f_i(t+\Delta t,\mathbf{x})}{\Delta x}.$$

Substituting the above terms in the DVM equation yields

$$\frac{f_i(t+\Delta t,\mathbf{x}) - f_i(t,\mathbf{x})}{\Delta t} + c\frac{f_i(t+\Delta t,\mathbf{x}+\Delta x\mathbf{e}_i) - f_i(t+\Delta t,\mathbf{x})}{\Delta x} = -\frac{1}{\tau}(f_i(t,\mathbf{x}) - f_i^{(0)}(t,\mathbf{x})).$$

In contrast to the transport, the terms in the right hand side (corresponding to the collisions) are evaluated explicitly at time t. We will analyse below the implications of this approach, concerning the original differential equation.
The crucial step to obtain the Lattice Boltzmann Equation (LBE) is to fix the grid spacing as $\Delta x = c\Delta t$. Then, two terms on the left side cancel out, and multiplication by Δt yields the final form

$$f_i(t+\Delta t,\mathbf{x}+\boldsymbol{\xi}_i\Delta t) = f_i(t,\mathbf{x}) - \frac{\Delta t}{\tau}(f_i(t,\mathbf{x}) - f_i^{(0)}(t,\mathbf{x})). \tag{3.5}$$

This scheme shows how the discretisation fits together with the chosen Cartesian lattice. The distributions move from a grid point positioned in $\mathbf{x}$ with particle speed c along the characteristic $\boldsymbol{\xi}_i$. After one time step, they are displaced by $\Delta t\boldsymbol{\xi}_i$ and arrive exactly at the next grid point situated in $\mathbf{x}+\Delta x\mathbf{e}_i$.
As mentioned above, due to explicit treatment of collisions, the difference scheme (3.5) is introducing some numerical viscosity. We want to accurately identify it and show how it can be

eliminated in consequence. To this purpose we look at the Taylor-series expansion of the term

$$\begin{aligned} f_i(t+\Delta t, \mathbf{x}+\boldsymbol{\xi}\Delta t) &= f_i + \frac{\partial f_i}{\partial t}\Delta t + \xi_{i\alpha}\frac{\partial f_i}{\partial x_\alpha}\Delta t + \\ &\quad \frac{1}{2}\frac{\partial^2 f_i}{\partial t^2}\Delta t^2 + \frac{1}{2}\xi_{i\alpha}\xi_{i\beta}\frac{\partial^2 f_i}{\partial x_\alpha \partial x_\beta}\Delta t^2 + \xi_{i\alpha}\frac{\partial^2 f_i}{\partial x_\alpha \partial t}\Delta t^2 + O(\Delta t^3) \end{aligned} \tag{3.6}$$

where all terms on the right side are evaluated in $(t, \mathbf{x})$. Substituting this expansion into the LBE (3.5) yields

$$\begin{aligned} \frac{\partial f_i}{\partial t} + \xi_{i\alpha}\frac{\partial f_i}{\partial x_\alpha} + \Delta t\Big(\frac{1}{2}\underbrace{\frac{\partial^2 f_i}{\partial t^2}} + \frac{1}{2}\xi_{i\alpha}\xi_{i\beta}\frac{\partial^2 f_i}{\partial x_\alpha \partial x_\beta} + \xi_{i\alpha}\underbrace{\frac{\partial^2 f_i}{\partial x_\alpha \partial t}}\Big) + O(\Delta t^2) \\ = -\frac{1}{\tau}(f_i - f_i^{(0)}). \end{aligned} \tag{3.7}$$

The time derivatives which are part of the error term $O(\Delta t)$ can be eliminated using the equation

$$\frac{\partial f_i}{\partial t} = -\xi_{i\alpha}\frac{\partial f_i}{\partial x_\alpha} - \frac{1}{\tau}(f_i - f_i^{(0)}).$$

The second derivatives in space and time are calculated from this term as

$$\begin{aligned} \frac{\partial^2 f_i}{\partial x_\alpha \partial t} &= \frac{\partial}{\partial x_\alpha}\Big(-\xi_{i\beta}\frac{\partial f_i}{\partial x_\beta} - \frac{1}{\tau}(f_i - f_i^{(0)})\Big) + O(\Delta t) \\ &= -\xi_{i\beta}\frac{\partial^2 f_i}{\partial x_\alpha \partial x_\beta} - \frac{1}{\tau}\frac{\partial}{\partial x_\alpha}(f_i - f_i^{(0)})\Big) + O(\Delta t) \\ \frac{\partial^2 f_i}{\partial t^2} &= \frac{\partial}{\partial t}\Big(-\xi_{i\alpha}\frac{\partial f_i}{\partial x_\alpha} - \frac{1}{\tau}(f_i - f_i^{(0)})\Big) + O(\Delta t) \\ &= -\xi_{i\alpha}\underbrace{\frac{\partial^2 f_i}{\partial t \partial x_\beta}} - \frac{1}{\tau}\frac{\partial}{\partial t}(f_i - f_i^{(0)})\Big) + O(\Delta t) \\ &= -\xi_{i\alpha}\xi_{i\beta}\frac{\partial^2 f_i}{\partial x_\alpha \partial x_\beta} + \frac{1}{\tau}\xi_{i\alpha}\frac{\partial}{\partial x_\alpha}(f_i - f_i^{(0)})\Big) - \frac{1}{\tau}\frac{\partial}{\partial t}(f_i - f_i^{(0)})\Big) + O(\Delta t) \end{aligned}$$

and are inserted into (3.7). All terms $\xi_{i\alpha}\xi_{i\beta}\frac{\partial^2 f_i}{\partial x_\alpha \partial x_\beta}$ cancel out, we obtain an additional collision term, and overall an equivalent differential equation g_i that is satisfied by the introduced difference scheme, denoting

$$g_i := \frac{\partial f_i}{\partial t} + \xi_{i\alpha}\frac{\partial f_i}{\partial x_\alpha} + \frac{1}{\tau}(f_i - f_i^{(0)}) - \frac{\Delta t}{2\tau}\Big(\frac{\partial}{\partial t} + \xi_{i\alpha}\frac{\partial}{\partial x_\alpha}\Big)(f_i - f_i^{(0)}) + O(\Delta t^2) = 0.$$

The additional term yields a modified, somewhat complicated relaxation time, but we can resolve it by building the continuity equations similar to Sec. 2.1:

$$\sum_i g_i = \frac{\partial \rho}{\partial t} + \frac{\partial \rho u_\alpha}{\partial x_\alpha} + O(\Delta t^2)$$

$$\sum_i \xi_{i,\alpha} g_i = \frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta}\Pi_{\alpha\beta} + \frac{\Delta t}{2\tau}\frac{\partial}{\partial x_\beta}(\Pi_{\alpha\beta} - \Pi_{\alpha\beta}^{(0)}) + O(\Delta t^2)$$

Inserting the Chapman Enskog expansion with small parameter ε (see Sec. 2.4), we obtain the following equation up to second order, i.e. up to $O(\Delta t^2)$ and $O(\varepsilon^2)$

$$\begin{aligned}\sum_i \xi_{i,\alpha} g_i &= \frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta}(\Pi^{(0)}_{\alpha\beta} + \varepsilon\Pi^{(1)}_{\alpha\beta}) - \frac{\Delta t}{2\tau}\frac{\partial}{\partial x_\beta}(\Pi^{(0)}_{\alpha\beta} + \varepsilon\Pi^{(1)}_{\alpha\beta} - \Pi^{(0)}_{\alpha\beta}) + O(\Delta t^2) + O(\varepsilon^2) \\ &= \frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial}{\partial x_\beta}(\Pi^{(0)}_{\alpha\beta} + \varepsilon(1 - \frac{\Delta t}{2\tau})\Pi^{(1)}_{\alpha\beta}) + O(\Delta t^2) + O(\varepsilon^2).\end{aligned}$$

Matching the terms with the Navier-Stokes equations yields for the viscous Stress tensor the relation

$$S_{\alpha\beta} = -\varepsilon(1 - \frac{\Delta t}{2\tau})\Pi^{(1)}_{\alpha\beta} + O(\Delta t^2) + O(\varepsilon^2).$$

In the Chapman-Enskog analysis we performed for the DVM the derivation resulted in

$$S_{\alpha\beta} = -\varepsilon(1 - \frac{\Delta t}{2\tau})\Pi^{(1)}_{\alpha\beta}$$

and evaluation of the first order tensor gave the relation $\nu = c_s^2 \rho_0 \tau$. Now, with the numerical viscosity identified as the term $\frac{\Delta t}{2\tau}$, it can be eliminated by using it as part of the physical model, resulting for the Lattice Boltzmann equation in the modified relation

$$\nu = -\varepsilon(1 - \frac{\Delta t}{2\tau}) = c_s^2 \rho_0 (\tau - \frac{\Delta t}{2}). \tag{3.8}$$

Obviously, to retain positive viscosity, the restriction $\Delta t < 2\tau$ has to be considered in practice. In total, it was possible to efficiently apply a discretisation which is basically first order accurate and, taking advantage of the Chapman Enskog method, to remove artificial viscosity. The obtained scheme is second order accurate and consistent with the Mach-error in the Boltzmann small velocity approximation of the Navier-Stokes equations. However, the relation (3.8) can be only used when solving the Lattice Boltzmann equation on a square lattice, with the explicit coupling of Δt and Δx. In case that time and space are treated independently by arbitrary numerical schemes, or even for a discretisation of the steady-state BE, the obtained order of convergence has to be reconsidered, especially in view of the accuracy against the NSE.

3.1.2. Lattice Boltzmann Method

Finally, we present the main advantages and also some disadvantages of the previosuly described special distretization. The LBE can be easily extended to obtain the characteristic Lattice Boltzmann method. In an operator-splitting approach of Marchuk-Yanenko type, the LBM divides equation (3.5) into two explicit sub-steps that can be performed with high computational efficiency. First, the f_i are distributed according to local collisions in the velocity-space:

$$f_i^\star(t+\Delta t,\mathbf{x}) = f_i(t,\mathbf{x}) - \frac{\Delta t}{\tau}(f_i(t,\mathbf{x}) - f_i^{eq}(t,\mathbf{x})) \tag{3.9}$$

Second, the distributions are transported to the next grid point along the characterstics by the simple shift

$$f_i(t+\Delta t,\mathbf{x}+\Delta t\boldsymbol{\xi}_i) = f_i^\star(t+\Delta t,\mathbf{x}). \tag{3.10}$$

Obviously, no solution of linear or nonlinear equations is necessary. The method can be naturally combined with parallelization and recently the use of modern hardware, for instance GPUs or supercomputers, is explored in scalable algorithms (see *www.skalb.de*).

Remark

We will present finally some practical implications introduced by the special discretisation of the LBE. Basically, it is restricted by $\Delta t < 2\tau$ (see [37]) and the numerical scheme suffers stability problems for values of τ close to the limit (see [28]).
Moreover, the LBE is due to construction consistent with the modeling error of order $O(\mathrm{Ma}^2)$, by fixing space-width and time-step on the configuration space through $c = \frac{\Delta x}{\Delta t}$. Consequently, we obtain directly a restriction on the time-step by

$$O(\mathrm{Ma}^2) = O((\frac{1}{c})^2) = O((\frac{\Delta t}{\Delta x})^2)$$

which means that Δt is a fixed simulation parameter in terms of Δx^2. Apparently, this is coherent for the translation of distributions to neighbouring nodes in the transport step. However, convergence against the incompressible Navier Stokes solution can only be achieved in the asymptotic limit of both $\Delta x \to 0$ and $\mathrm{Ma} \to 0$ (see [37]). During a grid refinement step the spacing is regularly divided with $\Delta x \to \frac{\Delta x}{2}$, at the same time the Mach number must be reduced. Increasing the sound speed by a factor of 2 results in an increased number of time-steps to drive the simulation (reach the simulation end, determined for example by a characteristic eddy turnover time) by a total factor of 4 with the equation

$$\frac{\Delta x}{2} = 2c \cdot \frac{\Delta t}{4}$$

ensuring that the transport to adjacent nodes remains consistent.This means the number of (micro) timesteps grows fast and can result in extreme computation times.

3.2. Advection explicit off-lattice Boltzmann schemes

To overcome the limitation to uniform grids, various approaches are possible, for example finite difference, finite element, finite volume or discontinuous Galerkin discretisations. A few can be found in literature, one of the first FD schemes appeared in [37], in [14] a FD method for curvilinear coordinates was given and recent examples are [41] (FV), [30] and [1] (FE) or [11] (DG), giving some 'modern' numerical approaches. The crucial point compared to the LBM is that the on-lattice approach enabled an explicit treatment for the implicit collision term. In one of the first off-lattice Boltzmann schemes, derived by Mei et al. in [32], this treatment was no longer valid. The authors used an extrapolation for the equilibrium distribution of the form $f_i^{eq,n+1} = 2f_i^{eq,n} - f_i^{eq,n-1}$ which is subject to severe stability restrictions. Bardow et al. (see [1]) described an alternative approach in the spirit of the previous section. After obtaining the form

$$\tilde{g}_i^{n+1} = \tilde{g}_i^n - \frac{\Delta t}{\lambda}(\tilde{g}_i^n - \tilde{g}_i^{eq,n}),$$

the authors applied the characteristic Galerkin discretisation of the advection equation by Zienkiewicz et al. [50] to their method.
Alternatively, Guo et al. were starting from the equation

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

where it should be noted that the transport term is evaluated at time t_n. For any choice of θ and again substituting a modified distribution function as in Eq. (3.3), they obtained the explicit form

$$g_i^{n+1} + \Delta t \boldsymbol{\xi}_i \cdot \nabla f_i^n = (1 - \frac{\Delta t}{\tau} + \frac{\Delta t}{\tau}\theta) f_i^n + \frac{\Delta t}{\tau}(1-\theta) f_i^{eq,n}.$$

It is obvious that a scheme without any (iterative) solution steps is obtained for the distribution function g_i, which can then in each successive time step be used to determine explicitly the needed f_i, resp., f_i^{eq}. As discretisation of the continuous transport term in above equation the authors proposed finite difference schemes, a central and an upwind scheme, both of second order accuracy. Additionally, a mixed scheme was proposed in case of strong numerical dissipation in flows at high Reynolds number. Finally, Guo et al. obtained

$$g_i^{n+1} + \Delta t \boldsymbol{\xi}_i \cdot \nabla_{mix} f_i^n = (1 - \frac{\Delta t}{\tau} + \frac{\Delta t}{\tau}\theta) f_i^n + \frac{\Delta t}{\tau}(1-\theta) f_i^{eq,n}$$

with ∇_{mix} denoting the mixed-difference scheme, controlled by a parameter ε:

$$\nabla_{mix} = \varepsilon \nabla_{upw} + (1-\varepsilon)\nabla_{central}$$

This shows that a treatment of the DVM in the spirit of Section 3.1 but without the restriction to structured grids is possible and was even accomplished in many cases. An important example is found in [11], where a discontinuous Galerkin discretisation was used and the order of the discretisation space was arbitrary. The gain, however, is only relative as the advection is treated purely explicitly, so again the CFL restriction holds. A remedy as mentioned in [1] would be treating the advection implicitly, but this sophisticated extension makes arise some interesting questions:

- Would it be possible to develop an intrinsically advection implicit method as was done for the collision operator and with this preserve the explicitness of the whole algorithm?
- Having an efficient, direct transport solver, what would be the advantage compared to the Lattice Boltzmann method, or to advection explicit off-lattice Boltzmann schemes?

During the thesis we will try to find some answers by looking closely at the possibilities to modify the equation, resp., at the numerical implications for the efficiency of the solution process.

3.3. Collision/advection implicit schemes

To perform a general time-discretisation of the discrete Boltzmann equation

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \nabla f_i = \Omega_i$$

we use a straightforward time-integration on $[t_n, t_{n+1}]$ quite similarly to Sec. 3.1 and obtain

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

The difference now is that we approximate both integrals using the θ scheme (compare to Li et al. [29]), instead of evaluating advection explicitly as in the previous section. The full θ-scheme reads

$$f_i^{n+1} - f_i^n + \Delta t[\theta \boldsymbol{\xi}_i \cdot \nabla f_i^{n+1} + (1-\theta)\boldsymbol{\xi}_i \cdot \nabla f_i^n] = \Delta t[\theta \Omega_i^{n+1} + (1-\theta)\Omega_i^n]. \tag{3.11}$$

Usually $\theta = 1/2$ is implemented, which corresponds to the Crank-Nicholson scheme and we will see that the resulting second order accuracy is especially important for time-dependent flows. Whatever space discretisation is used, the implicit treatment of the advection — while the only way to overcome the CFL restriction — always requires the solution of a linear (or nonlinear) system of equations. Li et al. used a least-squares finite-element (LSFE) space discretisation, but extrapolated the collision term in time $f_i^{eq,n+1} = 2f_i^{eq,n} - f_i^{eq,n-1}$ to avoid implicit treatment of nonlinearities. However, we refrained from this back-door solution, and faced the (fully) implicit coupling of advection and collision in our work. This approach can be even applied in a monolithic way to obtain directly steady-state solutions. A monolithic solver is on the one hand the 'simplest' possible method which avoids time-stepping, on the other hand it is numerically most demanding because the (nonlinear) coupling of the variables requires efficient solvers. We will present important implications in the next section.

3.3.1. Monolithic approach

In [21] we performed a similar time-discretisation as in Eq. (3.11), but wrote the scheme in the form

$$\alpha f_i + \boldsymbol{\xi}_i \cdot \nabla f_i = -\frac{1}{\tau}(f_i - f_i^{eq}) + g_i \quad , \quad i = 0, \ldots, N-1. \tag{3.12}$$

Different time-stepping schemes can be obtained by varying α with corresponding 'right hand side' g_i. However, for stationary problems at low Reynolds numbers it is just a coherent step to look at the limit $t \to \infty$. Consequently, in [21] the temporal argument t was dropped by setting $\alpha = 0$, $g_i = 0$ and we obtained a time-independent system of hyperbolic equations to be solved directly, i.e.

$$\boldsymbol{\xi}_i \cdot \nabla f_i + \frac{1}{\tau}(f_i - f_i^{eq}) = 0 \quad , \quad i = 0, \ldots, N-1. \tag{3.13}$$

One can say that this approach is contrary to the Lattice Boltzmann method. Instead of approaching the steady state solution of a given problem with (pseudo) time-stepping, we simply solve Eq. (3.13). However, in the new, monolithic approach we need to blend the transport and collision operators, as well as the boundary treatment, assuming that we find sufficiently powerful tools to deal with the nonlinearity and possibly very ill-conditioned linear problem.
Looking at Eq. (3.13) one can find it almost trivial being stripped from the temporal dimension: We have a (linear) differential operator together with a nonlinear coupling due to collisions, both 'simple' compared to the Navier Stokes equations composed of a nonlinear differential term and a second order Laplacian. However, we have nine local unknowns f_i in the standard D2Q9 lattice-set compared to pressure p and velocity (u_1, u_2) in the Navier Stokes model. Fortunately, the coupling of the nine distributions is quite sparse. What is more, though, is an additional dimension we have to consider, hidden at first sight, but attributable to the fact that using the Boltzmann equation to obtain Navier Stokes hydrodynamics, we use basically a compressible model to approximate incompressible flow. We identify this additional dimension with the Mach number $\mathrm{Ma} = \frac{U}{c_s}$ of the system, which is closely connected to the systems speed of sound $c_s = \frac{c}{\sqrt{3}}$. These terms in inverse relation we will discuss in detail.
In the derivation of Boltzmann schemes, a modeling error of order $O(\mathrm{Ma}^2)$ is taken into account starting with the BGK model of the collision term. Furthermore, the Chapman-Enskog expansion with small parameter $\varepsilon \sim \frac{1}{c}$ introduces several approximations in terms of order ε^2 throughout the analysis. In the Lattice Boltzmann method are introduced finite difference schemes with further assumptions which couple the discretisation of time and space. Grid length and timestep size are fixed by $\Delta x = c\Delta t$ (see [28]) through the 'unit of velocity' c, generally being set to one. It can be shown that the resulting discretisation errors are consistent with the overall second order compressibility error which is considered to determine the quadratic convergence of the LBM against the Navier-Stokes equations.
Unfortunately, using alternative discretisation schemes for the Boltzmann equation, it is not easy to leave this strict framework, mostly one has to decouple the treatment of space and time (and microscopic velocity) and make new assumptions. With our monolithic treatment of the DVM, c is no longer the unit of velocity $\frac{\Delta x}{\Delta t}$. Here, we have no more any timestep and, for that matter, no uniform grid spacing either, when we discretise the DVM on general meshes. Nevertheless, we found that the choice of c influences significantly the accuracy and computational efficiency of our model. Starting the work on the discrete Boltzmann equation it was not clear how to treat c, which appears throughout our discretisation. The idea to choose it as big as possible to reduce the Mach-error proved to be a wrong assumption because for a fixed mesh it resulted in errors proportional to c. A similar behaviour appeared in [6], wherein Dellar discussed the Mach number dependence of MRT models. Different authors dealing with an asymptotical analysis for the BE state that c must be chosen depending on h (see [23], [37]). Generally speaking, without the dependence of the two

variables we would not achieve a consistent approximation of the Navier-Stokes equations in the asymptotic limit. Quantitatively, the aim is to achieve an algorithm with quadratic convergence in $O(\mathrm{Ma}^2)$, because even the best discretisation cannot achieve a higher accuracy than given by the modeling error. But what exact relation gives optimal results?
In order to obtain an answer, we look at Eq. (3.13) and assume an arbitrary discretisation of order γ of the convective term

$$\begin{aligned}
\boldsymbol{\xi}_i \cdot \nabla f_i + \frac{1}{\tau}(f_i - f_i^{eq}) &= c(\boldsymbol{e}_i \cdot \nabla f_i) + \frac{1}{\tau}(f_i - f_i^{eq}) \\
&= c(\nabla_h f_i + O(h^\gamma)) + \frac{1}{\tau}(f_i - f_i^{eq}) \\
&= c\nabla_h f_i + O(c\, h^\gamma) + \frac{1}{\tau}(f_i - f_i^{eq}) \\
&= c\nabla_h f_i + O(\mathrm{Ma}^{-1} h^\gamma) + \frac{1}{\tau}(f_i - f_i^{eq}).
\end{aligned}$$

We obtain a mixed error term in the grid-spacing h and c in accordance with the observed numerical behaviour. However, in view of the fundamental compressibility error of order $O(\mathrm{Ma}^2)$, we attempt a simplified analysis. In Figure 3.1, we plot the asymptotic behaviour of the basic functions Ma^2 representing the modeling error and the discretisation-error on a fixed grid represented by Ma^{-1}. Obviously, when assuming the overall error being obtained by summation of both functions, we have two antipodal contributions in Ma. The second term has to be reduced by successive grid refinement and higher order space-discretisation techniques. In Figure 3.2 we plot the theoretical overall error as a function of $\mathrm{Ma}^2 + \mathrm{Ma}^{-1} h_i^\gamma$, with $h_i = 2^{-i}$ and different values $\gamma \in \{1,2,4\}$. Even in this coarse diagram it becomes obvious that (only) after the range falls below a certain compressibility limit, the resulting accuracy can be improved by choosing h very small and with high order γ. Finally, the asymptotic behaviour of the complete numerical approach can be optimized by choosing h dependent on the parameter c. From the stipulation

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

it follows that the discretisation error $O(\mathrm{Ma}^{-1} h^\gamma)$ will behave like $O(\mathrm{Ma}^2)$, therefore being consistent with the modeling error.
This assumption will have to undergo verification by various numerical tests. In Section 7.1 we present corresponding numerical results in good agreement with the theory for our monolithic approach with finite difference upwind discretisation of first and second order. But we think that similarly, arbitrary schemes are able to yield accurate results with small number of grid-points due to higher order space discretisations on unstructured, adapted grids. Quite early, a fourth order centered difference scheme was used for the convective term by Reider in [37] and, recently, Duester introduced in [11] a discontinuous Galerkin discretisation up to eigth order. However, we will present an important advantage of our FD upwind scheme, which yields lower triangular transport matrices due to a special numbering technique, and is supposed to improve considerably the efficiency of solving the resulting system of linear equations.

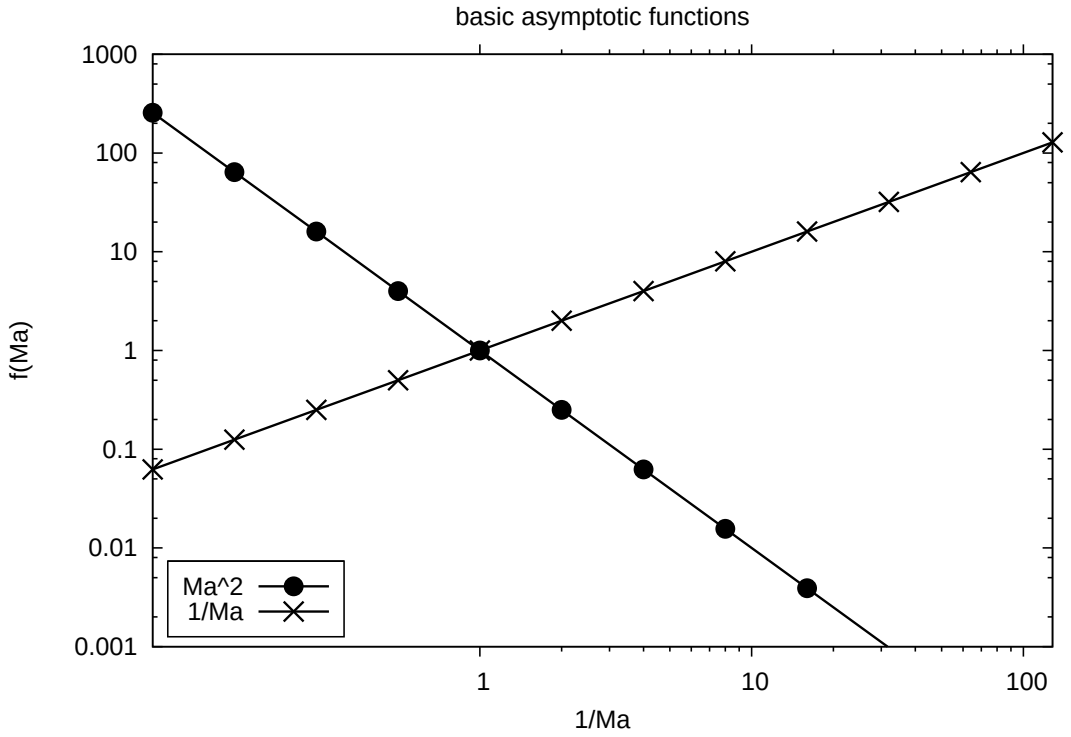


Figure 3.1: Logarithmic plot of Ma^2 (modeling error) against Ma^{-1}

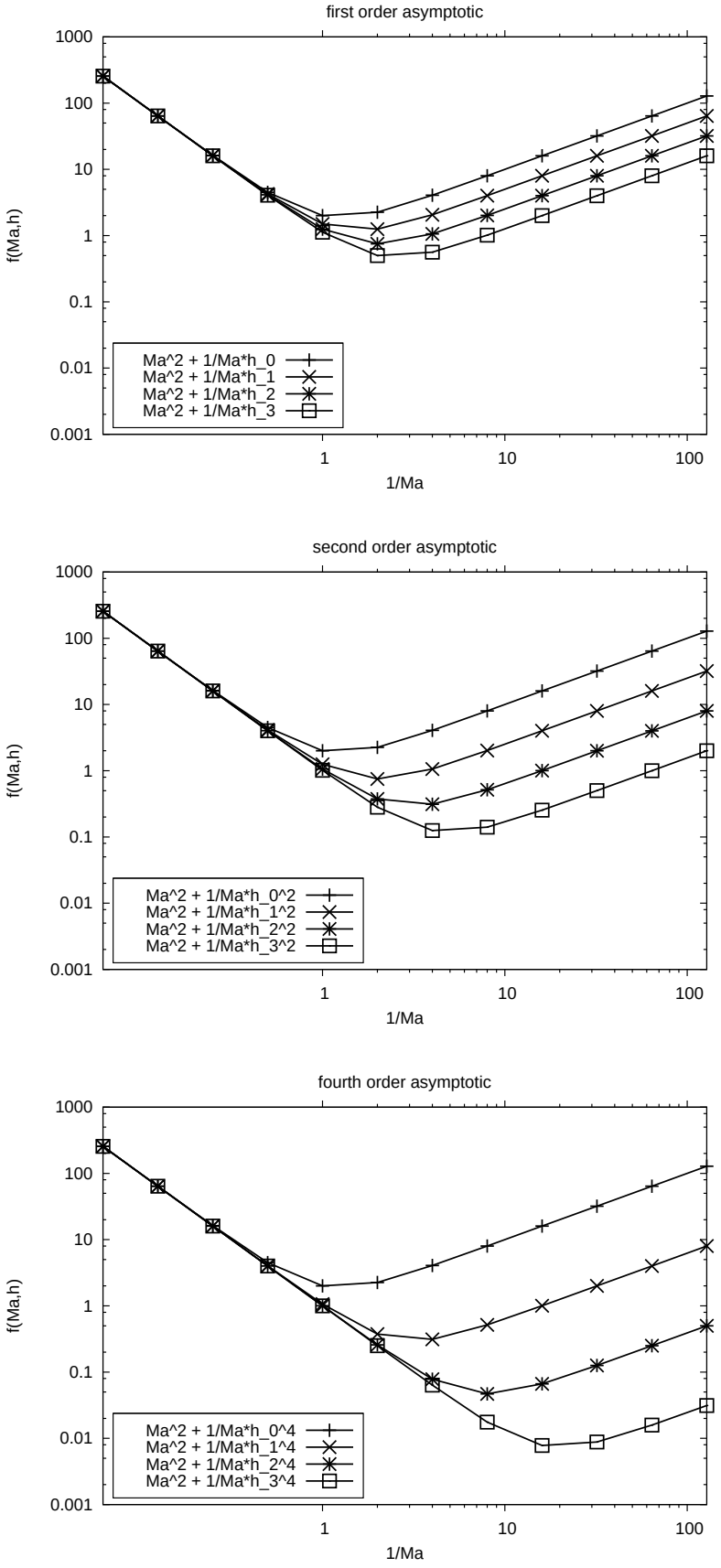


Figure 3.2: Asymptotic error dependence on Ma and order of h

3.4. Summary of Part I

The aim of Part I was to provide a basic understanding of the Boltzmann equation and to look beyond the popular but limited LBM. On the one hand it is important to understand that numerical schemes for the discrete Boltzmann equation yield — in the asymptotic limit of both, mesh size h and Mach number Ma — the same flow results as the incompressible Navier-Stokes equations. On the other hand, we showed that the explicit, two-step Lattice Boltzmann algorithm is just one basic, though computationally efficient discretisation.
To give a sufficiently complete round-up, we started by explaining the BGK model which introduces a single relaxation time approximation of the collision integral in phase space, compared to the advanced multiple-relaxation-time model. The discrete velocity model was derived by fixing a special set of lattice vectors, obtaining a PDE with constant characteristics. The Chapman-Enskog analysis on a discrete level was demonstrated for the incompressible 9-velocity BGK model, deriving the viscous stress tensor of the Navier-Stokes equations.
To account for the next discretisation steps, we tried to break down the DVM and analyse its parts regarding a numerical treatment, starting with the collision operator which is responsible for a local coupling of the microscopic variables. The implicitness which is necessary for stability is in general successfully avoided by a modified distribution function on structured grids, but also in off-lattice methods, if the advection is treated explicitly. Summing up the work of Bardow et al. [1] and Guo et al. [14] and based on [30], [32], [16] we categorized two classes of Boltzmann methods, the first one being advection explicit schemes which require usually no solution of any system of equations, instead, an explicit time-stepping is carried out with small timestep-size due to the CFL condition. Into the other main class fall methods with advection implicit treatment. They can overcome the CFL condition and obtain the solution of time-dependent flows with large time-steps using accurate, higher order time-stepping schemes. However, we focus on the specific case of steady-state problems in constructing a new, monolithic solver for the DVM on unstructured grids. In this approach we can no longer avoid implicit treatment of the collision operator which results in a coupled, nonlinear system of equations and requires efficient linear and nonlinear (iterative) solvers.

Part II

Efficient Numerical Methods for the Discrete Boltzmann Equation

4

Total discretisation of the DVM

The actual work on the topic of this thesis started by developing our own approach to the discretisation of the DVM. The motivation behind the following steps was to apply special techniques from Numerics for PDE in order to obtain an extended and more variable model based on the discrete Boltzmann equation, focusing on advection implicit schemes and thereby following the ideas from Section 3.3. The whole process will be covered by five sections in this chapter:
Firstly, in Section 4.1 we describe our time-discretisation wherein both collision *and* transport are included in the presented implicit schemes. Our second main aspect is high order spatial discretisation on unstructured grids which will be described in Section 4.2 by introducing constant characteristic upwind of first and second order. Additionally, we describe a special feature of the upwinding, which makes possible to solve directly pure convection problems by exploiting a special numbering of the unknowns. The next part shortly describes the boundary treatment which is a quite important and non-trivial part of the solution process. In Section 4.4 we combine all ingredients and write the resulting (monolithic) system of algebraic equations to be solved. Finally we give one main result of this thesis in Section 4.5 by introducing the *generalized equilibrium formulation*. Therein, we rewrite the discrete velocity model (1.2), resp., the obtained algebraic system, proposing a new Boltzmann-type equation that is directly connected to our special transport discretisation and is therefore advantageous in the solution process.

4.1. Implicit time-discretisation

We start the time-discretisation from the discrete velocity model

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \nabla f_i = -\frac{1}{\tau}(f_i - f_i^{eq}) \tag{4.1}$$

with a given set of discrete velocities $\boldsymbol{\xi}_i$ in phase space, most commonly the D2Q9 model is employed. To overcome typical stability restrictions, we want to treat the entire DVM implicitly to allow large time steps or even to solve directly for stationary flow in a monolithic approach, while a time-stepping scheme shall be used only for configurations with high Reynolds number and for fully nonstationary flow problems. For the latter, we denote $f_i^{n+1} = f_i(x,t)$, $f_i^n = f_i(x,t-\Delta t)$ and the collisions as $\Omega_i^n = -\frac{1}{\tau}(f_i^n - f_i^{n,eq})$ and introduce for the above equation a general time-discretisation, a θ-scheme similar to Eq. (3.11) and resorted into

$$f_i^{n+1} + \theta\Delta t\left(\boldsymbol{\xi}_i \cdot \nabla f_i^{n+1} - \Omega_i^{n+1}\right) = (1-\theta)\Delta t\left(\Omega_i^n - \boldsymbol{\xi}_i \cdot \nabla f_i^n\right) + f_i^n. \tag{4.2}$$

θ can be chosen arbitrary from the interval $[0,1]$, for example

- $\theta = 0$ corresponds to the explicit Euler, which we will omit in order to focus on implicit numerical schemes.
- $\theta = 1$ yields the implicit Euler discretisation of first order in time:

$$f_i^{n+1} + \Delta t\left(\boldsymbol{\xi}_i \cdot \nabla f_i^{n+1} - \Omega_i^{n+1}\right) = f_i^n \tag{4.3}$$

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

$$f_i^{n+1} + \frac{\Delta t}{2}\left(\boldsymbol{\xi}_i \cdot \nabla f_i^{n+1} - \Omega_i^{n+1}\right) = \frac{\Delta t}{2}\left(\Omega_i^n - \boldsymbol{\xi}_i \cdot \nabla f_i^n\right) + f_i^n \tag{4.4}$$

In order to treat transient flow problems a high (higher than first) order time-discretisation 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 (4.3) with the accuracy of (4.4) in the so-called fractional step θ scheme described in [47].
It is also possible to omit the time-dependence for the direct treatment of steady-state problems with equation

$$\boldsymbol{\xi}_i \cdot \nabla f_i(x) + \frac{1}{\tau}(f_i(x) - f_i^{eq}(x)) = 0 \tag{4.5}$$

which was mainly discussed in [21].
In this work, we concentrate on Eq. (4.5) when we treat steady state problems up to moderate Reynolds numbers, but we need to introduce solvers with sufficient numerical efficiency for this monolithic approach. On the other hand, we use the first order scheme (4.3) only as a model equation to compare convergence rates in (pseudo) time-stepping simulations, in case the stationary case is too ill-conditioned. For nonstationary flow problems, in our case a periodically oscillating flow, we use mainly the second order scheme (4.4) to obtain sufficient accuracy in time. We demonstrate its superiority over the simple implicit Euler for the *flow around cylinder* benchmark by comparing against given reference numbers for drag and lift.

Independent of the applied time-scheme the discretisation needs to be completed in space, which will be accomplished by a special finite difference technique in the next section. Unfortunately, it will turn out that with our proposed implicit treatment of the advection, which is a main part of this thesis, the implicitness of the collision term cannot be avoided like described in Section 3.2. The nonlinear, local coupling of distributions will be part of a whole algebraic system to be solved in each timestep, resp., in the monolithic approach.

4.2. The short-characteristic discretisation procedure

The transport operator for each constant characteristic in Eq. (1.2) can be described in the standard LB method as a trivial streaming process of particles to neighbouring nodes (see Section 3.1.2), due to the intertwining of space discretisation and mesh. However, in this Section we will describe our quite general finite difference discretisation of the differential term, since we allow unstructured meshes. This task has been performed very efficiently in [19], [20] 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 onedimensional differential equation. Actually, this procedure does not only apply for the set of lattice velocities, but for any arbitrary characteristic β (see Fig. 4.1). Therefore, and for reason of simplicity, we assume a hyperbolic equation with pure convection

$$\mathbf{n}_\beta \cdot \nabla u(x) = f(x), \tag{4.6}$$

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 discretisation procedure for the transport term in (4.6), of first and second order accuracy, respectively.

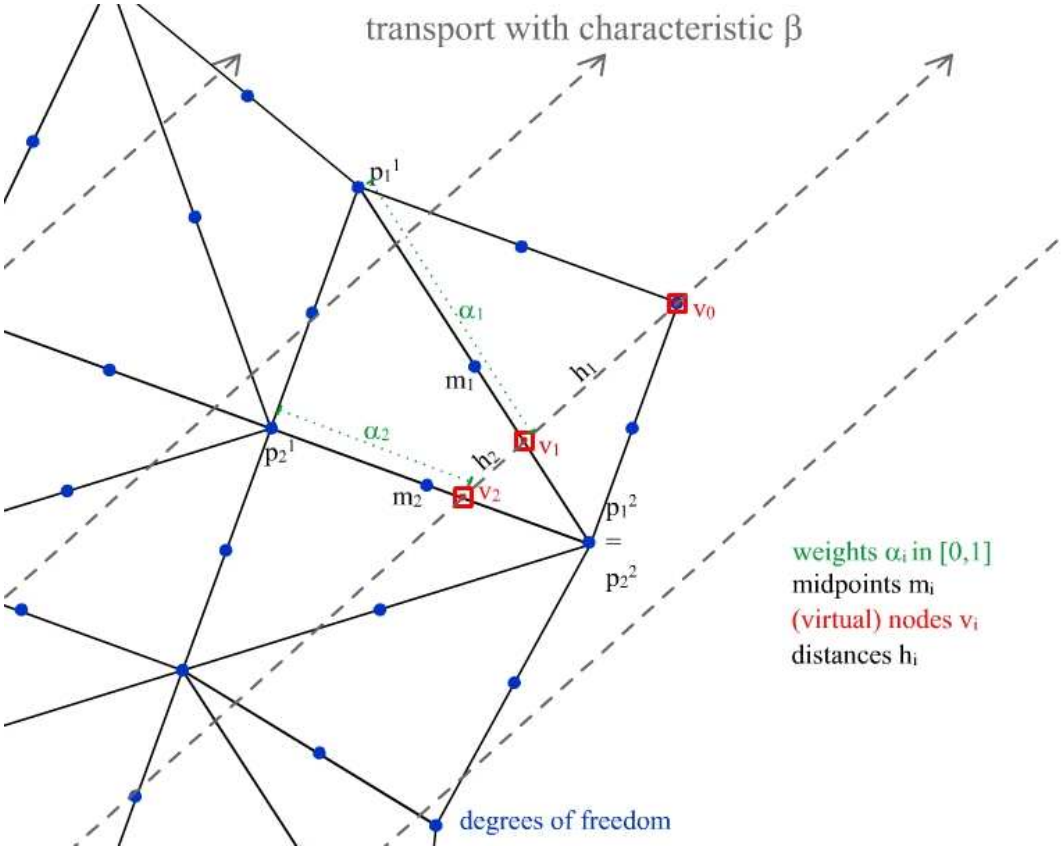


Figure 4.1: Constant characteristic upwind

4.2.1. First order upwind

Due to the constant characteristics in the Boltzmann equation as well as in the exemplary convection equation (4.6), we can view the problem as purely onedimensional as in Fig. 4.2. Consequently, we can write the spatial derivative as

$$\mathbf{n}_\beta \cdot \nabla u(v_0) = u'(v_0) = \frac{u(v_0) - u(v_1)}{h_1} + O(h_1), \tag{4.7}$$

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

$$\nabla_{upw1} u(v_0) := \frac{u(v_0) - u(v_1)}{h_1}. \tag{4.8}$$

This means we discretise in each grid-point using local and backward information, in Fig. 4.1 represented by nodes v_0 and v_1, respectively.

Figure 4.2: 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 discretisation scheme (4.8).

4.2.2. Second order upwind

It is obvious that in every point the difference quotient for the second order upwinding needs in total 3 function values, from the local and two backward nodes, to achieve the improved accuracy on arbitrary grids. The coefficients in the scheme can be found using the so-called 'polynomial fitting' technique as shown in [19]. It is also possible by a simple Taylor expansion in 1D; as previously we can assume a simplified onedimensional problem due to our constant characteristic approach. We give the Taylor series for $h := h_1$:

$$u(v_1) = u(v_0 - h) = u(v_0) - hu'(v_0) + \frac{h^2}{2}u''(v_0) + O(h^3)$$

However, we take into account a non-equidistant distribution of the grid-points, so we give for $r \cdot h := h_1 + h_2$ the Taylor series

$$u(v_2) = u(v_0 - rh) = u(v_0) - (rh)u'(v_0) + \frac{(rh)^2}{2}u''(v_0) + O(h^3).$$

Merging the two equations and cancelling the h^2 term yields

$$r^2u(v1) - u(v_2) = (r^2 - 1)u(v_0) + (rh - r^2h)u'(v_0) + O(h^3)$$

respectively

$$\frac{(1-r^2)u(v_0) + r^2u(v1) - u(v_2)}{h(r-r^2)} = u'(v_0) + O(h^2).$$

We summarize that we found the coefficients for the *second order upwind* scheme (to be denoted as upw2) in v_0:

$$\mathbf{n}_\beta \cdot \nabla u(v_0) \sim \nabla_{upw2}u(v_0) := \frac{-(1-r^2)u(v_0) - r^2u(v_1) + u(v_2)}{h_1(r^2 - r)}$$

The equidistant case ($h_1 = h_2 \Leftrightarrow r = 2$) results in the well known scheme

$$\nabla_{upw2}u(v_0) = \frac{3u(v_0) - 4u(v_1) + u(v_2)}{2h_1}. \tag{4.9}$$

Analogously, the differential operator in the DVM will be discretised for each of the 8 constant characteristics using either ∇_{upw1} or ∇_{upw2} and scaling by parameter c:

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

The scaling factor of $\sqrt{2}$ for the diagonal characteristics is due to the specific construction of the D2Q9 model with an underlying square lattice.
The requirement of two backward nodes for upw2 cannot be satisfied in a layer one node away from the inflow boundary. The second order could be retained by applying a central scheme for these points, but such treatment would contradict the overall upwind character which is necessary to obtain lower triangular matrices from the discretisation. Instead, the scheme simply switches to the first order difference quotient (4.8), giving results which are considered accurate enough, as showed various numerical tests (see [19]).

FEM-like interpolation

We extend the view again from 1D to our unstructured, twodimensional mesh and consider the aspect of interpolation in virtual nodes as in the case of first order upwinding. However, to sustain the total second order of the previously described scheme, it is not sufficient to use linear interpolation using two points per edge. Instead, we upgrade our mesh falling back on basic theory from Finite Element methods. This means that we need some additional degrees of freedom — to obtain second order accuracy on triangular grids we need 6 nodes per element — which we place in the edge midpoints. Function values in the virtual nodes, for example in v_1, are then interpolated from 3 nodes situated on the edge by in the following scheme (compare Fig. 4.1)

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

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:

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

Mixed upwind

The second order scheme can exhibit unphysical oscillations in areas of strong gradients, while the first order scheme lacks accuracy and tends to 'smearing' of the solution, as shown in [19]. A simple remedy presented therein was to use a linear combination of both schemes in the way of

$$\nabla_{mix} = \varepsilon \nabla_{upw2} + (1-\varepsilon)\nabla_{upw1} \quad , \quad \varepsilon \in [0,1]$$

obtaining a mixed order scheme which obviously has the same lower triangular matrix property as the two upwindings. In general, an appropriate local strategy for the linear parameter should be chosen, in order to achieve high accuracy by setting ε to 1 where it is allowed, while shifting $\varepsilon \to 0$ in case of steep gradients in the solution and thereby smoothing out possible over- and undershoots. In the results to be presented later, the obtained solutions were not so critical as to give immediate need for the mixed scheme. An extreme case given by the flow in the cavity, which is unsteady in the upper corners of the domain, was approximated accurately. In the future, we will analyse the discretisation ∇_{mix} in connection with the Lattice Boltzmann equation, as soon as there will arise a promising field of application.

4.2.3. Special sorting technique

Due to the upwind discretisation 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. (4.6). In fact, as described in [20], 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 discretisation 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 [7]) and can be written in pseudo-code (see Alg. 4.2.1) which was previously presented in [19].

Algorithm 4.2.1 Topological Sorting

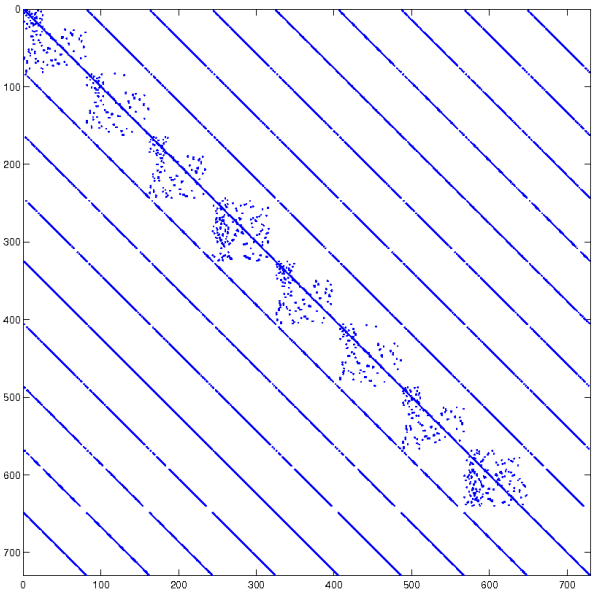
ORDER *(QUEUE[*], IN-NODES[*][*], OUT-NODES[*][*], NVT)*

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*

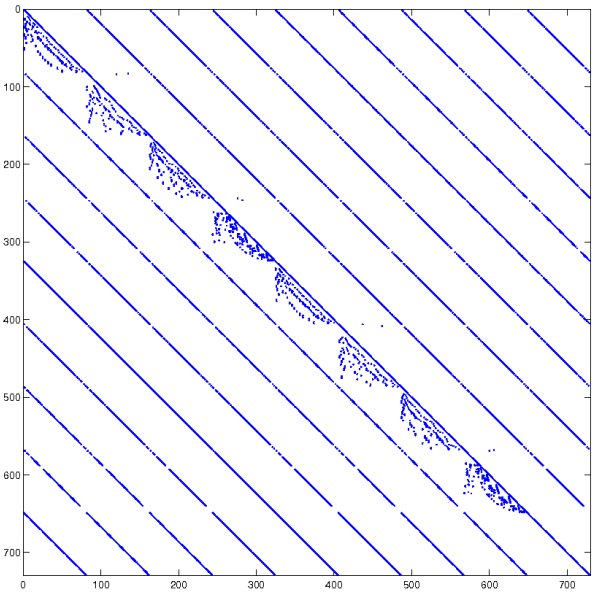
1. DO WHILE *k* $<$ *NVT*
a.) *v=QUEUE[k]*
b.) ***IF*** *v=0* ***THEN OUTPUT*** *'Graph is cyclic!',* ***STOP!***
c.) ***FOR EACH*** *j* ***IN*** *IN-NODES[v][*]* ***DO:***
d.) *OUTDEG[j]- -*
e.) ***IF*** *OUTDEG[j]=0* ***THEN*** *j* $\rightarrow$ *QUEUE*
f.) END FOR
g.) *k = k + 1*

The corresponding algorithm to sort a total number of n nodes is based on topological sorting used in the proof of the proposition in [7]. The graph's adjacency matrix as well as its transposed are stored using sparse matrix techniques. Another array is initialized which stores the number of row entries of the matrix during the algorithm (corresponding to the out-degree of every node). A queue is used to work off all nodes, at the same time it will contain the resulting numbering. In step 0, all inflow boundary nodes are written into the queue. They have out-degree 0 by definition, because the solution is given there and they are not using other nodes in any difference scheme. In step 1, nodes being in the queue are successively removed from the graph, and for the vanishing edges the algorithm decrements the out-degrees using the information from the transposed matrix. If an out-degree falls to zero, the node is written into the queue. The order in which the nodes have been written to the queue depicts the new numbering of the nodes. If the queue ended before all n nodes were extracted, this would mean that the graph contained a circle. So, the algorithm yields the desired numbering requiring $O(n)$ operations and the runtime to solve our transport problem using this numbering is linear, too.
Finally, we visualize the result of the resorting in Fig. 4.3: The standard allocation of the matrix entries is shown in Fig. 4.3a. After applying the permutation matrices (representing the new numbering) to the 8 transport parts situated on the block-diagonal of the system matrix, we see a change of the original allocation. Those entries, that belong to the finite difference discretisation of the differential operator are permutated into a lower triangular allocation in the matrix, see Fig.

4.3b. The off-diagonals contain the collision-term entries. Additionally, we can implement this discretisation in a matrix-free style, i.e. we can calculate all matrix entries (including the difference quotient and collision coefficients) 'on-the-fly'.



(a) Unsorted



(b) Sorted

Figure 4.3: Symbolic representation of effect of sorting algorithm

4.3. Boundary treatment

In CFD simulations there is a multitude of possible boundary conditions, in general defined by imposing velocity and pressure in the case of the Navier-Stokes equations. These macroscopic moments can easily be used to define Dirichlet-, Neumann- or Robin conditions. Other ways are to use a pressure drop that drives the flow, or periodic boundary conditions for example on the opposing walls of a channel. In the case of the Boltzmann equation, we have to use given macroscopic values to impose boundary conditions on the microscopic distributions. From the set of distributions in a boundary node (and in the case of a Dirichlet boundary) we impose values only on the distributions facing into the domain Ω, i.e. variables $f_i(\mathbf{x})$ on the sub-set

$$\Gamma_i^- := \{\mathbf{x} \in \partial\Omega \quad | \quad \mathbf{n}_\mathbf{x} \cdot \mathbf{e}_i < 0\}$$

with the outward normal unit vector $\mathbf{n}_x$.

The boundary configuration is in our case quite simple as we are using unstructured meshes. These are adapted to the geometry of the domain, therefore we can easily place all boundary nodes right on the wall. In contrast, for the Lattice Boltzmann method on uniform grids a boundary node can occur which is between a fluid node and an 'outside' node. A special treatment is therefore required to obtain higher order accurate schemes, approximations of the exact distributions lead to so-called Knudsen layers as studied in [5]. Bouzidi et al. apply therefore interpolation (see [3]) using four post-collision populations from the last two fluid nodes before the boundary along a characterictic. Third-order kinetic accuracy was obtained by Ginzbourg et al. in [13] by introducing multireflection boundary conditions and using six populations in a sophisticated way.
However, it should be distinguished between time-dependent and steady-state flow, higher order boundary schemes in time are not so important for the latter case. It is also not clear how schemes in the spirit of [3] and [13] would be implemented in our aspired monolithic solver and what would be their potential gain, especially compared to implicit boundary treatment. Therefore, we restrict ourselves in this thesis to local boundary schemes and use information from neighbouring nodes only for special cases. We will present in the following different local (bounce-back) methods, but first it is necessary to fix a numbering of the lattice vectors. 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.

Bounce-back scheme

In the case of a no-slip wall, boundary conditions $\mathbf{u}_{bc} = 0$ are enforced by 'reflecting' distributions $f_{-i}(\mathbf{x})$ which go out at the boundary back into the domain, it means we prescribe

$$f_i(\mathbf{x}) = f_{-i}(\mathbf{x}) \quad \text{on} \quad \Gamma_i^-. \tag{4.10}$$

It follows that opposing contributions in the sum $\sum_i \boldsymbol{\xi}_i f_i$ resulting in a zero momentum. This so-called bounce-back scheme causes, as in the case of collisions, a coupling of distributions with

varying characteristics.

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 [27]. Therein, the above bounce-back treatment is extended by adding components of the momentum of the wall, obtaining

$$f_i = f_{-i} + 2\rho_0 \cdot \omega_i \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}_{bc}}{c_s^2}. \tag{4.11}$$

The implementation of (4.11) is simple, as example we give the resulting equations for the D2Q9 model in case of a south wall aligned with the x-axis:

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

Zou-He scheme

Zou and He introduced in [51] a new bounce-back scheme which takes into account the nonequilibrium term of the distributions. 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.

Implementation of boundary conditions into the system-matrix

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

$$f_i - f_{-i} = 2\rho_0 \cdot \omega_i \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}_{bc}}{c_s^2}. \tag{4.12}$$

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. (4.2). Then the bounce back contribution of the outgoing f_{-i} is taken from the last time step, and appears in the right hand side:

$$f_i^{n+1} = f_{-i}^{n} + 2\rho_0 \cdot \omega_i \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}_{bc}}{c_s^2} \tag{4.13}$$

We treated the boundary implicitly throughout this thesis for two main reasons: One, in the case of the direct stationary solution an impicit treatment is necessary for the efficient solution of the

nonlinear equation (see Section 5.2), and, two, for nonstationary flow it improves stability when using large time stepping.

Singular points

In some situations, we have special boundary nodes that are not by definition in the set of $\Gamma^-_{\theta_i}$, that is we cannot apply the scheme described above, instead we have to introduce a special treatment. In Fig. 4.5a we have a pair of opposing distributions exactly in the corner enclosing the flow domain, which cannot be treated by the bounce-back equation (4.12), 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 [45] define some points at concave walls (secant running through solid) as incoming boundary values, for example the north and south distributions in Fig. 4.5b). 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

$$f_i^{eq}(\rho, 0) = \omega_i \rho. \tag{4.14}$$

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

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

We can derive the opposing distribution in an analoguous way, or just identify $f_{-i} := f_i$ according to Eq. (4.12) for the considered no-slip case where opposing contributions must cancel out.

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 (4.14) does not hold. Looking again at Figure 4.4, 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 unknows by extrapolation and to this purpose use our finite difference discretisation. We exploit upwind information of an adequate characteristic and use known values from inside of the domain in the constant extrapolation scheme

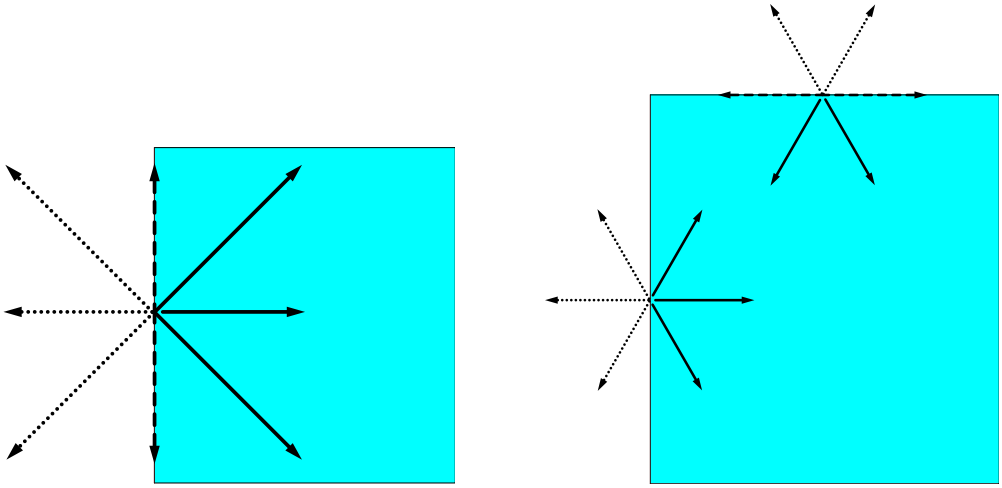
$$f_4(v_0) = f_4(v_1) = \lambda_1 f_4(p_1^1) + \lambda_m f_4(m_1) + \lambda_2 f_4(p_1^2), \tag{4.15}$$

or the linear extrapolation

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

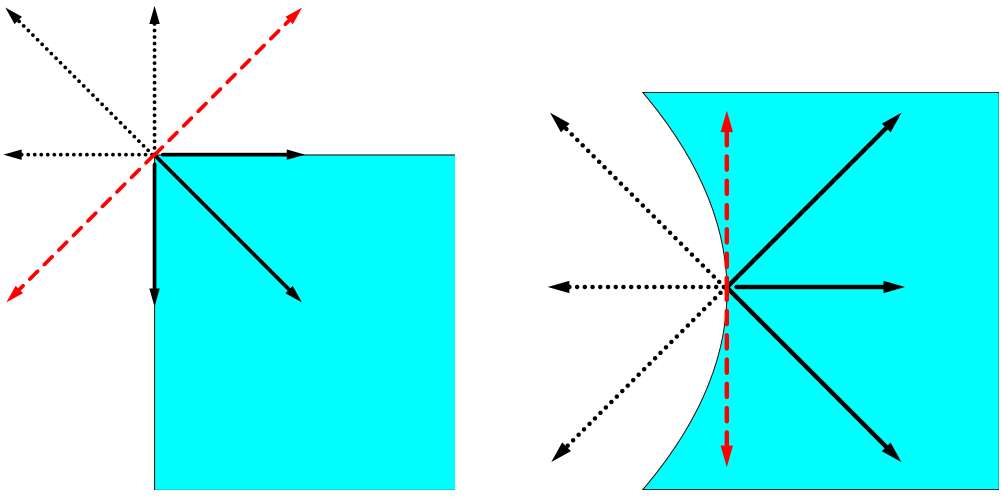
Neumann boundary

The final configuration to be discussed is the Neumann boundary condition (natural BC in NSE) as used for example for the outflow of the channel in the *flow around cylinder* benchmark ([39]). To satisfy the natural condition $\frac{\partial \mathbf{u}}{\partial n} = 0$, we use the constant extrapolation (4.15), 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 LBM, where the values are 'copied' from the last layer before the boundary in the structured mesh.



(a) West boundary in D2Q9 model (b) Boundary treatment in D2Q7 model

Figure 4.4: Standard boundary configurations



(a) Treatment in corners (b) Concave case

Figure 4.5: Special boundary configurations

4.4. Algebraic system resulting from basic discretisation

Having introduced a number of semi-discrete schemes in time and the special space discretisation of upwind type, we can finally write the equations derived from the DVM as a fully discretised system. For this purpose we denote

$$\mathbf{T}_i f_i \sim \boldsymbol{\xi}_i \cdot \nabla f_i + \frac{1}{\tau} f_i$$

with the operator T_i according to the first, resp., second order upwind discretisation of the transport term. Then we obtain the following discrete, nonlinear algebraic systems with $i = 0, \ldots, 8$:

θ-scheme:

$$f_i^{n+1} + \theta \Delta t \left(\mathbf{T}_i f_i^{n+1} - \frac{1}{\tau} f_i^{n+1,eq} \right) = (1-\theta)\Delta t \left(\frac{1}{\tau} f_i^{n,eq} - \mathbf{T}_i f_i^n \right) + f_i^n \tag{4.16}$$

Implicit Euler system ($\theta = 1$):

$$f_i^{n+1} + \Delta t \left(\mathbf{T}_i f_i^{n+1} - \frac{1}{\tau} f_i^{n+1,eq} \right) = f_i^n$$

Crank-Nicholson system ($\theta = \frac{1}{2}$):

$$f_i^{n+1} + \frac{\Delta t}{2} \left(\mathbf{T}_i f_i^{n+1} - \frac{1}{\tau} f_i^{n+1,eq} \right) = \frac{\Delta t}{2} \left(\frac{1}{\tau} f_i^{n,eq} - \mathbf{T}_i f_i^n \right) + f_i^n$$

Monolithic system for stationary problems:

$$\mathbf{T}_i f_i - \frac{1}{\tau} f_i^{eq} = 0 \tag{4.17}$$

Finally, in view of a reformulation of the obtained systems and also in preparation of nonlinear solution methods in Chapter 5, we introduce a linearization of the algebraic system.

Linearization of the collisions

All 4 algebraic systems stated above are, due to implicit treatment of the collisions, nonlinear 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

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

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). 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 (2.12) 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

$$\begin{aligned} f_i^{eq} &= \omega_i \left(\rho + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})(\boldsymbol{\xi}_i \cdot \tilde{\mathbf{u}})}{2c_s^4} - \frac{(u_1\tilde{u}_1 + u_2\tilde{u}_2)}{2c_s^2} \right) \\ &= \omega_i \Big(\sum_k f_k + 3\sum_k D_{ik} f_k \\ &\quad + \frac{9}{2} \sum_k D_{ik} f_k \sum_k D_{ik} \tilde{f}_k \\ &\quad - \frac{3}{2} (\sum_k D_{1k} f_k \sum_k D_{1k} \tilde{f}_k + \sum_k D_{2k} f_k \sum_k D_{2k} \tilde{f}_k) \Big) \\ &=: \sum_k \tilde{\omega}_{ik} f_k. \end{aligned} \tag{4.18}$$

Thus resolving above equation down to the distribution level 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.

Compressible vs. incompressible model:

The equilibrium term in the compressible model, i.e.

$$f_i^{eq} = \omega_i \rho \left(1 + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{(u_1^2 + u_2^2)}{2c_s^2} \right)$$

in the macroscopic variables density ρ and momentum ρu would introduce a different nonlinearity due to f_i appearing as rational polynomials which makes use of a Newton solver especially difficult. After some initial tests, improved accuracy was not to be expected using this model, so we refrained to the equilibrium term of the incompressible model which is only a quadratic polynomial in f_i.

Complete linear system and regularity:

We obtain the full system according to the linearization of our stationary monolithic approach by:

$$\left(\begin{bmatrix} \mathbf{T}_0 & & & & \\ & \mathbf{T}_1 & & & \\ & & \mathbf{T}_2 & & \\ & & & \ddots & \\ & & & & \mathbf{T}_8 \end{bmatrix} - \frac{1}{\tau} \begin{bmatrix} \tilde{\omega}_{00} & \tilde{\omega}_{01} & \tilde{\omega}_{02} & \cdots & \tilde{\omega}_{08} \\ \tilde{\omega}_{10} & \tilde{\omega}_{11} & \tilde{\omega}_{21} & & \\ \tilde{\omega}_{20} & \tilde{\omega}_{21} & \tilde{\omega}_{22} & & \vdots \\ \vdots & & & \ddots & \\ \tilde{\omega}_{80} & & \cdots & & \tilde{\omega}_{88} \end{bmatrix} \right) \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_8 \end{bmatrix} = 0$$

The coefficients of above system can be computed on-the-fly, but it is also possible to save the global, sparse matrix (with implicit boundary conditions) and obtain the solution using a direct linear solver (see Sec. 6.1). However, a careful treatment is required in this case. This is due to the density which is uniquely defined only up to a constant (similarly to the pressure in the Navier-Stokes equations). Consequently, the original system is singular and the direct solver (like UMFPACK) might not give any solution. In some CFD simulations, one can normalize the average pressure after each matrix-vector multiplication in an iterative scheme. In the Boltzmann context,

we have no actual simulation-variable for the pressure, only the density ρ (which is the scaled pressure $p = c_s^2\rho$) given by the sum of the local distributions. Consequently, as a regularisation-step of the system, we chose *one* outgoing distribution, say a boundary variable f_i^*, and fix it by setting $f_i^* = \omega_i$. Using a direct solver, it means introducing an identity row in the assembled matrix.
Variations in the solution are then usually given by $f_i = \omega_i\rho_0 + O(\mathrm{Ma})$ which can result in roundoff errors when computing the difference of f_i and f_i^{eq} which is of the same order (see [6], [40]). Skordos therefore rearranged the simulation variables to $f_i - \omega_i\rho_0$ by substracting out the equal term. We can obtain similar positive effect by setting $f_i^* = 0$. This yields solutions of order $O(\mathrm{Ma})$, with negative distributions appearing what is actually no unphysical behaviour. However, in the case with $f_i^* = \omega_i$ no errors could be observed using 16 digit arithmetics. In practive, the solution in the vicinity of the fixed point looks slightly distorted especially for small sound parameter c. Reducing the Mach number makes this distortions disappear, but it would be interesting to analyse this behaviour further.

4.5. Generalized equilibrium formulation (GEF)

In [20],[46] the concept of the *generalized mean intensity* (GMI) had been successfully combined with a direct transport solver to treat radiative transfer problems. The idea is to convey this technique from that integro-differential equation to the LBE: In this case we cannot improve the storage cost like in the GMI approach due to the coefficients $\tilde{\omega}_{ik}$ in Eq. (4.18) varying with each characterisitic i. Nevertheless, the GMI was also able to combine the advantages of an efficient transport discretisation on the one hand with special preconditioning to deal with stiff problems due to large scattering on the other hand. When we draw the connection between radiative transfer and the Boltzmann equation, we can assume that dominant collision in the latter would, just as in the case of dominant scattering, require special preconditioning. The spectral analysis of the system matrix in Section 6.2 confirmed that the sole use of an efficient transport solver can be insufficient. Therefore, we introduce an algebraic reformulation of the Boltzmann equation, which is completely new and was only recently published by the author in [21]. The procedure we carry out in the following is analoguous for the time-stepping θ-schemes (4.16). However, here we refrain to the monolithic steady scheme and start our reformulation with the discrete algebraic equation (4.17). Equation

$$\mathbf{T}_k f_k = \frac{1}{\tau} f_k^{eq} \quad k = 0, \ldots, 8 \tag{4.19}$$

can be formally rewritten as

$$f_k = \mathbf{T}_k^{-1} \frac{1}{\tau} f_k^{eq} \quad k = 0, \ldots, 8 \tag{4.20}$$

by applying the inverse transport operator (discrete transport-matrices). Next, we multiply with the corresponding weights $\tilde{\omega}_{ik}$ from equation (4.18) for each $i = 0, \ldots, 8$ and continue with the generalized form

$$\tilde{\omega}_{ik} f_k = \tilde{\omega}_{ik} \mathbf{T}_k^{-1} \frac{1}{\tau} f_k^{eq} \quad k = 0, \ldots, 8.$$

Summing up over k finally gives us for each equilibrium term f_i^{eq}:

$$f_i^{eq} = \sum_k \tilde{\omega}_{ik} f_k = \sum_k \tilde{\omega}_{ik} \mathbf{T}_k^{-1} \frac{1}{\tau} f_k^{eq} \quad i = 0, \ldots, 8$$

respectively

$$f_i^{eq} - \sum_k \tilde{\omega}_{ik} \mathbf{T}_k^{-1} \frac{1}{\tau} f_k^{eq} = 0 \quad i = 0, \ldots, 8. \tag{4.21}$$

The f_k^{eq} in equation (4.21) are linearized exactly as previously, resulting in the *generalized equilibrium formulation* (GEF) written in the new matrix form:

$$\left(\mathbf{I} - \begin{bmatrix} \tilde{\omega}_{00} \frac{1}{\tau} \mathbf{T}_0^{-1} & \tilde{\omega}_{01} \frac{1}{\tau} \mathbf{T}_1^{-1} & \cdots & \tilde{\omega}_{08} \frac{1}{\tau} \mathbf{T}_8^{-1} \\ \tilde{\omega}_{10} \frac{1}{\tau} \mathbf{T}_0^{-1} & \tilde{\omega}_{11} \frac{1}{\tau} \mathbf{T}_1^{-1} & & \vdots \\ \vdots & & \ddots & \vdots \\ \tilde{\omega}_{80} \frac{1}{\tau} \mathbf{T}_0^{-1} & \cdots & \cdots & \tilde{\omega}_{88} \frac{1}{\tau} \mathbf{T}_8^{-1} \end{bmatrix} \right) \begin{bmatrix} f_0^{eq} \\ f_1^{eq} \\ \vdots \\ f_8^{eq} \end{bmatrix} = 0 \tag{4.22}$$

This means that in order to solve equation (4.19), we first solve for the terms f_i^{eq} and afterwards we obtain the f_i from a simple post-processing step according to equation (4.20). It is even possible to compute the moments of density and velocity directly from the equilibrium values, according to the chapter treating the Chapman-Enskog expansion and Equations (2.15) and (2.15) therein. To evaluate higher order moments like stress, though, the distributions f_i are necessary (see Sec. A2.1).

In practice, the resulting system matrix is not given explicitly and cannot be used in direct linear solvers as the basic discretisation for instance. Here, the system contains the transport steps in an inverse manner, which means that we obtain an *implicit* matrix only as we do not calculate the actual inverses T_i^{-1}. What is done, though, is applying the inverse to a vector which is implemented as the solution of a linear system for a given right hand side. Due to our special discretisation and the assumed lower triangular form of the transport matrices T_i, this step is very cost efficient. The GEF even allows additional preconditioning for example if we succeed in obtaining information on the actual matrix-entries. The implicit matrix in (4.22) consists of an identity and the (weighted) inverse transport blocks. We can directly obtain at least the diagonal entries of each $\mathbf{T}_i^{-1}$, linear algebra states that they are simply given by the inverse of the diagonal entries of $\mathbf{T}_i$. It follows that all diagonals of the composed system are known explicitly, scaled by weights $\tilde{\omega}_{ik}$ and the relaxation time $\frac{1}{\tau}$, based on this part we will present a special preconditioner for collision dominated configuration in Section 6.4.

How is a matrix-vector multiplication of the above system carried out in practice? Although the matrix looks very dense with 81 inverse transport steps, the implementation of applying it to a vector looks slightly different in our code . First, we apply transport solution steps for the 9 characteristics towards the vector f^{eq} and, second, the resulting vector is multiplied by weights ω_{ik} which corresponds to building the sum as in Eq. (4.21). So the implicit system matrix is implemented in the efficient manner

$$\left(\mathbf{I} - \frac{1}{\tau} \begin{bmatrix} \tilde{\omega}_{00} & \tilde{\omega}_{01} & \cdots & \tilde{\omega}_{08} \\ \tilde{\omega}_{10} & \tilde{\omega}_{11} & & \vdots \\ \vdots & & \ddots & \vdots \\ \tilde{\omega}_{80} & \cdots & \cdots & \tilde{\omega}_{88} \end{bmatrix} \begin{bmatrix} \mathbf{T}_0^{-1} & & & 0 \\ & \mathbf{T}_1^{-1} & & \\ & & \ddots & \\ 0 & & & \mathbf{T}_8^{-1} \end{bmatrix} \right) \begin{bmatrix} f_0^{eq} \\ f_1^{eq} \\ \vdots \\ f_8^{eq} \end{bmatrix} = 0.$$

4.6. Summary and outlook

In this chapter we showed how to apply modern numerics for PDEs to obtain a discretisation of the DVM in the spirit of collision/advection implicit schemes described in Section 3.3. By introducing implicit Euler and Crank-Nicholson schemes we showed that arbitrary time-discretisations are valid for the time-dependent Boltzmann equation. What is more, we introduced a monolithic, fully coupled nonlinear system for steady-state problems, for this purpose applying implicit boundary conditions. The presented finite difference space discretisation of 1st and 2nd order is just one possible way to treat the advection, applying FEM-like interpolation for the second order scheme on arbitrary triangular grids. However, a high numerical efficiency is obtained due to a special sorting technique that yields lower triangular matrices for the transport steps. This technique was used in a pioneering work for an algebraic transformation of the given discrete system. The developed *generalized equilibrium formulation* incorporates the inverse transport steps into the equations. This results in a system matrix which is known only implicitly but can be efficiently applied to a given vector.

In the following, we can exploit the obtained numbering also directly in a transport preconditioner in view of iterative (Krylov-space) linear solvers. This combination is supposed to give for convection dominated configurations excellent, level-independent convergence rates which we expect also in solving the plain GEF system. The GEF approach, however, has potential to deal with stiff configurations and bad condition numbers by using additional preconditioning. In particular, we will show that the GEF can be efficiently embedded in a multigrid framework, both as smoother and coarse grid solver. We present details about nonlinear and linear strategies for

the basic discretisation as well as the advanced *generalized equilibrium formulation* in the next chapters and a proof of concept for our numerical methods is given in Part III of this thesis.

5

Nonlinear solvers

In the previous chapter we derived a system of algebraic equations which is obviously nonlinear (quadratic) due to the terms $(\boldsymbol{\xi}_i \cdot \mathbf{u})^2$ and $\mathbf{u}^2$ in the collision operator of the SRTBGK model. Using the linearization (4.18) it would be possible to update $\tilde{f} = f^n$ only once every step or apply extrapolation $\tilde{f} = 2f^n - f^{n-1}$ in a (pseudo) time-stepping with very small Δt. However, the stability and accuracy of such a scheme would be poor, a fully stationary approach impossible. In view of modern numerics we aim to obtain full control of the nonlinear defect, either by fixed-point iteration or the Newton method. In the following, we will see that both schemes applied to the Boltzmann equation are similar in view of computational cost and implementation, but differ extremely in efficiency. Therefore, our description of the nonlinear problem can be discharged shortly, in contrast to the section about linear problems and solution tools.

5.1. Damped fixed point iteration

We denote the system we derived in Section 4 as

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

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

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \omega N(\mathbf{x}^n)^{-1}(N(\mathbf{x}^n)\mathbf{x}^n - g) \quad , \quad \omega > 0$$

with optional damping by a factor ω. The results in Sec. 9.1 will show that the convergence of this basic scheme is too poor to fully reduce the nonlinear defect of the stationary problem (4.17). 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.

5.2. Newton method

As alternative to the fixed-point iteration we present Newton's scheme and discuss some implications in using this method. To that purpose we write system (5.1) in residual form

$$R(\mathbf{x}) = N(\mathbf{x})\mathbf{x} - g = 0. \tag{5.2}$$

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

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

The application of the full Newton method in the Navier-Stokes equations proves difficult when dealing with terms of the form $\mathbf{u}\nabla\mathbf{u}$ which are not given analytically in the discretised equations. Mostly one reverts to an approximation of the Jacobian by a difference quotient in the so-called Newton-Raphson scheme. Fortunately, in our case we have given a fully analytical representation in our discrete equations, making it easy to obtain a full Jacobian. In the algebraic systems (4.16) — (4.17) we have mostly linear operators due to transport and mass terms. An exception is the equilibrium term which in the incompressible model is a quadratic polynomial in the macroscopic velocity and density. Written in the primary variables it is quadratic as well, therefore the derivation of f_i^{eq} is easy. In short, summing up the partial derivatives $\frac{\partial f_i^{eq}}{\partial f_k}$ for all k and linearizing similarly to previous Section 4.4 yields:

$$\begin{aligned} df_i^{eq} &= \omega_i\Big(\sum_k f_k + 3\sum_k D_{ik}f_k + \frac{9}{c^2}\sum_k D_{ik}f_k(\boldsymbol{\xi}_i\cdot\tilde{\mathbf{u}}) - \frac{3}{c}\Big(\sum_k D_{1k}f_k\tilde{u}_1 + \sum_k D_{2k}f_k\tilde{u}_2\Big)\Big) \\ &=: \sum_k \bar{\omega}_{ik}f_k \end{aligned} \tag{5.3}$$

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. (4.18). 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

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

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

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

The next section will widely discuss linear solution methods for these systems, we will distinguish the two variants especially in view of applied preconditioners.

6

Solution of the linear system

The outer nonlinear solver iterates on a system we denoted as $N(\mathbf{x})\mathbf{x} = \mathbf{b}$ with $\mathbf{x}$ representing the discrete solution vector of all f_i. After linearization of $N(\mathbf{x})$ (resp., the equilibrium term) we obtain a system matrix A corresponding to the time and space discretisation previously described in Section 4. In this section we will focus on the linear solution tools that we actually used throughout the thesis.
Beforehand, let us look at the obtained structure of the linear system to have a basic understanding of the problem. The main (block) diagonal of the system matrix is determined by two basic numbering techniques. The 'standard' case, listing for every direction the spatial nodes just as they were identified by the grid generator, produces nine so-called 'transport blocks' of size $n \times n$ (see Fig. 6.1a). The solution vector is then given by

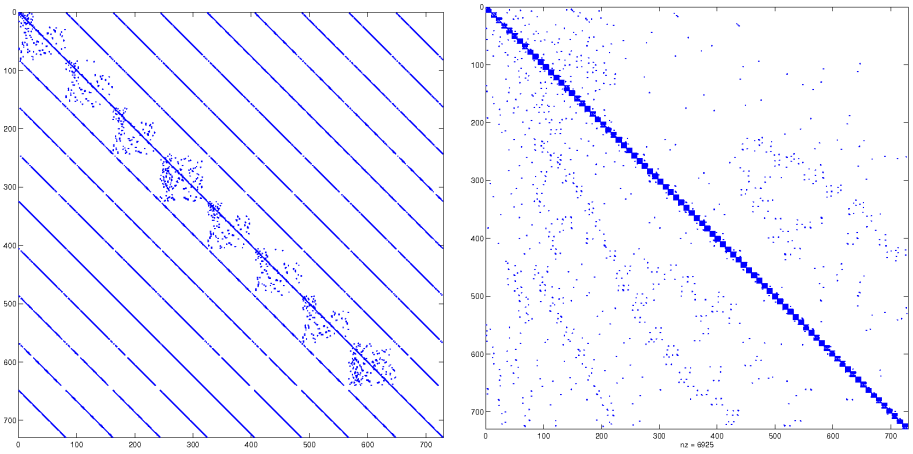
$$\underbrace{f_1(x_1), \ldots, f_1(x_n)}_{\mathbf{f_1}}, \underbrace{\ldots}_{\mathbf{f_i}}, \underbrace{f_9(x_1), \ldots, f_9(x_n)}_{\mathbf{f_9}}.$$

Instead of the directional variable, we can use the local unknowns as the first index, obtaining a narrower block-diagonal of 9×9 blocks (see Fig. 6.1b), with the solution vector enumerated as

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

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 4.2.3, see Fig. 4.3) 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. In every row we have 9 (diagonal and off-diagonal) matrix-entries corresponding to the discrete velocities, it means a 9 by 9 collision block for the distributions in each 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, *neighbouring* nodes are mainly coupled through the transport term, in this case we talk about distributions with the same microscopic velocity. Altogether, the resulting matrix allocation is quite sparse.

There are different ways how to deal with the given system: Direct solvers like LAPACK or UMFPACK are efficient for smaller problems, but scale badly with increasing number of unknowns (see Sec. 6.1), so that iterative methods have to be used for many gridpoints. The Richardson iteration is just a basic possible solver. This well-known, but usually very slow defect correction method, especially in the case of an ill-conditioned matrix A (see Sec. 6.2), can be accelerated by preconditioning techniques. The same holds for Krylov-space methods like BiCG-Stab [48] and GMRES [38] presented in Sec. 6.5. These schemes are especially efficient in case of strong clustering of



(a) Indexing due to directions (b) Local index first

Figure 6.1: Matrix allocation depending on primary index (and sorting)

eigenvalues, we present two different preconditioners also in view of this advantageous feature in Sec. 6.4. Finally, we describe the multigrid algorithm as an advancement of our iterative solvers with special preconditioning.

6.1. Direct linear solvers

Wherever any systems of linear equations occur in science or engineering, the first 'natural' impulse is to use direct solution methods, be it using Gaussian elimination for small 'pen and paper' systems, or implement the scheme with few lines of code. With modern computers, the cubic complexity of the algorithm is no hindrance, at least at the beginning. With growing number of unknowns, as the next logical step engineers avail themselves of powerful linear algebra libraries. Using LAPACK or UMFPACK in a black-box manner they can comfortably obtain (sooner or later) the exact solution, until they hit the memory wall with RAM and runtime demands of $O(n^2)$, resp., $O(n^3)$ in the worst case, even sophisticated libraries have superlinear complexities.
Also for the author of this thesis, a direct linear solver became a very useful tool at the beginning of his work, when the implementation of efficient iterative solvers was still some way ahead. UMFPACK yielded important initial results, i.e., wrong boundary conditions became obvious from the plotted pictures, well-posedness and regularity of the system obtained by our discretisation were indicated by the software. Additionally, we had to deal with the 'new' parameter of the sound-speed, which on the one hand significantly influences the conditioning of the system matrix (see Sec. 6.2) and on the other hand shows a specific (asymptotic) behaviour concerning the overall accuracy (see Sec. 7.1). Analysing the latter, while not being overcome by extreme ill-conditioning, was feasible using an efficient direct solver. In Table 6.1 we show the memory and cpu-time required by UMFPACK to solve a linear system, in this case resulting from a discretisation of the *driven cavity* model on a structured mesh (see Appendix A1.1). The superlinear scaling of the runtime is obvious and the memory of a 4GB machine is only sufficent to solve problems on a grid with 66049 points, even less for the 2nd order upwind which uses a more complex difference quotient. However, the runtime was independent of arbitrary choices for the sound-parameter c and we obtained comparable results.
Having overcome the first obstacles, we could start designing efficient linear iterative solvers — knowing at last the appropriate range of configurations — which will be presented in the following sections.

	upw1			upw2		
#dof	entries	memory	runtime	entries	memory	runtime
81	6920	37928	1,60E-2	8502	41544	2,01E-2
289	26232	135016	5,20E-2	32918	147368	1,44E-1
1089	102104	508904	3,80E-1	129558	554280	2,00E+0
4225	402840	1975528	3,41E+0	514070	2149160	1,98E+1
16641	1600280	7784168	2,97E+1	2048022	8463144	2,97E+2
66049	6379032	30903016	3,02E+2	8175638	—	—

Table 6.1: Number of non-zero entries, resp., memory-usage (in bytes) and runtime (sec.) for UMFPACK direct solver

6.2. Analysis of the matrix condition numbers and preconditioning

In this section we will present important preliminaries before dealing with the special linear iterative solvers and preconditioning. Given the full discretisation of the D2Q9 discrete velocity model, we take a look at the resulting (linearized) system of equations in order to analyse the conditioning of the system matrix A. The system we take into account here is obtained without the *generalized equilibrium formulation*, as the implicit matrix is not directly given. Instead, we take the Jacobian obtained in the previous section in the Newton method, with our upwind space discretisation and fully implicit boundary conditions included in the matrix. We analyse the monolithic steady approach, but also take a look at the system obtained by an Implicit Euler time-discretisation with different Δt. Early tests using a simple damped Richardson scheme as linear solver showed a rapidly degrading convergence behaviour for $c \to \infty$, so a detailed analysis of the condition $\kappa = \kappa(c)$ was necessary. Furthermore, in view of applying preconditioners, two aspects are of main interest:

- What spectrum of eigenvalues is characterizing A and how is it influenced by parameters c, h, Δt or by the matrix' parts (basically the transport part and the collision part)?
- What is in short the overall condition κ and how can it be improved in view of iterative solvers by special preconditioning?

As testcase was chosen the *driven cavity* configuration on a structured grid at $\mathrm{Re} = 10$ and the distribution of all eigenvalues for 1st order uwpind was obtained with sufficient accuracy by 100 iterations of the QR-algorithm. The first Figures 6.2–6.4 show results for the time-dependent system at $\Delta t = 1$, while together with cases $\Delta t = \infty$ (monolithic approach) and the easier configuration $\Delta t = 0.1$ condition numbers due to extreme values are given in the tables 6.2–6.4. The spectrum plotted in Figure 6.2 for initially $n = 81$ gridpoints shows for the eigenvalues a strong dependence on parameter c. For $c = 1$ the distribution is moderate, with maximum values at 32 and a number of lowest values at 1. Looking still at the first figure, we see that with increasing c to 10 and 100 the conditioning of the system matrix becomes ill with the largest eigenvalues growing fast with c. Nevertheless, we have throughout positive values due to the upwind-disctretization of the differential term.
In the following tables the influence of c onto the condition of the plain system is also obvious, being almost of order $O(c^2)$. The resulting κ for the case $\Delta t = 0.1$ (Table 6.2) is moderate compared to the case $\Delta t = \infty$ (Table 6.4). The conditioning is also influenced by the refinement level h. In the left column of all tables we see that λ_{max} is of order $O(h^{-1})$, additionally λ_{min} in Table 6.4 is decreasing with h resulting in a strong h-dependence even quadratic with the number of unknowns for the direct solution approach. The time-dependent cases always show $\lambda_{min} = 1$ and an interesting observation is that for the extreme case of $c = 100$ almost no level-dependence of the condition number is visible, like in the moderate cases.
The two observed influences we attribute to the discretisation of the collision term (c-dependence) and the transport term (h-dependence). We expected to improve condition numbers by special preconditioning techniques, for example remove the level-dependence (if existing) by transport preconditioning. In order to verify this assumption we tried to systematically analyse parts of the system, by extracting them from the matrix and computing their inverses which we applied to A. The large transport-blocks gathered into a matrix T and the collisions on the (off-)diagonals assembled into a matrix C consisting of small 9×9 blocks were obvious candidates.

Operator T consisting of $N \times N$ transport blocks:

In the middle columns of Tables 6.2–6.4 we present the corresponding eigenvalues of the matrix $T^{-1}A$. The largest eigenvalues are throughout bounded by 1.8. On the other hand, λ_{min}

was also reduced, but the overall condition number is improved by an order of magnitude, being level-independent except for the case $c = 100$. The case $\Delta t = \infty$ also is an exception, but here the condition number is even lowered by two orders. In Figure 6.3 we look at the influence of transport-preconditioning on the time-dependent system for the whole range of c, showing a clustering which is especially favourable for Krylov-space iterative solvers, although eigenvalues close to zero for $c = 100$ can be problematic. Figure 6.5 shows the important monolithic steady case and clearly we see once more the level-independence for the transport-dominated case $c = 1$, where the distribution of eigenvalues shows a strong clustering around 1. With this configuration we can expect to obtain the steady-state solution efficiently, even with increasing refinement level.

Operator C due to local 9×9 collision blocks:

The eigenvalues of $C^{-1}A$ now all have negative sign, as seen in Fig. 6.4, but are still bounded inside $[-2,0)$, the condition is mainly influenced by λ with smallest absolute value. The right columns of the tabulated results show that the collision preconditioned system is quite robust against increasing c, especially for a small system size. The level-dependence, however, is much stronger than for the matrix modified by inverse transport blocks. An interesting observation can be made in the last Figures 6.6–6.8 presented for the monolithic steady approach. The eigenvalues of $C^{-1}A$ show a specific allocation around -1, an extreme example is given by $c = 100$ where all values are almost aligned. Higher refinement levels mean more deviations from this line, and for $c = 1$ the values are more scattered with just a small group of -1 values.

In summary we showed that the proposed preconditioners improve the condition numbers, depending however on the range for c and h. Convection dominated configurations can be solved very efficiently using the lower triangular transport blocks resulting from our special numbering technique. A strong clustering effect on the eigenvalues is additionally favourable for Krylov-space solvers. Stiff configurations due to large c can be also dealt with, using a block-Jacobian approach for the local collisions. On the way to combine both techniques we developed the *generalized equilibrium formulation* approach, we will give results only numerically in Section 9.2, without eigenvalue analysis because the GEF system matrix is only implicitly known.

	plain			tr-pre			bl-jac		
grid	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ
				c=1					
81	4.1E+0	1.0E+0	4.1E+0	1.1E+0	8.4E-1	1.3E+0	1.4E+0	5.9E-1	2.4E+0
289	7.3E+0	1.0E+0	7.3E+0	1.1E+0	8.4E-1	1.3E+0	1.6E+0	3.6E-1	4.5E+0
1089	1.4E+1	1.0E+0	1.4E+1	1.1E+0	8.4E-1	1.3E+0	1.8E+0	2.0E-1	8.8E+0
				c=10					
81	6.3E+1	1.0E+0	6.3E+1	1.7E+0	1.0E-1	1.7E+1	1.7E+0	2.4E-1	7.0E+0
289	9.5E+1	1.0E+0	9.5E+1	1.8E+0	8.5E-2	2.1E+1	1.9E+0	1.0E-1	1.8E+1
1089	1.6E+2	1.0E+0	1.6E+2	1.8E+0	7.4E-2	2.4E+1	1.9E+0	4.6E-2	4.3E+1
				c=100					
81	3.6E+3	1.0E+0	3.6E+3	1.8E+0	6.1E-3	2.9E+2	1.8E+0	1.4E-1	1.3E+1
289	3.9E+3	1.0E+0	3.9E+3	1.8E+0	3.6E-3	5.0E+2	1.9E+0	4.2E-2	4.7E+1
1089	4.6E+3	1.0E+0	4.6E+3	1.8E+0	2.2E-3	8.2E+2	2.0E+0	1.2E-2	1.6E+2

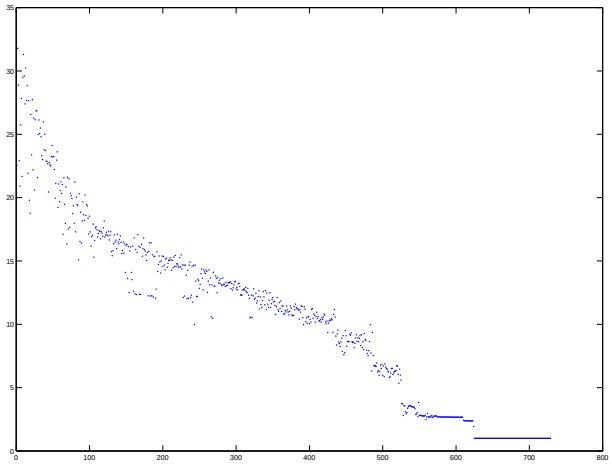
Table 6.2: Extrema of absolute eigenvalues and condition numbers of the system matrix, $\mathrm{Re} = 10$, $\Delta t = 0.1$

	plain			tr-pre			bl-jac		
grid	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ
				c=1					
81	3.2E+1	1.0E+0	3.2E+1	1.4E+0	5.4E-1	2.7E+0	1.7E+0	3.4E-1	5.0E+0
289	6.4E+1	1.0E+0	6.4E+1	1.4E+0	5.4E-1	2.7E+0	1.8E+0	1.6E-1	1.1E+1
1089	1.3E+2	1.0E+0	1.3E+2	1.4E+0	5.4E-1	2.6E+0	1.9E+0	8.0E-2	2.4E+1
				c=10					
81	6.2E+2	1.0E+0	6.2E+2	1.8E+0	6.7E-2	2.7E+1	1.8E+0	1.7E-1	1.0E+1
289	9.4E+2	1.0E+0	9.4E+2	1.8E+0	4.8E-2	3.7E+1	1.9E+0	6.0E-2	3.2E+1
1089	1.6E+3	1.0E+0	1.6E+3	1.8E+0	3.7E-2	4.8E+1	2.0E+0	2.3E-2	8.7E+1
				c=100					
81	3.6E+4	1.0E+0	3.6E+4	1.8E+0	5.7E-3	3.2E+2	1.8E+0	1.3E-1	1.3E+1
289	3.9E+4	1.0E+0	3.9E+4	1.8E+0	3.1E-3	5.8E+2	1.9E+0	3.7E-2	5.3E+1
1089	4.6E+4	1.0E+0	4.6E+4	1.8E+0	1.8E-3	1.0E+3	2.0E+0	1.0E-2	1.9E+2

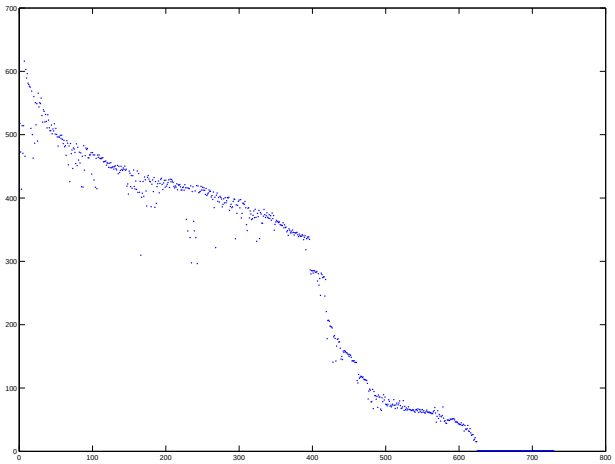
Table 6.3: Extrema of absolute eigenvalues and condition numbers of the system matrix, Re $= 10$, $\Delta t = 1$

	plain			tr-pre			bl-jac		
grid	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ
				c=1					
81	2.9E+1	1.1E-2	2.8E+3	1.7E+0	6.9E-3	2.5E+2	2.0E+0	3.7E-3	5.3E+2
289	6.3E+1	4.9E-3	1.3E+4	1.8E+0	3.2E-3	5.7E+2	2.0E+0	8.5E-4	2.3E+3
1089	1.3E+2	2.4E-3	5.3E+4	1.9E+0	1.5E-3	1.3E+3	2.0E+0	2.0E-4	9.8E+3
				c=10					
81	6.2E+2	4.1E-2	1.5E+4	1.8E+0	1.7E-3	1.0E+3	1.9E+0	6.8E-3	2.7E+2
289	9.4E+2	1.8E-2	5.3E+4	1.8E+0	7.0E-4	2.5E+3	1.9E+0	1.4E-3	1.4E+3
1089	1.6E+3	7.6E-3	2.1E+5	1.8E+0	2.9E-4	6.1E+3	2.0E+0	3.0E-4	6.5E+3
				c=100					
81	3.6E+4	5.8E-2	6.2E+5	1.8E+0	2.3E-4	7.9E+3	1.8E+0	9.0E-3	2.0E+2
289	3.9E+4	3.0E-2	1.3E+6	1.8E+0	1.1E-4	1.7E+4	1.9E+0	2.1E-3	9.0E+2
1089	4.6E+4	1.4E-2	3.2E+6	1.8E+0	5.0E-5	3.6E+4	2.0E+0	5.0E-4	3.9E+3

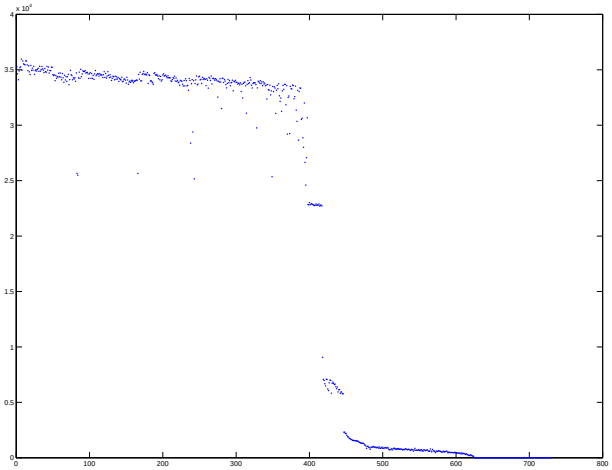
Table 6.4: Extrema of absolute eigenvalues and condition numbers of the system matrix, Re $= 10$, $\Delta t = \infty$



(a) $c = 1$

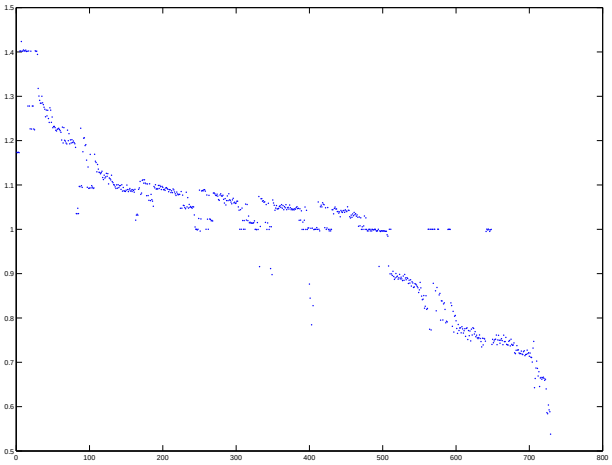


(b) $c = 10$

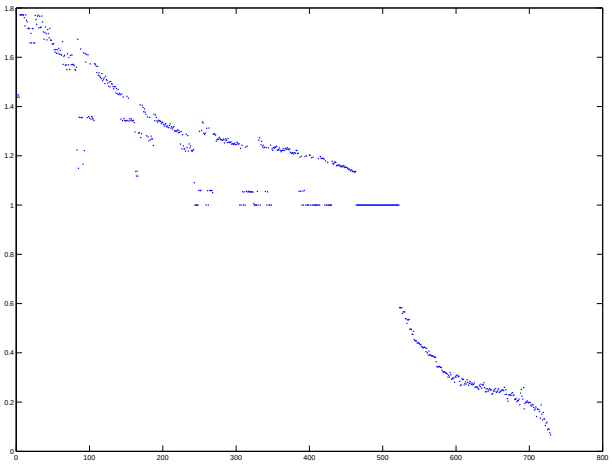


(c) $c = 100$

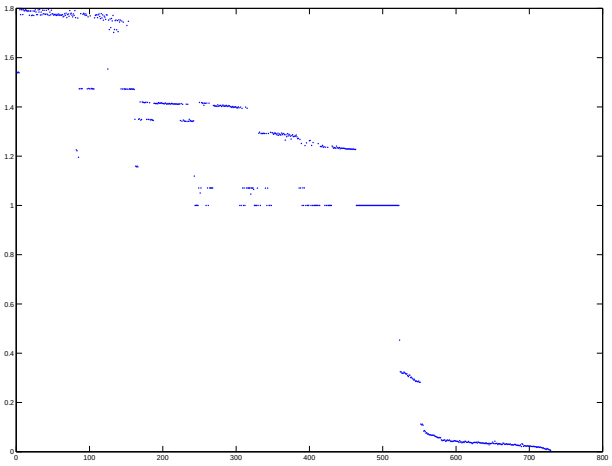
Figure 6.2: Eigenvalues for $\Delta t = 1$: c-dependence of system



(a) $c = 1$

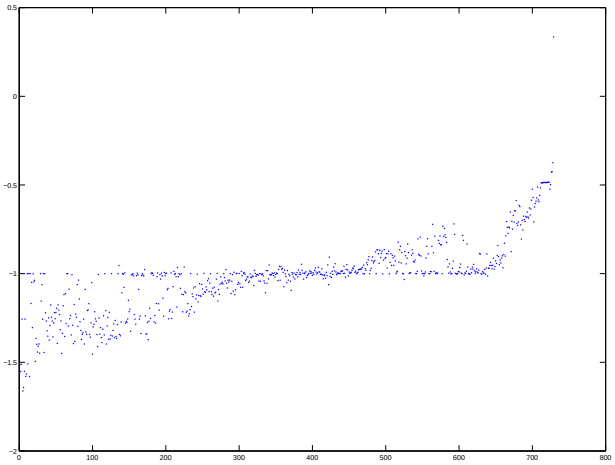


(b) $c = 10$

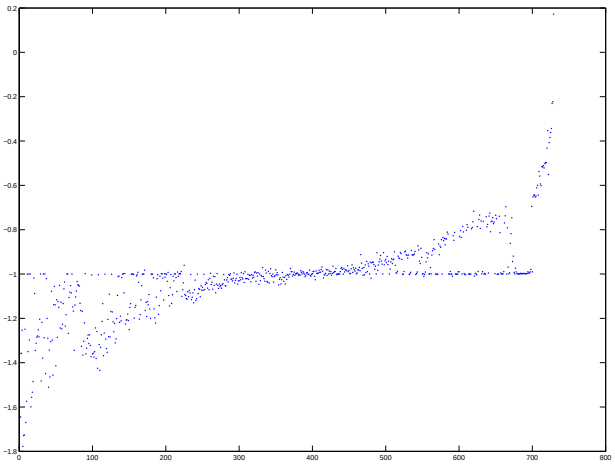


(c) $c = 100$

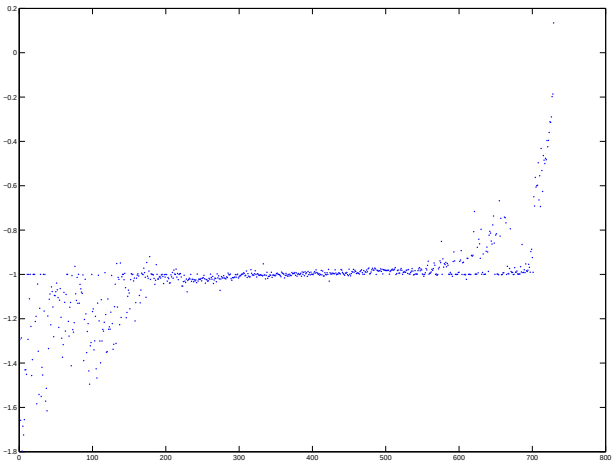
Figure 6.3: Eigenvalues for $\Delta t = 1$: c-dependence of the system multiplied by transport inverse



(a) $c = 1$

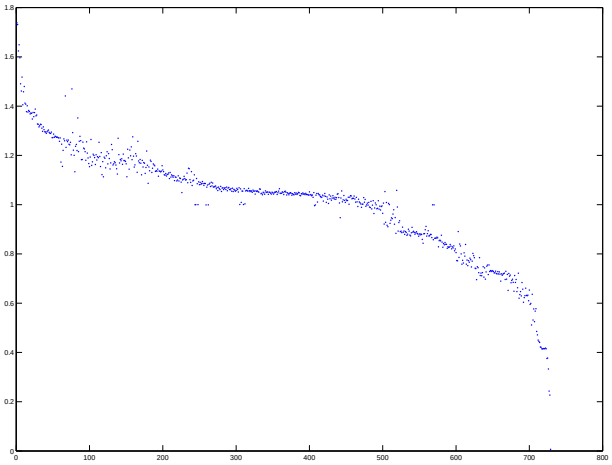


(b) $c = 10$

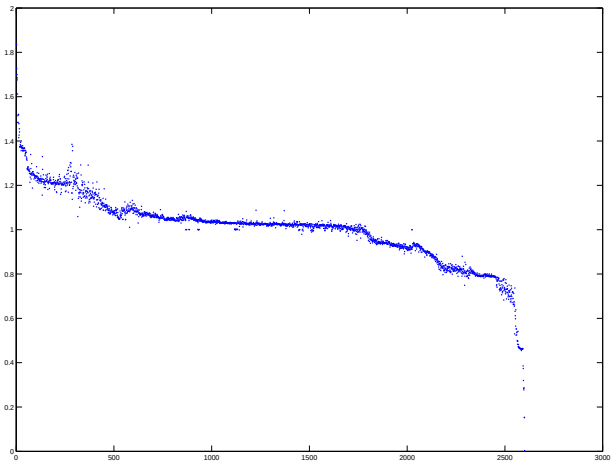


(c) $c = 100$

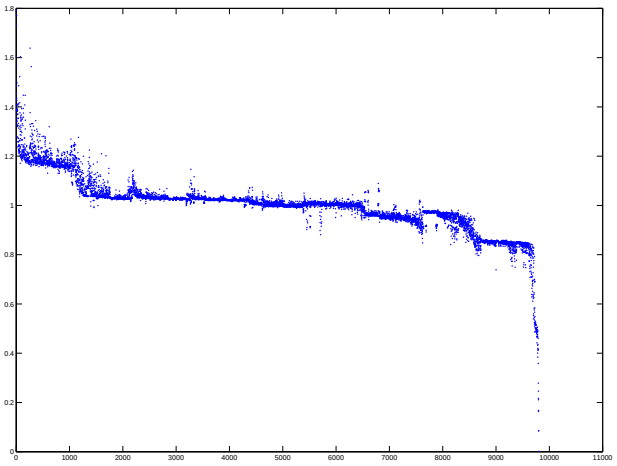
Figure 6.4: Eigenvalues for $\Delta t = 1$: c-dependence of the system multiplied by collision inverse



(a) $n = 81$

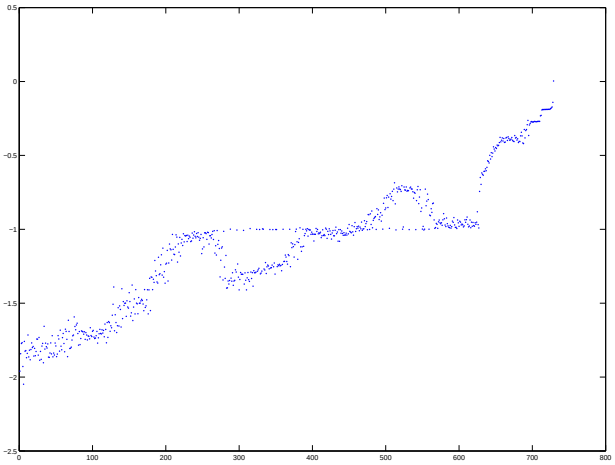


(b) $n = 289$

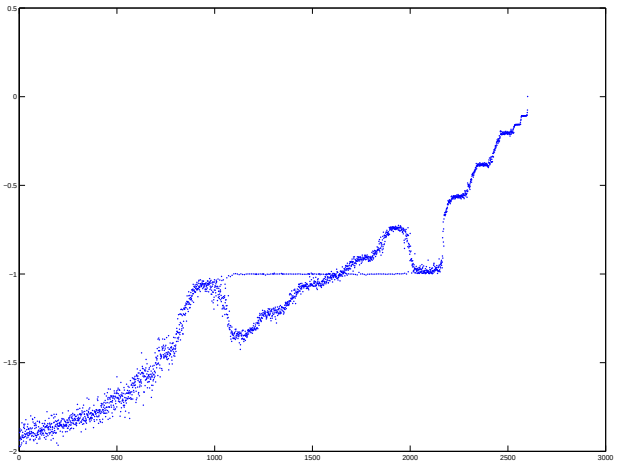


(c) $n = 1089$

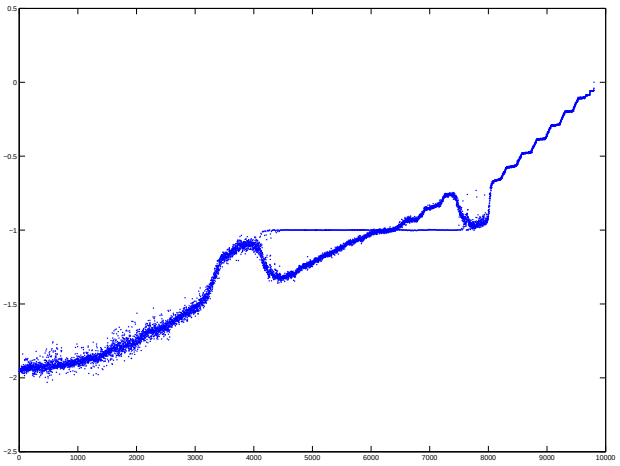
Figure 6.5: Eigenvalues for $\Delta t = \infty$: Level independence of transport preconditioning $c = 1$



(a) $n = 81$

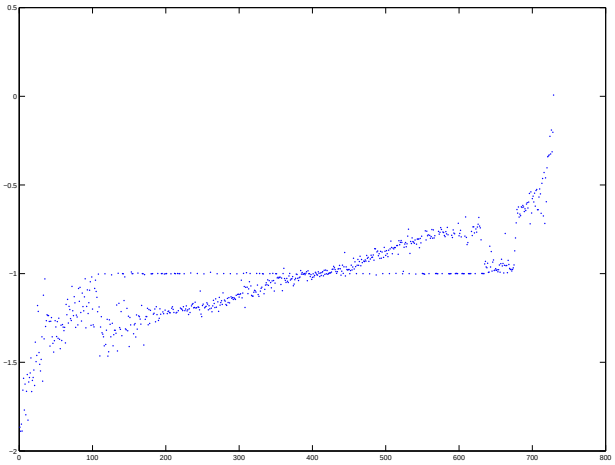


(b) $n = 289$

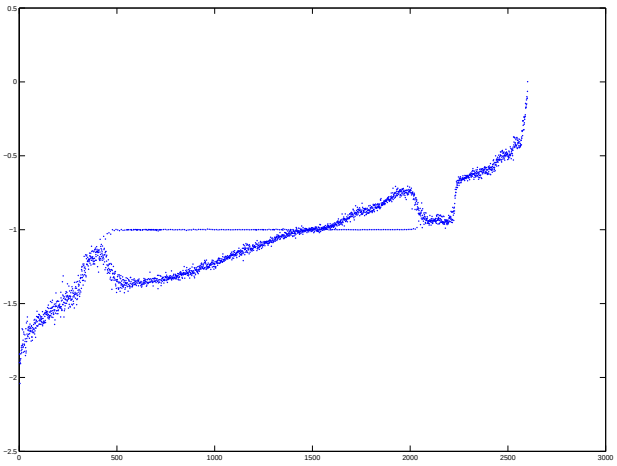


(c) $n = 1089$

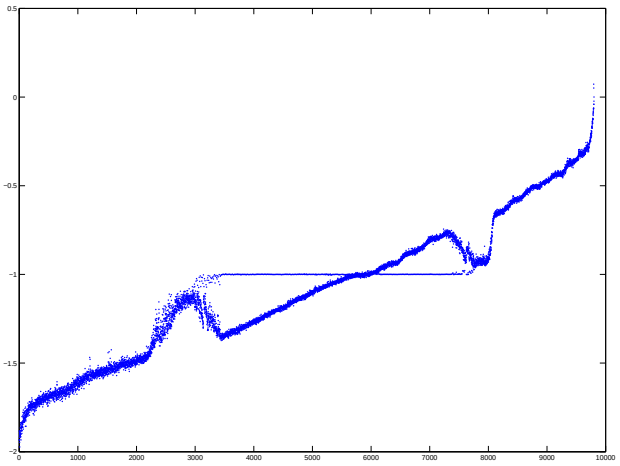
Figure 6.6: Eigenvalues for $\Delta t = \infty$: Level-dependence of bl-jac preconditioning $c = 1$



(a) $n = 81$

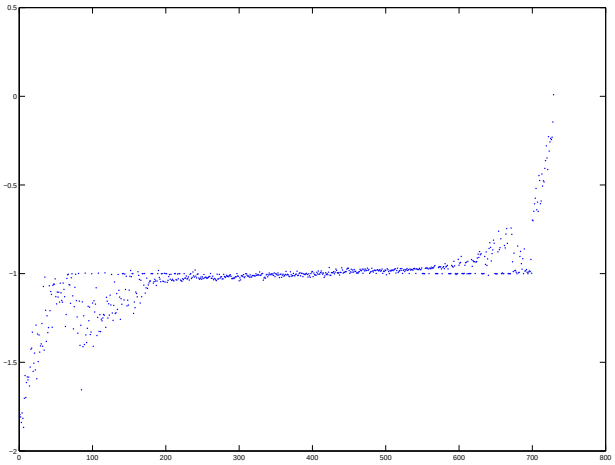


(b) $n = 289$

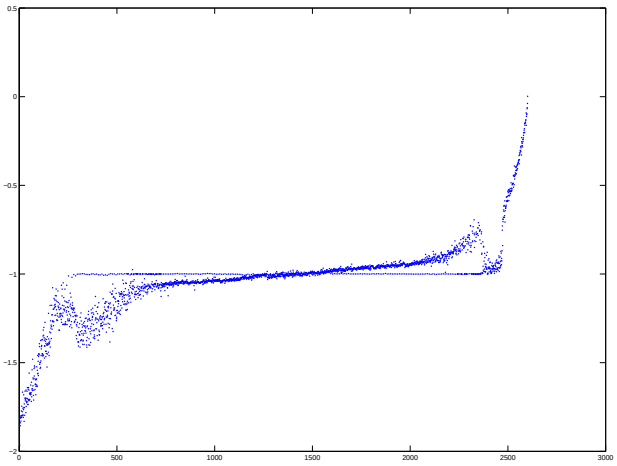


(c) $n = 1089$

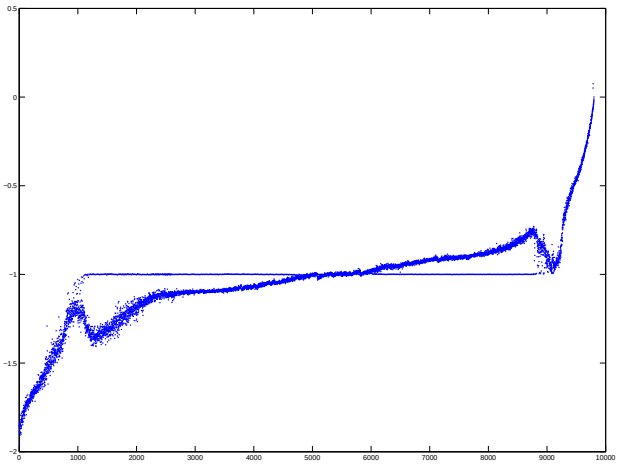
Figure 6.7: Eigenvalues for $\Delta t = \infty$: Level-dependence of bl-jac preconditioning $c = 10$



(a) $n = 81$



(b) $n = 289$



(c) $n = 1089$

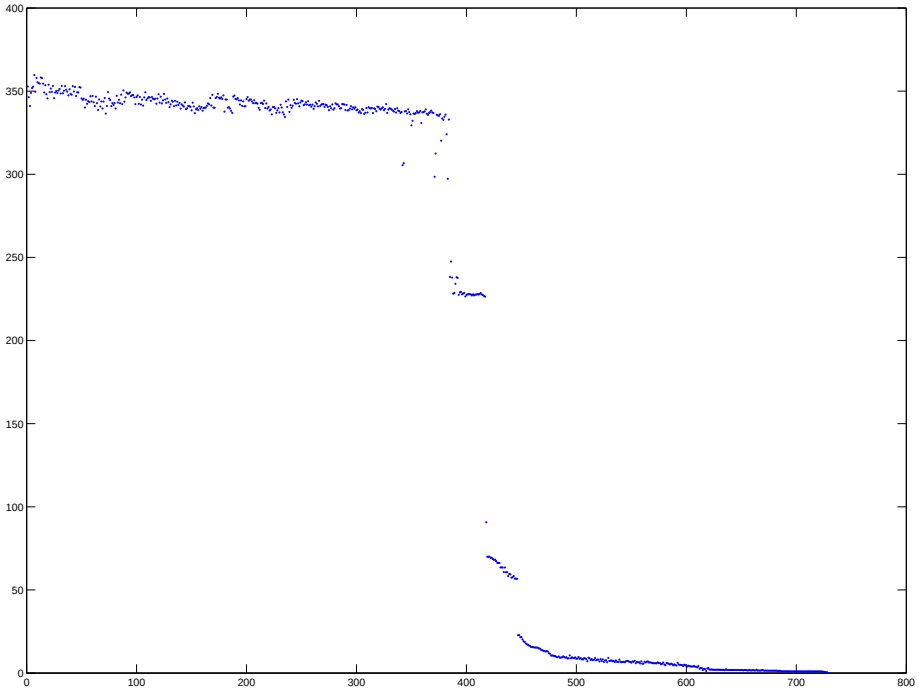
Figure 6.8: Eigenvalues for $\Delta t = \infty$: Level-dependence of bl-jac preconditioning $c = 100$

6.3. Scaling

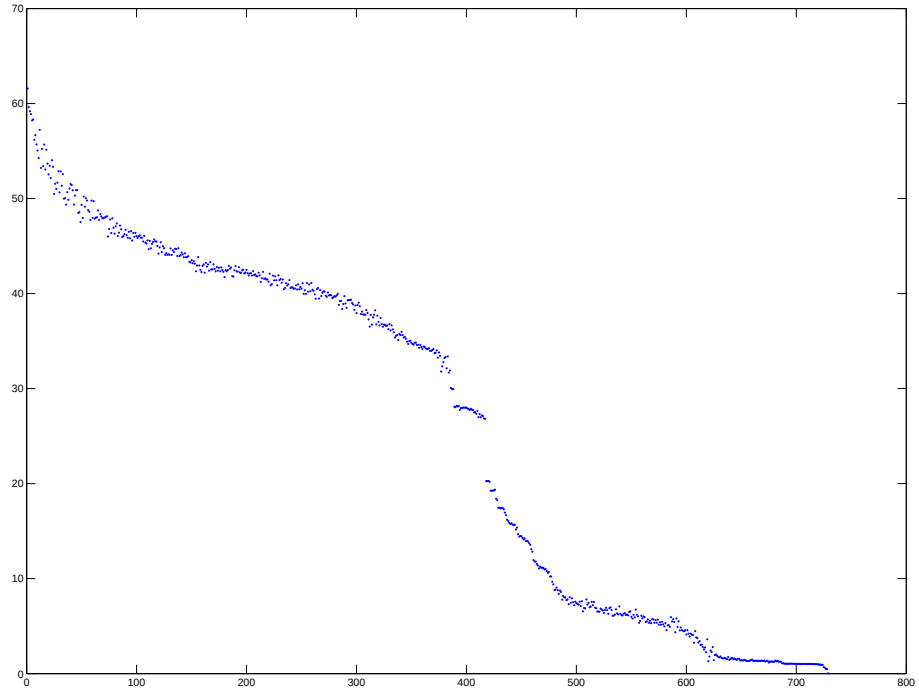
Before closing the aspect of conditioning as presented previously, we take again a closer look at the given system of equations. Especially, aiming towards a monolithic steady solver, we saw that the lack of a small Δt and any identity term as in Eq. (4.2) yielded us very ill-conditioned matrices for large parameters up to $c = 100$. We consider again equation (4.5) wherein c appears in the transport-term on the left and as c^2 in the collisions on the right. The relaxation time $\frac{1}{\tau} = \frac{c^2}{3\nu}$ can reach very large values for example due to low viscosity $\nu = 0.001$ in the *flow around cylinder* benchmark. We computed the eigenvalues again, but this time we tried to improve numerical stability by scaling the whole equation (4.5) by $\frac{1}{c}$ and in fact the condition could be improved significantly as seen in Table 6.5 and Figure 6.9. However, this applies only to the plain system A, the 'preconditioned' matrices $T^{-1}A$ and $C^{-1}A$ yielded exactly the same results as previously. Obviously the scaling factor is already included therein.

	plain			tr-pre			bl-jac		
grid	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ	λ_{max}	λ_{min}	κ
				c=1					
81	2.9E+1	1.1E-2	2.8E+3	1.7E+0	6.9E-3	2.5E+2	2.0E+0	3.7E-3	5.3E+2
289	6.3E+1	4.9E-3	1.3E+4	1.8E+0	3.2E-3	5.7E+2	2.0E+0	8.5E-4	2.3E+3
1089	1.3E+2	2.4E-3	5.3E+4	1.9E+0	1.5E-3	1.3E+3	2.0E+0	2.0E-4	9.8E+3
				c=10					
81	6.2E+1	2.0E-2	3.1E+3	1.8E+0	1.7E-3	1.0E+3	1.9E+0	6.8E-3	2.7E+2
289	9.4E+1	8.4E-3	1.1E+4	1.8E+0	7.0E-4	2.5E+3	1.9E+0	1.4E-3	1.4E+3
1089	1.6E+2	3.5E-3	4.5E+4	1.8E+0	2.9E-4	6.1E+3	2.0E+0	3.0E-4	6.5E+3
				c=100					
81	3.6E+2	2.6E-2	1.4E+4	1.8E+0	2.3E-4	7.9E+3	1.8E+0	9.0E-3	2.0E+2
289	3.9E+2	1.2E-2	3.2E+4	1.8E+0	1.1E-4	1.7E+4	1.9E+0	2.1E-3	9.0E+2
1089	4.6E+2	5.8E-3	7.9E+4	1.8E+0	5.0E-5	3.6E+4	2.0E+0	5.0E-4	3.9E+3

Table 6.5: Extrema of absolute eigenvalues and condition numbers of the system matrix, $\mathrm{Re} = 10$, $\Delta t = \infty$, scaled

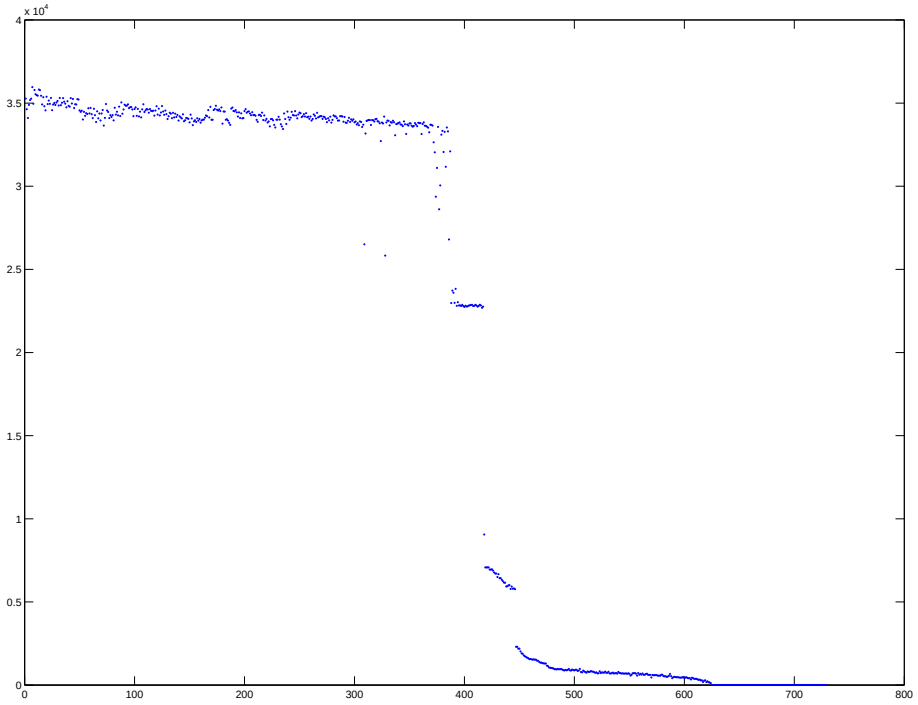


(a) plain

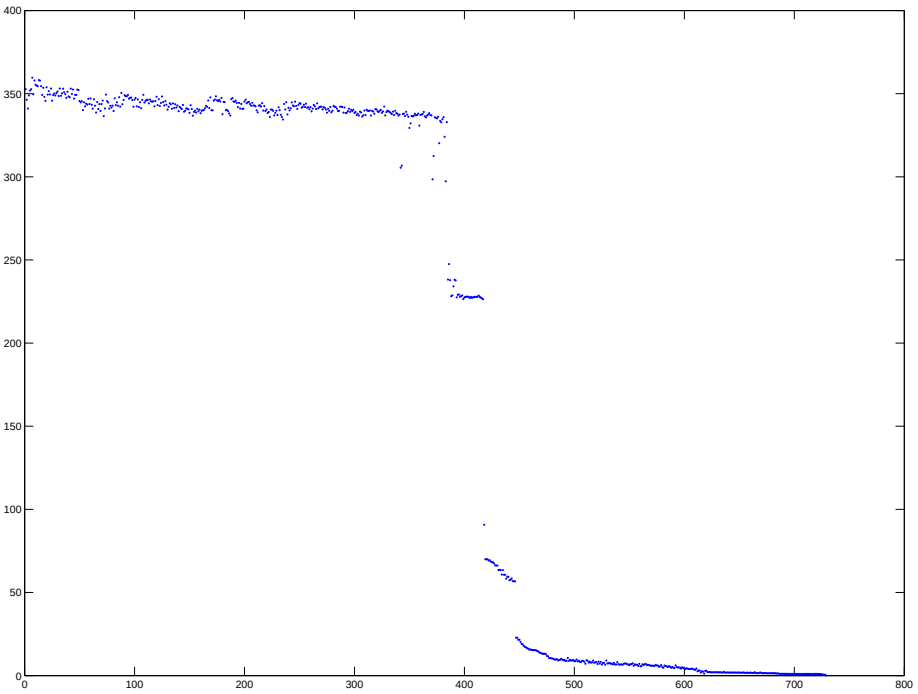


(b) scaled

Figure 6.9: Eigenvalues: scaling of the system by $1/c$, $c = 10$



(a) plain



(b) scaled

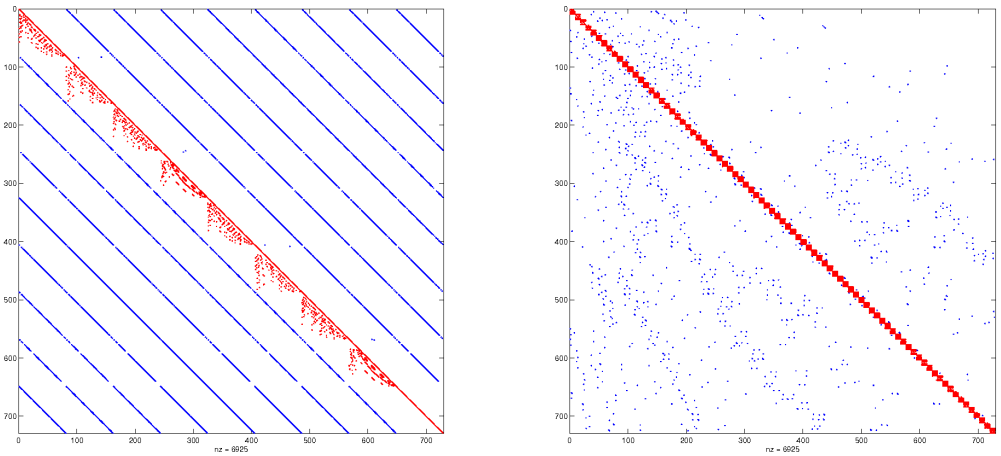
Figure 6.10: Eigenvalues: scaling of the system by $1/c$, $c = 100$

6.4. Preconditioning techniques

As hinted initially in this chapter, a basic example of improving linear convergence is to apply suitable preconditioning in the Richardson iteration

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \sigma\widetilde{A}^{-1}(A\mathbf{x}^{(n)} - b)$$

where $\widetilde{A}$ should be easy to invert in order to apply it efficiently in numerous iteration steps. At the same time, while usually it is not feasible to take $\widetilde{A} = A$, the preconditioner should be 'similar' to A in the sense that $\widetilde{A}^{-1}A \sim I$. This suggests to take one part of the matrix (as in Fig. 6.11) that dominates the system, while the structure is sufficiently simple or, for that matter, solvable. Once we find the appropriate $\widetilde{A}$, we can substitute the expression $\widetilde{A}^{-1}A$ for A as iteration matrix in any given iterative (Krylov-space) method to solve the modified system $\widetilde{A}^{-1}A = \widetilde{A}^{-1}b$ with significantly improved convergence. We showed in Section 6.2 that the condition number was reduced by multiplying A with the inverse block-diagonal which in one case consists of the transport matrices. Accordingly, we will describe the so-called transport-preconditioner in the first place. In the second part we will discuss an alternative preconditioner that focuses on the collisions. Finally, the GEF model allows advanced techniques which are in the presented form not possible in the original discretisation.



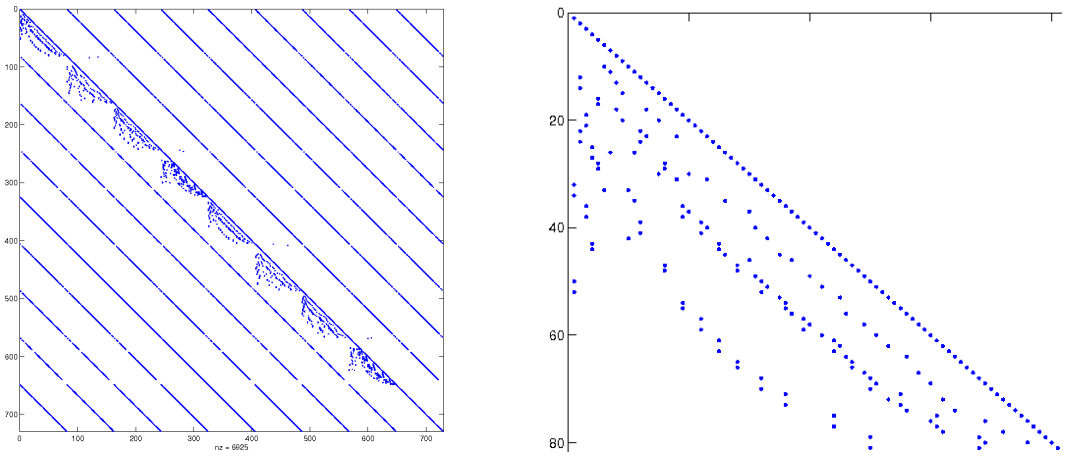
(a) Block-diagonal of transport matrices

(b) Collisions on block-diagonal

Figure 6.11: Matrix allocation depending on primary index (and sorting)

6.4.1. Direct transport preconditioning (tr-pre)

Our special numbering technique as described in Section 4.2.3 makes it possible to turn our upwind characteristic discretisation into a very efficient preconditioner. Especially for transport dominated configurations, it is quite obvious to choose the convection part as the analogon $\widetilde{A}$ to the whole system A. In this case we improve the condition number of $\widetilde{A}^{-1}A$ significantly and independently of the mesh size as seen from our eigenvalue analysis. Furthermore, the resulting clustering of the eigenvalues is advantageous for Krylov-space methods like BiCG-Stab or GMRES. These and other iterative solvers sometimes need many 'solution steps' of the preconditioner, which accordingly needs to be accomplished very efficiently. Here we can directly invert $\widetilde{A}$ consisting of lower triangular blocks — taking for every characteristic the prepared numbering as shown in Fig. 6.12. This results in a linear runtime in the number of unknowns, additionally we find a parallelization approach in the fact that transport blocks for the different characteristics are solved independently of each other. In the D2Q9 model 9 processors could be used in parallel, while in 3D the number would increase to 15 or 19 depending on the model and set of used lattice vectors.
The transport preconditioning is closely connected to the *generalized equilibrium formulation* introduced in Section 4.5. An improvement of the condition number due to 'tr-pre' is there already built in, as the inverse transport becomes part of the GEF equations. So the linear convergence of iterative solvers using the plain (meaning without preconditioning) GEF is expected to be comparable to results obtained with 'tr-pre' as preconditioner. A further improvement of 'tr-pre' used in the basic discretisation showed to be only achievable by the multigrid method where the use is feasible and can be accomplished with high efficiency. In case of stiff configurations due to low Mach number avantages using 'tr-pre' cannot be expected, so an alternative approach is presented next.



(a) Special characteristic sorting (b) Transport-block for one direction

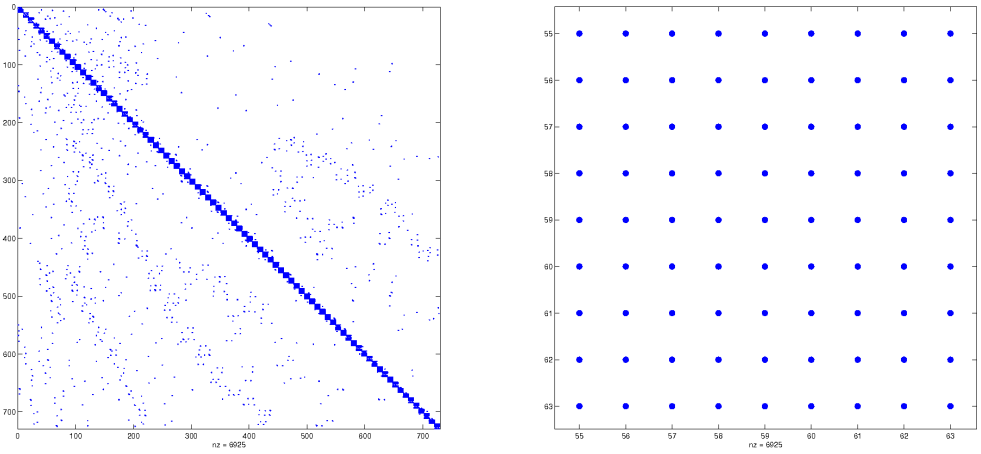
Figure 6.12: Preconditioning by transport part

6.4.2. Block-diagonal collision preconditioning (bl-jac)

Our spectral analysis clearly showed that large values of c cause for the linear system a very bad condition number, which is even beyond remedy of the proposed transport-preconditioner. This is due to the collisions situated on the off-diagonals (see Fig. 6.12a) becoming dominant as they scale with $-\frac{1}{\tau} = -\frac{c^2}{3\nu}$. As alternative to 'tr-pre', we constructed a 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 distributions $f_i,\ i = 1,\ldots,9$ for all discrete velocities in the first point x_1, then the next 9 variables for x_2, and so on for all remaining points. Consequently, the local collisions consist of 9×9 blocks on the diagonal. At the same time, the entries for the spacial coupling due to transport are freely scattered off the block-diagonal (see Fig. 6.13). The solution process of the preconditioner (to be denoted 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 (with n denoting the number of grid points), i.e.

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

which we can calculate by a 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 (although the constant in the expression $O(n)$ is not small). 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 chapter concerning preconditioning will show a technique which combines the advantages of both numbering techniques.



(a) Local collision numbering (b) One 9×9 block

Figure 6.13: Block-Jacobian preconditioning by collision part

6.4.3. Special preconditioning of the GEF system-matrix

In Section 4.5 we described how the inverse transport became part of the *generalized equilibrium formulation*, and how 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. The first candidate to improve the condition number follows directly the proposed collision preconditioning from Sec. 6.4.2, while the second candidate uses advanced multigrid methods.

Additional preconditioning by collisions

The GEF matrix at hand (see Eq. 4.22) consists mainly of weighted inverses of transport matrices, easily to be solved due to their lower triangular form. Following basic linear algebra theory, we take 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

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

although the implementation is matrix-free. However, applying C^{-1} in iterative solvers we expect an additional stabilising effect in the case of large c (or collision rate $\frac{1}{\tau}$), while the solution of the GEF even without preconditioning should perform well for transport dominated configurations due to the fundamental construction. Results in Sec. 9.2 confirm the suppositions for the scheme denoted as $GEF(\backslash)$.

Multigrid as preconditioner

Instead of the quite simple collision preconditioning, we propose an alternative scheme using the multigrid method. Suppose we apply the GMRES iterative solver for the *generalized equilibrium formulation* which intrinsically holds the inverse transport. In place of the preconditioner $\mathbf{C}^{-1}$ in algorithm 6.5.2 we call a multigrid routine with a given accuracy. For example, gaining one or two digits in each multigrid call will boost significantly the convergence of the outer GMRES algorithm. It is also possible to run only one full multigrid cycle as preconditioner, sweeping from the given level L down to coarsest level 1, with adjustable number of smoothing steps after assembling the intermediate results (see Alg. 6.6.1). As smoother can be chosen an arbitrary iterative scheme with good smoothing property giving optimal overall convergence, in practice 16 GMRES steps of the plain GEF is a good choice. Repeated MG cycles and smoothing steps can be performed then with good numerical efficiency due to the fast algorithm directly solving the transport steps. As coarse grid solver it is applicable to use the GEF approach with collision preconditioning, because GEF($\backslash$) is supposed to yield good linear convergence even for stiff configurations in case

the number of unknowns on the coarsest level is not too high. Using multigrid as preconditioner in Krylov-space iterative schemes we expect advantages compared to the basic algorithm which is 'just' iterating MG cycles to reduce the linear defect.

Remark:

In contrast to the GEF which allows the combination of two efficient and robust preconditioning techniques, we regard the operator splitting approach. We have introduced two efficient solvers aimed for the operators appearing in the DVM. As we can easily deal with transport and collision independently, the idea to apply operator splitting techniques to solve the complete system is quite obvious. However, we did not follow the concept beyond some initial tests, because it contradicts the fully implicit (monolithic) approach towards solving the Boltzmann equation. Operator splitting often needs to revert to small time-steps for stability reasons, which our tests confirmed. Nevertheless, one should keep in mind the described preconditioners in this context for possible future research.

6.5. Krylov-space iterative solvers

The class of Krylov-space methods consists of various algorithms like the CG method for symmetric matrices or, for more general problems, the improved BiCG-Stab algorithm 6.5.1. In short, the method iterates the solution with reference to the matrices A and A^T but without actually computing the transposed (see see [48]). Three auxiliary vectors are sufficient for an implementation of the method, it means a small usage of memory, especially compared to direct linear solvers. In each BiCG iteration two matrix vector multiplications are performed. In view of solving the preconditioned system

$$\begin{aligned} Ax &= b \\ \widetilde{A}^{-1}Ax &= \widetilde{A}^{-1}b \end{aligned}$$

it means corresponding two solution steps of the preconditioner, which can be the solution of a lower triangular matrix or, somewhat more expensive, a multigrid step, each. Furthermore, the GMRES [38] method given in Alg. 6.5.2 is a commonly popular linear solver, and we used it extensively to obtain results. In contrast to the previous algorithm, GMRES stores all solution vectors and uses them to compute the next iteration. For large systems, this can mount up and exceed available memory before the stopping criterion is satisfied, in this case a restart strategy is applied when the maximum number of iterations — the so-called restart-level — is reached. However, only one matrix-vector operation is performed in every GMRES step.
The main advantage of GMRES over BiCG-Stab is its more monotone convergence behaviour. The defect is decreasing in every step, so much the faster, in case the stored basis is considerably larger (see Figs. 6.14 – 6.16). In contrast, the changes in the defect of the BiCG-Stab are larger but not monotonously decreasing, it reaches the final convergence usually faster than GMRES, but can fail for ill-conditioned systems. We obtained very good and stable convergence rates using GMRES in combination with preconditioning by the multigrid method. The results were considerably better than using the multigrid algorithm 9.4.1 as linear solver (see results in Sec. 9.3).

Algorithm 6.5.1 (BiCG-Stab method)
BiCG-Stab($\mathbf{x}, \mathbf{b}, \mathbf{A}, \mathbf{C}, K$***)***
0. INIT: $\mathbf{r} = \mathbf{C}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}) \quad , \quad \hat{\mathbf{r_0}} = \mathbf{r} \quad , \quad \rho_0 = \alpha = \omega_0 = 1, \quad , \quad \mathbf{v} = \mathbf{p} = \mathbf{0}$

1. DO WHILE $(|r| >$ *stopping criterion)* ***AND*** $k < K$
$k = k + 1$
$\beta = \frac{\rho_k}{\rho_{k-1}} \cdot \frac{\alpha}{\omega} \quad , \quad \mathbf{p} := \mathbf{r} + \beta(\mathbf{p} - \omega\mathbf{r})$
$\mathbf{v} = \mathbf{C}^{-1}A\mathbf{p}$
$\alpha = \frac{\rho_k}{\hat{\mathbf{r_0}}^T\mathbf{v}} \quad , \quad \mathbf{s} = \mathbf{r} - \alpha\mathbf{r} \quad , \quad \mathbf{t} = \mathbf{C}^{-1}A\mathbf{s}$
$\omega = \frac{\mathbf{t}^T\mathbf{s}}{\|\mathbf{t}\|_2} \quad , \quad \rho_{k+1} = -\omega\hat{\mathbf{r_0}}^T\mathbf{t}$
$\mathbf{x} = \mathbf{x} + \alpha\mathbf{p} + \omega\mathbf{s}$
$\mathbf{r} = \mathbf{s} - \omega\mathbf{t}$

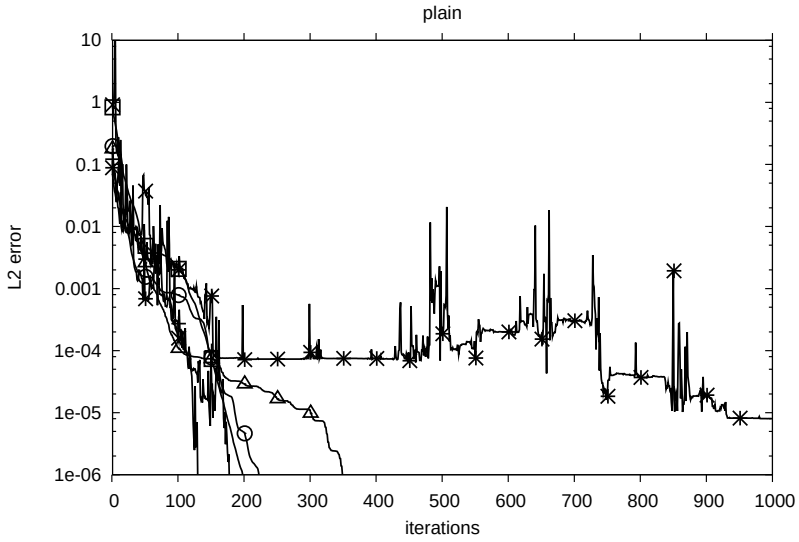
Now we preview the results from Section 9.2 in order to demonstrate the convergence behaviour of both solvers: Most obvious is the strong dependence on c in Fig. 6.14 due to the condition $\kappa = \kappa(c)$. Only the block-Jacobian preconditioning is very robust on the small chosen level with 81 gridpoints. The second aspect to be noted is the highly oscillating behaviour of the BiCG-Stab solver, especially for the extreme case of $c = 100$. The transport preconditioner gives no improvement there, either. In contrast to the BiCG solver, we see strictly monotone behaviour of the GMRES solver, the slope is at the beginning quite small, but increases for a

growing vectorspace used in the iteration. In configurations with moderate condition number, however, BiCG-Stab proves to be very efficient, even overtaking the monotone GMRES scheme. Next, in Fig. 6.15 we confront the results obtained so far with plots for the scaled system: Scaling the stationary equation by $\frac{1}{c}$ (compare Sec. 6.2), the overall convergence is greatly improved, only the results for the block-Jacobian preconditioning do not change much as this variant had large values of c already under control. Apart from that, the defect is decreasing quite monotonously, with less 'wiggles' than in the non-scaled case.
Next, we have a look at Figure 6.16 with changed configuration ($\mathrm{Re} = 100$ and increased level with 289 gridpoints) and facing 'scaled' results: With increased Reynolds number the plain schemes have difficulty to reduce the defect without help of preconditioners. The schemes using block-Jacobian preconditioner have no longer top performance due to the increased problem size, and the rates will decrease further on higher refinement levels.
All in all, in view of constructing a monolithic solver, the use of GMRES is promising to be more stable, while the BiCG-Stab/tr-pre combination still should give peak performance for transport dominated problems and time-stepping schemes (with small Δt) which are better conditioned.

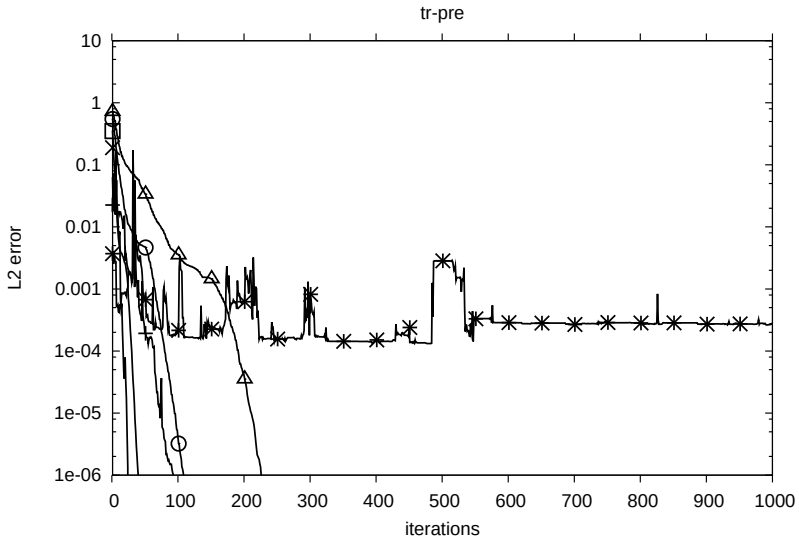
Algorithm 6.5.2 (GMRES(M) method)
GMRES($\mathbf{x}, \mathbf{b}, \mathbf{A}, \mathbf{C}, M$ ***)***
0. INIT: $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$, ***SOLUTION IF*** $\mathbf{r_0} = 0$
$\mathbf{v_1} = \frac{\mathbf{r_0}}{\|\mathbf{r_0}\|_2}$

1. DO WHILE $j < M$
FOR $i = 1, \ldots, j$: $h_{ij} = (\mathbf{v_i}, \mathbf{C}^{-1} A \mathbf{v_j})$
$\omega_j = \mathbf{C}^{-1}\mathbf{A}\mathbf{v_j} - \sum_{i=1}^{j} h_{ij}\mathbf{v_i}$
FOR $i = 1, \ldots, j-1$: $\begin{pmatrix} h_{ij} \\ h_{i+1,j} \end{pmatrix} = \begin{pmatrix} c_{i+1} & s_{i+1} \\ s_{i+1} & -c_{i+1} \end{pmatrix} \begin{pmatrix} h_{ij} \\ h_{i+1,j} \end{pmatrix}$
$\beta = \sqrt{h_{jj}^2 + h_{j+1,j}^2}$, $s_{j+1} = \frac{h_{j+1,1}}{\beta}$, $c_{j+1} = \frac{h_{jj}}{\beta}$, $h_{jj} = \beta$
$\gamma_{j+1} = s_{j+1}\gamma_j$, $\gamma_j = c_{j+1}\gamma_j$

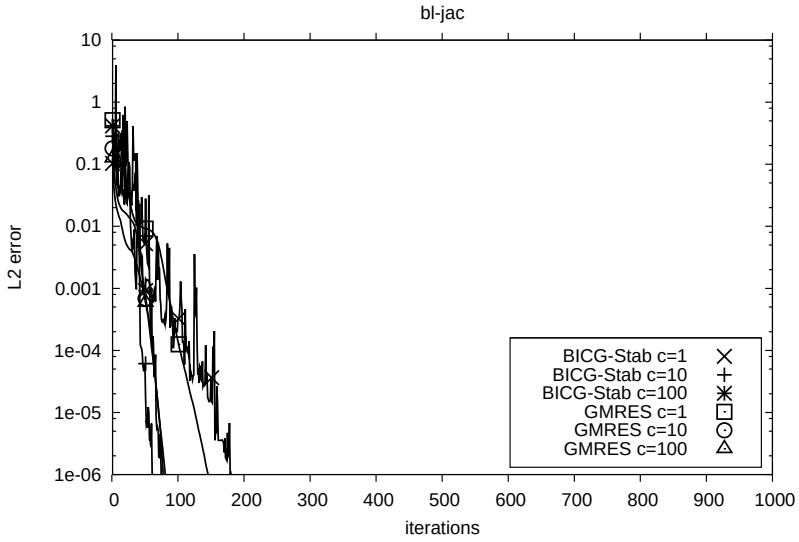
IF ($\gamma_{j+1} <$ *stopping criterion* ***) THEN***
FOR $i = j, \ldots, 1$ $\alpha_i = \frac{1}{h_{ii}}(\gamma_j - \sum_{k=i+1}^{j} h_{ik}\alpha_k)$
SOLUTION $\mathbf{x} = \mathbf{x_0} + \sum_{i=1}^{j} \alpha_i \mathbf{v_i}$
ELSE
$\mathbf{v_{j+1}} = \frac{\mathbf{w_j}}{h_{j+1,1}}$



(a) plain

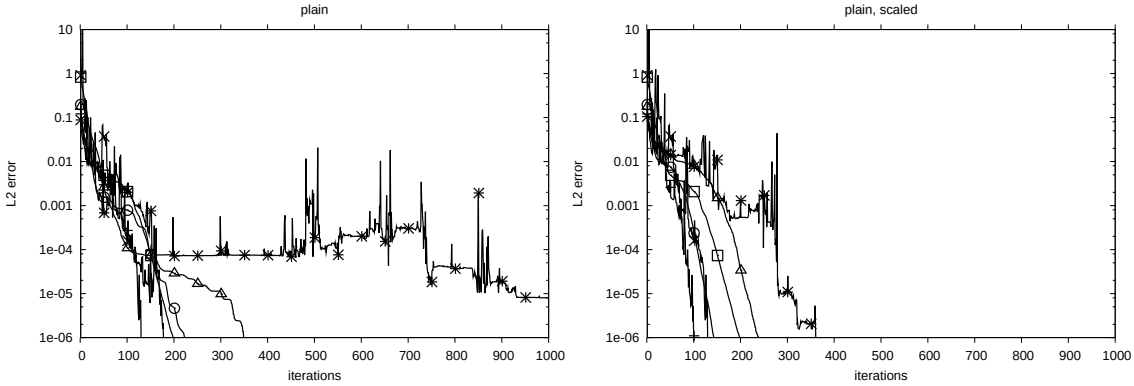


(b) tr-pre

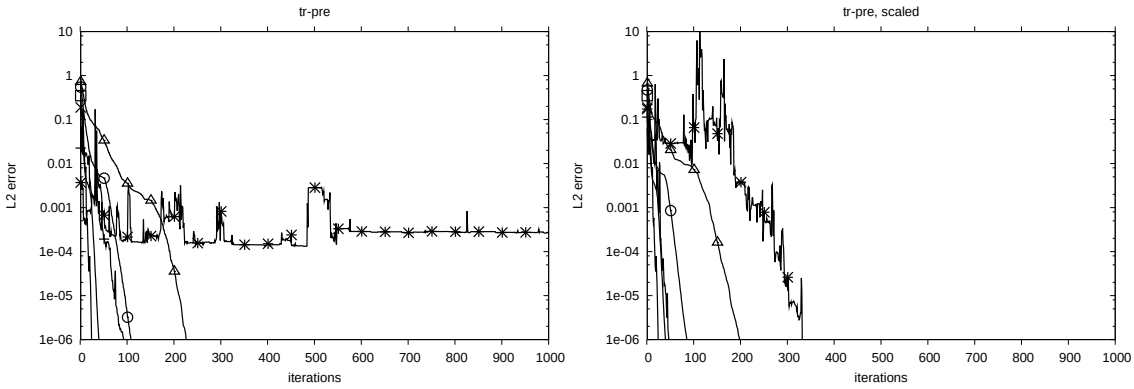


(c) bl-jac

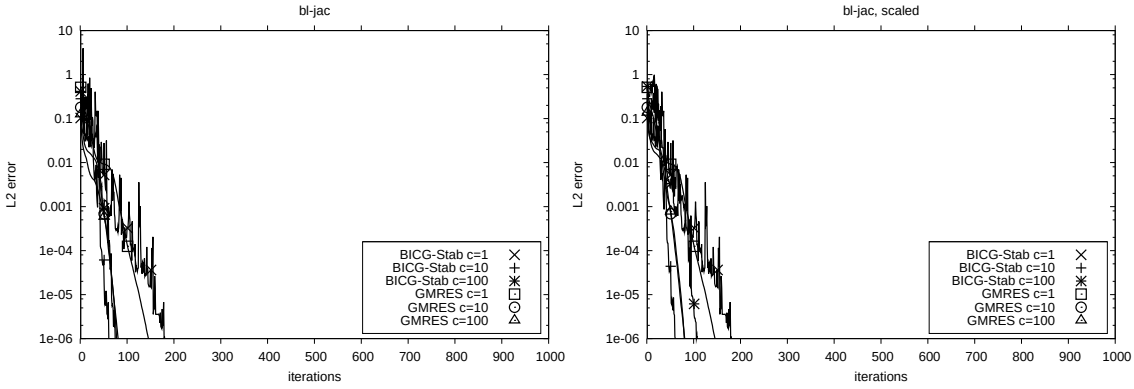
Figure 6.14: Linear convergence of BiCG vs. GMRES on 81 grid, Re = 10



(a) plain

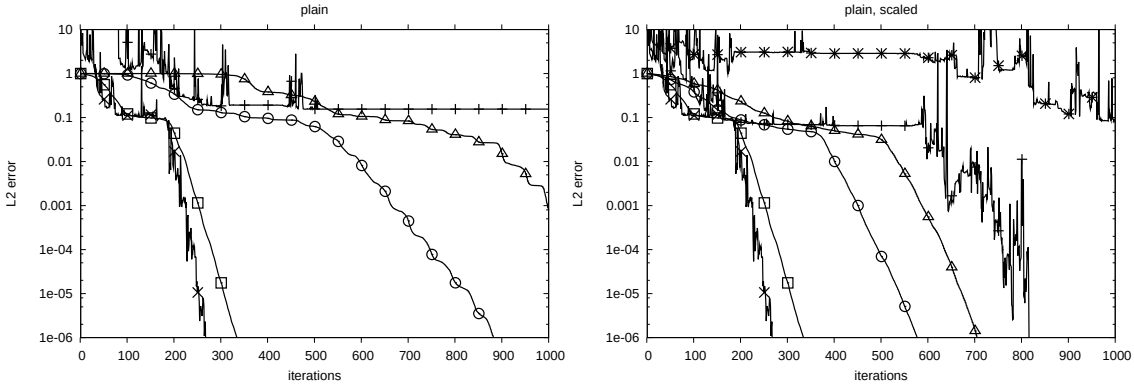


(b) tr-pre

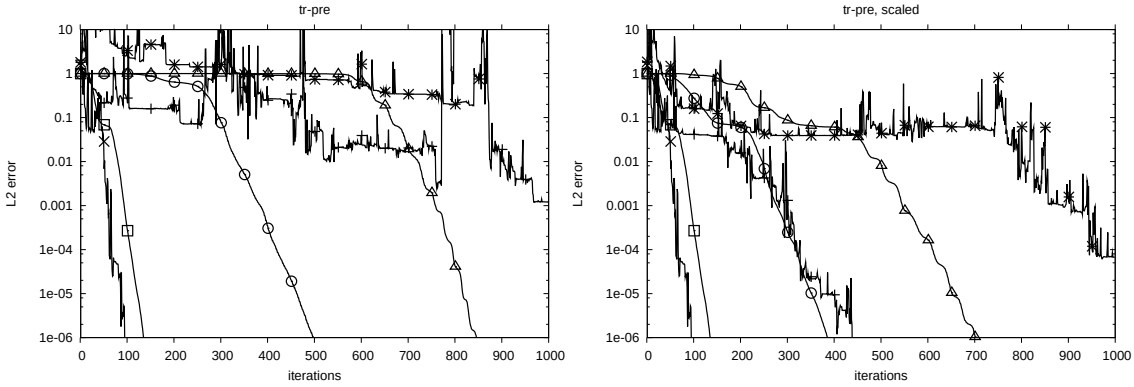


(c) bl-jac

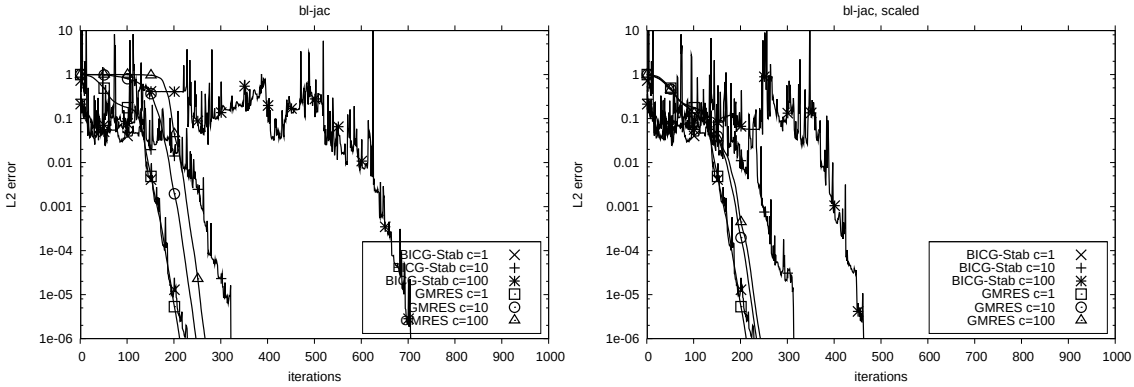
Figure 6.15: Linear convergence of BiCG vs. GMRES and of scaled system on 81 grid, Re $= 10$



(a) plain



(b) tr-pre



(c) bl-jac

Figure 6.16: Linear convergence of BiCG vs. GMRES and of scaled system on 289 grid, Re = 100

6.6. Multigrid method

Algorithm 6.6.1 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	$\mathbf{x}$	$=$	$S_l^{s_2}(\mathbf{x}, \mathbf{b})$

The multigrid algorithm is a powerful solver generally applied for problems showing level-dependent convergence behaviour. The sophisticated tool needs a hierarchy of grids (see Sec. A1) and appropriate grid transfer operators. To complete the round up, we need to use an efficient coarse grid solver and choose an iterative solution method as smoother. As basis we used the *generalized equilibrium formulation* with GMRES linear solver. At first, we give a standard two-grid algorithm in 6.6.1 which is performed with a certain number $s = s_1 + s_2$ of smoothing steps and iterated until the linear defect is sufficiently reduced. In Section 9.3 we present correspondingly linear twogrid contraction rates, mainly for the *driven cavity* testcase. 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, better results were obtained using 'multigrid as preconditioner', i.e. performing one coarse-grid correction gaining several digits of accuracy in every GMRES iteration. The linear contraction rates given in the results section are significantly improved and we perform also numerical comparisons against a Navier-Stokes solver. In Section 9.4 we give CPU times for the monolithic solution of stationary *flow around cylinder* using a multigrid sweep on a mesh hierarchy with W-cycle.

The implementation of grid transfer operators is a main part in obtaining a multigrid solver. While the prolongation is performed elementwise, 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 neighbouring 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 (in 2D), resp., 6 elements in the case of a uniform triangular mesh. Moreover, we will distinguish in the following between the first and second order discretization.

Prolongation

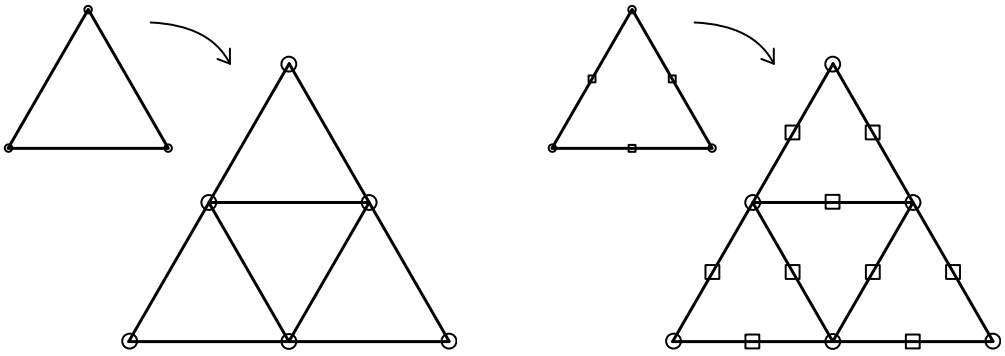
We present the prolongation P_h from a coarse grid G_h (with NEQ nodes) to a refined grid $G_{\frac{h}{2}}$ (with NEQNEW nodes). At first we discuss the upwinding of first order which is more straightforward, because we have degrees of freedom only in the element corners. In general, during one grid refinement step each element in the triangulation is divided into 4 elements. To this purpose, for an element as depicted in Fig. 6.17a we create 3 new nodes in the edge midpoints, which are then connected by new edges. In this way a whole hierarchy of unstructured meshes is generated and any level can be traced back to the starting coarse grid. In the actual prolongation, the solution vector $\mathbf{x}_h$ is interpolated onto the fine grid and the projection $\mathbf{x}_{\frac{h}{2}} = P_h\mathbf{x}_h$ is obtained. The first part

of the prolongation is an identity mapping of the values in the existing, old nodes:

$$x^j_{\frac{h}{2}} = x^j_h \qquad j = 1,\ldots,NEQ$$

The second part is an interpolation of the values in the new nodes from the adjacent, old nodes:

$$x^j_{\frac{h}{2}} = \frac{1}{2}(x^{adj1(j)}_h + x^{adj2(j)}_h) \qquad j = NEQ,\ldots,NEQNEW$$



(a) Refinement of a first order triangular element with corners (circles)

(b) Refinement of a second order element with midpoints (squares)

Figure 6.17: Hierarchical grid refinement

This procedure is somewhat more complicated for the upwinding of second order, mainly due to the additional degrees of freedom in the midpoints. In the refinement step as depicted in Fig. 6.17b, the former corners become corners of 3 new elements which are created by connecting the existing midpoints, which on their part become the corners of the fourth new element. For this inner element are created new midpoints situated *inside* the area of the original element. For the three outer elements, additional midpoints are created on the edges. In the prolongation step one must distinguish between these two kinds of midpoints.
As before, an identity mapping is performed for the existing nodes. The values in the new midpoints which are situated on the edges of the original element, are obtained by quadratic interpolation. Because of the regular refinement the interpolation is always of the form

$$x^j_{\frac{h}{2}} = \frac{1}{8}(3x^{adj1(j)}_h + 6x^{adj2(j)}_h - 1x^{adj3(j)}_h),$$

where $x^{adj1(j)}_h$ and $x^{adj3(j)}_h$ are values in the closer, resp., remote corner and $x^{adj2(j)}_h$ is the value in the adjacent midpoint of the original coarse element. This treatment is second order accurate, to obtain similar accuracy for the 'inside-midpoints', we have to construct a Finite Element like interpolation between the 6 adjacent nodes

$$x^j_{\frac{h}{2}} = \frac{1}{8}(4x^{adj1(j)}_h + 4x^{adj2(j)}_h + 2x^{adj3(j)}_h - 1x^{adj4(j)}_h - 1x^{adj5(j)}_h + 0x^{adj6(j)}_h),$$

with $x^{adj1(j)}_h$ and $x^{adj2(j)}_h$ being values from the adjacent midpoints, resp., $x^{adj3(j)}_h$ from the remote midpoint of the original element. $x^{adj4(j)}_h$ and $x^{adj5(j)}_h$ belong to the remote corners, while the contribution from the closer corner $x^{adj6(j)}_h$ is always zero. This interpolation is valid for arbitrary triangular meshes with regular refinement.

Restriction

The next operator R_h is needed for the restriction of a solution vector $\mathbf{x}_{\frac{h}{2}}$ from a fine grid $G_{\frac{h}{2}}$ to the corresponding coarse grid G_h. In general the restriction is defined as the adjoint operator to the prolongation. The procedure is in practice similar for the two upwindings and can be graphically explained as each node 'collecting' the values it was 'distributing' previously in the prolongation-step. Due to the arbitrary number of adjoining elements for each node, the implementation is a little tricky. Lets assume that the value from x_h^j was used for interpolation to new nodes $y_{\frac{h}{2}}^l,\ l = 1,\ldots,L$ with a weighting of $\frac{1}{2}$. Additionally we had an identity mapping to its corresponding node $x_{\frac{h}{2}}^j$ on the fine grid. Then the restriction is defined as

$$x_h^j = \frac{2x_{\frac{h}{2}}^j + \sum_{l=1}^{L} y_{\frac{h}{2}}^l}{2+L}, \quad j = 1,\ldots,NEQ.$$

Similarly can be obtained a quadratic restriction operator for the second order upwinding, by saving the coefficients and neighbouring nodes used in the prolongation and then dividing the summed up values with an appropriate weight. Another possible treatment is a constant restriction operator which obtains the result by a simple injection. We implemented both schemes and we found that for the second order upwinding a constant restriction gave more stable results so far, while a linear operator was better for the first order discretisation.

6.7. Summary

The linear solution of our fully (time and space) discretised system was, from the beginning, more difficult than solving the nonlinearity. An eigenvalue analysis showed that the conditioning is extremely challenging, especially in the asymptotic limit of small Mach number which is necessary to approximate the solution of the incompressible Navier-Stokes equations. The proposed Krylov-space methods are just basic iterative tools applied in this thesis, without preconditioning they cannot efficiently solve stiff problems ($c \to \infty$) with a large number of unknowns ($h \to 0$). The system obtained from our basic discretisation can be preconditioned by the convection-part of the DVM. The matrix-free implementation of the inverse transport is a central point in this thesis, achieving high efficiency due to a special numbering technique. Alternatively, the condition can be improved using the collision part of the system in a block-Jacobian approach.
The advantages of both variants were combined using the *generalized equilibrium formulation*. The obtained $GEF(\backslash)$ scheme is expected to do well for transport dominated problems due to construction, while the additional collision preconditioning should make it robust and, at least for moderate refinement levels, also fast.
In order to obtain excellent results for large systems and low Mach numbers, we further improved the GEF approach with the application of multigrid as a solver in the spirit of modern numerics. A straightforward way is to use $GEF(\backslash)$ as a robust coarse grid solver and in smoothing steps. But we will show that best results are obtained with a more sophisticated extension of MG methods to the GEF approach. In practice, the use of multigrid as preconditioner in a GMRES iteration to solve the GEF system is an interesting combination which will prove superior to the 'simple' multigrid iteration.

Part III

Numerical Results for Stationary and Nonstationary Flow

7

Validation of the monolithic approach

As already mentioned in the previous Section 6.1, the discretisation we applied to the DVM had to be initially verified. The development of space discretisation techniques can be quite straightforward (see Sec. 4.2). In case of a simple convection equation (4.6), the implementation of a finite difference scheme is followed by verification using analytical (polynomial) functions with easily obtained right hand side and boundary values. Then, using our second order method, the error would naturally behave as $O(h^2)$ compared to the given exact solution.
Unfortunately, we found that obtaining analytical solutions for the DVM itself is practically impossible due to the complicated collision term and not trivial coupling of several distributions on the boundaries. Instead, we will show that our approach to the discrete Boltzmann equation yields (asymptotically) the same results as the Navier-Stokes equations, by taking into account theory presented in Part I of the thesis. First of all, we note that we do not deal directly with the continuous Boltzmann equation (2.1), but with the BGK approximation which introduces a quadratic error in the Mach number. Additional consistent approximations of order $O(\mathrm{Ma}^2)$, as described in the Boltzmann derivation-process in Chapter 2, are made in the DVM for the phase space, which is at last followed by our special discretisations of space and time with main focus on our monolithic approach. Only in this last step we can improve accuracy by using higher order numerical schemes as for the convective term, but the overall accuracy against the NSE is essentially influenced by the Mach number. Unfortunately, the computational effort is increased for low Ma which means a fundamental trade-off between accuracy and efficiency. In practice $\mathrm{Ma} < 0.15$ (see [17]) is a common choice to reduce simulation time while $\mathrm{Ma}^2 \sim 0.02$ is considered sufficiently low. On the other hand, in complex transient flows the arising compressibility can affect the numerical results considerably, as shown in [15], with errors exceeding 10% which is not acceptable in high-accuracy computations.
In this Chapter we will not focus on lowest possible simulation times, but analyse how to choose optimally c depending on h, as discussed in Section 3.3.1. Based on our numerical results we will confirm that, without the correct asymptotical dependence, the error can remain constant while the mesh is successively refined. Conversely, while c is increased keeping h constant, the error is growing with the asymptotic rate of Ma^{-1} (compare [6]). Given our finite difference upwind discretisation of first and second order for the convective term, we follow the assumptions from Sec. 3.3.1 in order to match the discretisation error of order $O(\mathrm{Ma}^{-1}h)$, resp., $O(\mathrm{Ma}^{-1}h^2)$ with compressibility effects of the model in $O(\mathrm{Ma}^2)$. In our monolithic approach we apply for the mesh width h and discretization parameter $c = O(\mathrm{Ma}^{-1})$ the relations

$$\begin{aligned} h &= O((1/c)^3) \quad \text{for first order upwind,} \\ h^2 &= O((1/c)^3) \quad \text{for second order upwind.} \end{aligned}$$

Then, we should achieve optimal convergence of $O(\mathrm{Ma}^2)$ in the asymptotic limit, but we will also try to find the minimal error for every refinement level. In the following, extensive numerical tests

against steady-state solutions (analytical or CFD reference) of the Navier-Stokes equations will be followed by analysing the influence of multiple relaxation times on the results.

7.1. Accuracy and dependence of $c \leftrightarrow h$

For our analysis we calculate the L_2-error on successive levels and start with arbitrary initial h_0 and c_0. In order to conform to the aforementioned asymptotical dependence, we have to keep the relation $c_l^3 \cdot h_l^\gamma$ always fixed, depending on the order γ of the discretisation. Practically, with h being reduced by a factor of 2 in every refinement step, the simulation parameter c has to be accordingly increased by a factor of $2^{1/3}$ for first order upwinding, resp., $2^{2/3}$ for second order upwinding. In Fig. 7.1 we plot the theoretical slope of quadratic convergence, the points in the slope are indicating the distance for a refinement step. Against the theoretical numbers we enter the relative error computed for three different CFD testcases: *Driven cavity*, *rotating couette flow* and *flow around cylinder* (see Appendix A1). The results show a good agreement with the theory, neglecting the initial results as we are interested mainly in the asymptotic behaviour. Compared to the 1st order upwinding, the 2nd order discretisation obviously gives a smaller error and needs less grid refinement steps, at the cost of bigger sound parameter c. We can extend the asymptotical analysis to higher refinement levels using powerful linear solvers.
Although the asymptotic behaviour was confirmed, it is interesting to see if we can distinguish between modeling and discretisation error and get more information about the separate contributions to the overall error. With this information we could find a good starting guess for c_0 and obtain optimal results on all successive levels. That is why we continued with the numerical analysis, now varying c for fixed mesh width h. The plots (see Fig. 7.2) show the $L2$-errors for different testcases and we can effectively observe on each level the existence of a (unique) optimal choice, say a c_{opt}. Using a finer mesh, this optimal point is shifting to the right, so $c_{opt} = c_{opt}(h)$ is obviously level-dependent. But a fundamental observation is that in the parameter range beyond c_{opt}, the error is again increasing continuously with the rate of $O(c) = O(\mathrm{Ma}^{-1})$. This confirms our assumptions regarding the discretisation error, and the obtained numerical results are basically similar to the plots in Figure 3.2 which showed simplified and purely theoretical graphs. A special observation is made for the *rotating couette flow* which has given analytical solutions *independent* of the viscosity ν. Consequently, the approximation only depends on the overall value of $\tau = \frac{3\nu}{c^2}$. Comparing Figures 7.2a – 7.2b it is clear that any alteration in ν can be *exactly* compensated by a shift in c, resulting in a similar solution.
This observation is transferable to problems which are dependent on the Reynolds number. Although there we cannot arbitrarily exchange shifts in ν and c, nevertheless it follows that very low values of viscosity should be solved with smaller parameter c, like for example in the case of the flow around cylinder benchmark (see Fig. 7.2c). Similar results were obtained for the *driven cavity* testcase at varying Reynolds numbers, see the plots in Figure 7.3. Obviously, a higher Reynolds number results in a slightly larger L_2-error comparing similar refinement levels. Apart from that, c_{opt} is getting smaller for increasing Re, so it shifts together with the viscosity ν of the simulated medium. This eventually faces us two problems, one is that for both, increasing Re and smaller c, the nonlinearity is more difficult to solve (see Sec. 9.1). The other is the increased compressibility error for larger Mach numbers. As remedy it is proposed to use multiple-relaxation-time models (MRT, see [8], [9], [10], [28]) which are subject of our current research (see results in Sec. 7.2).
So far we could derive a rule of thumb for the choice of c in each testcase, which is also relevant for the linear and nonlinear solvers. 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.

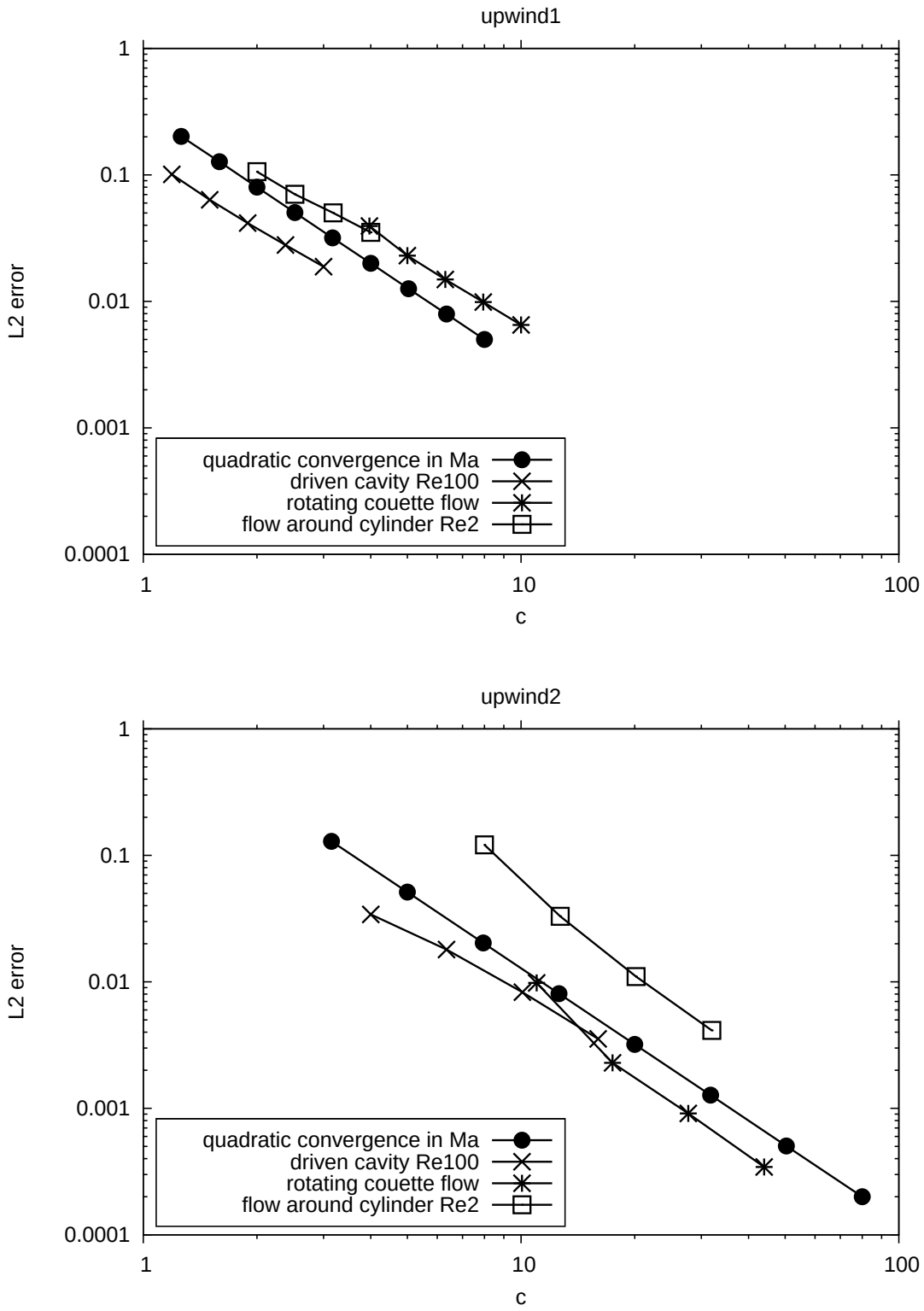
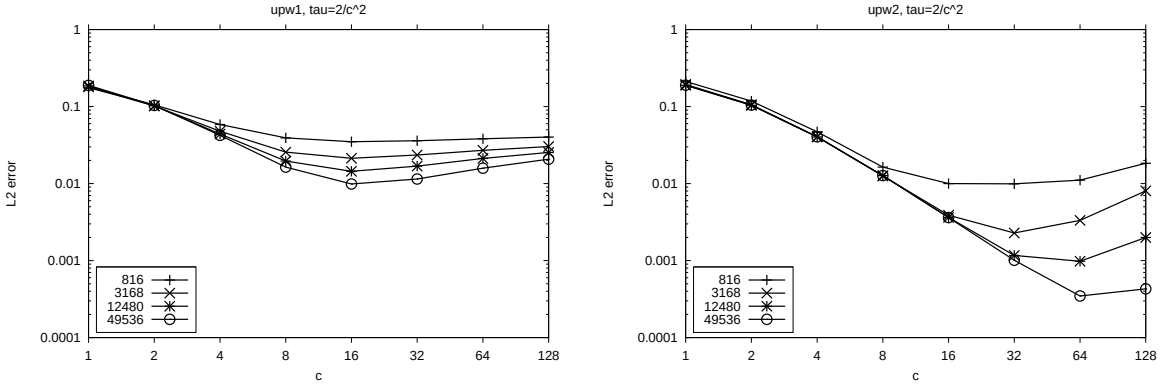
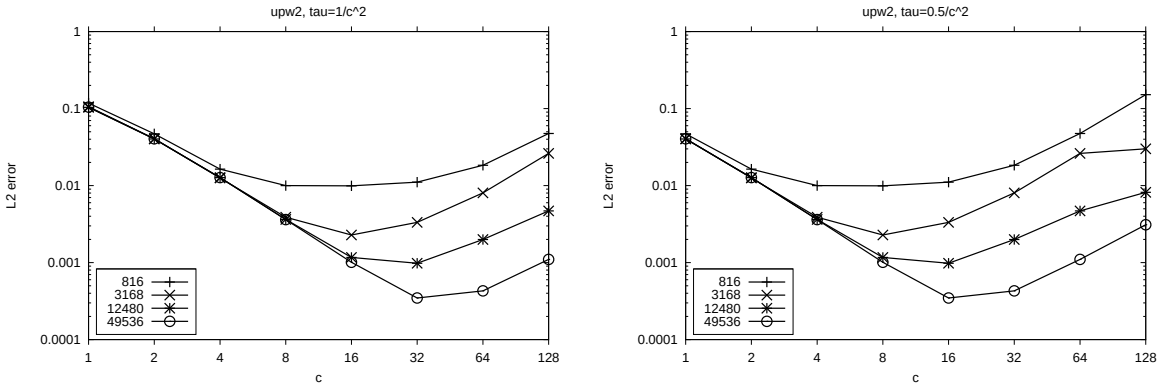


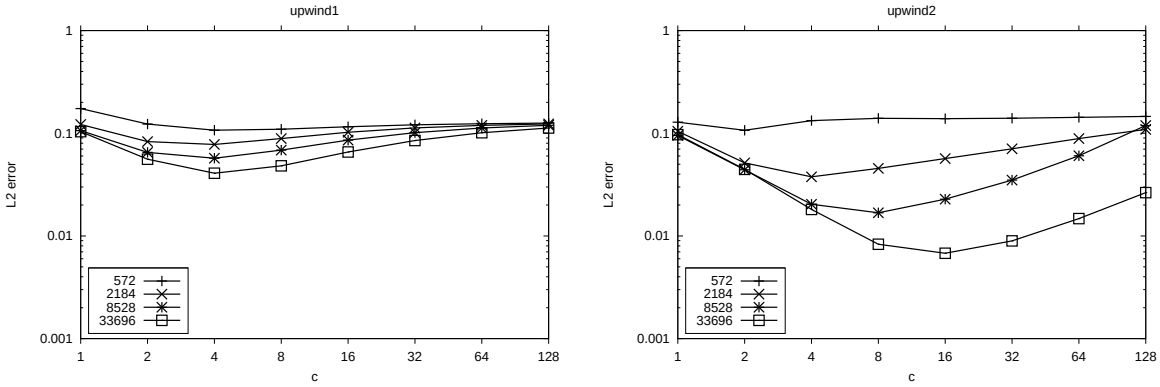
Figure 7.1: Comparison of L_2-error vs. theoretical (quadratic) convergence with fixed relation $c^3 \cdot h^\gamma$



(a) *Rotating couette flow*, L_2-error for 1st and 2nd order upwind



(b) *Rotating couette flow*, for 2nd order upwind with changed τ



(c) *Flow around cylinder* at $\mathrm{Re} = 2$, L_2-error for 1st and 2nd order upwind

Figure 7.2: Results for accuracy of the D2Q9 model on unstructured meshes

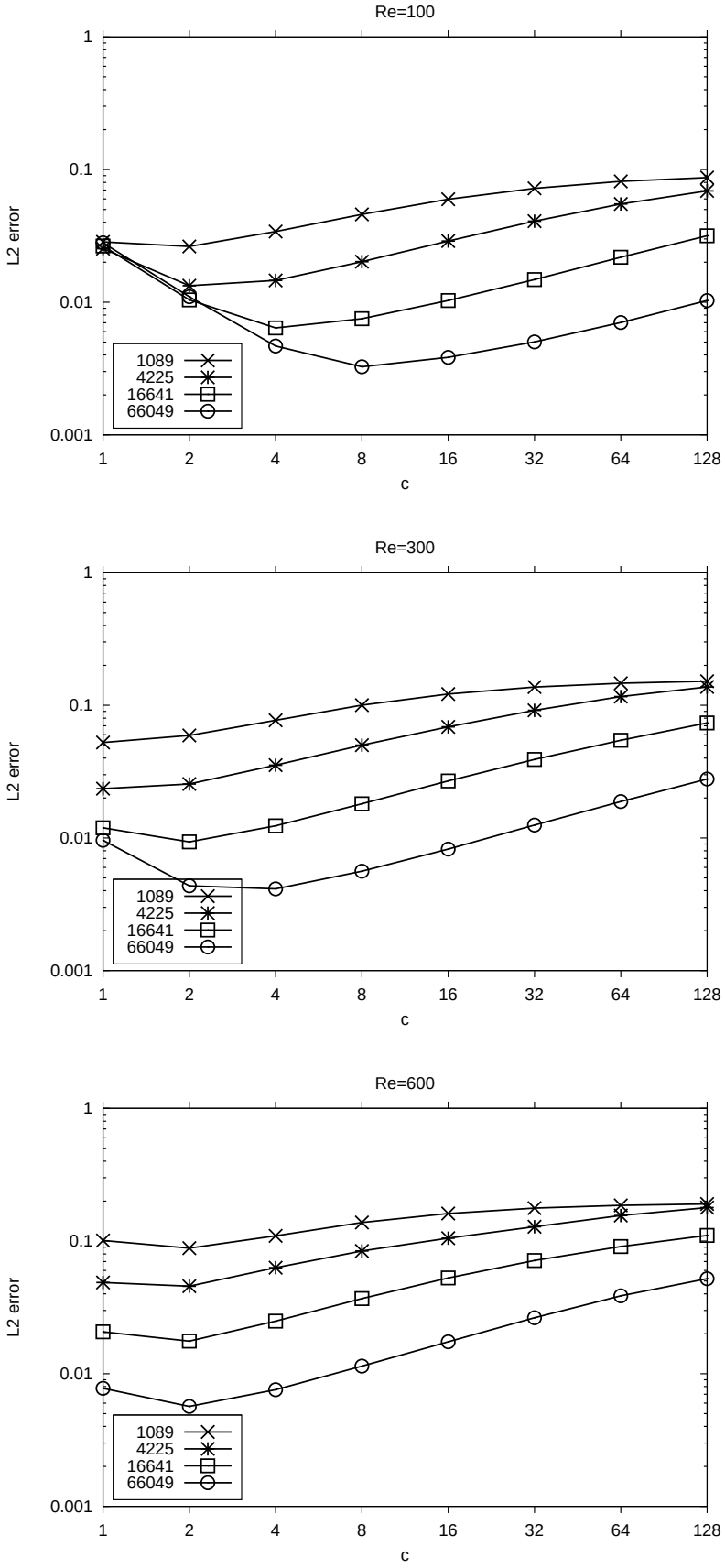
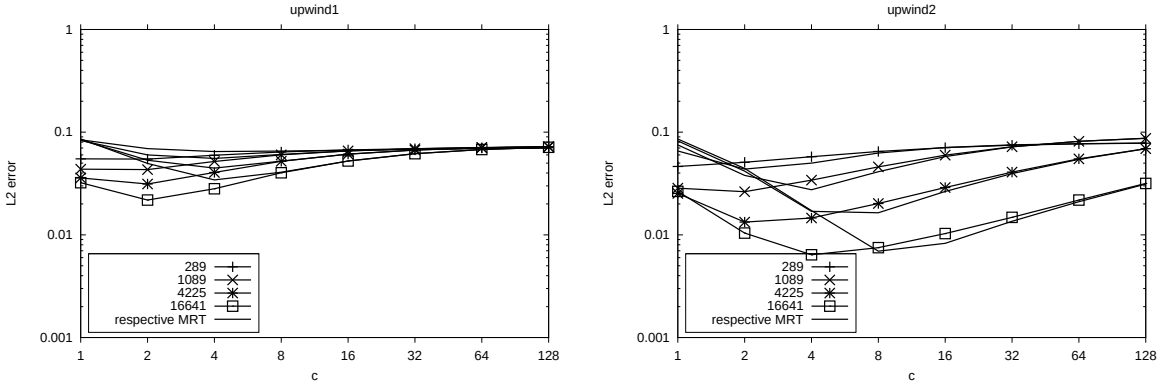


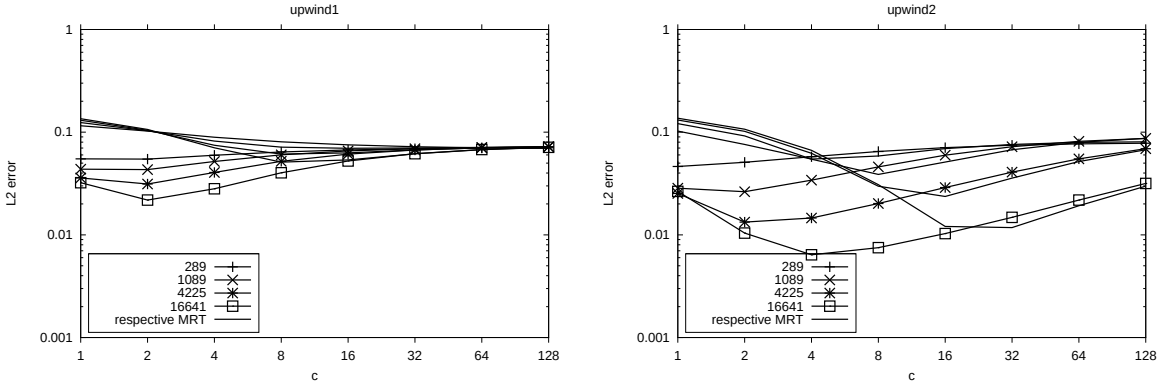
Figure 7.3: *Driven cavity* with 2nd order upwind: error dependence on Re number, Re $\in \{100, 300, 600\}$

7.2. MRT results

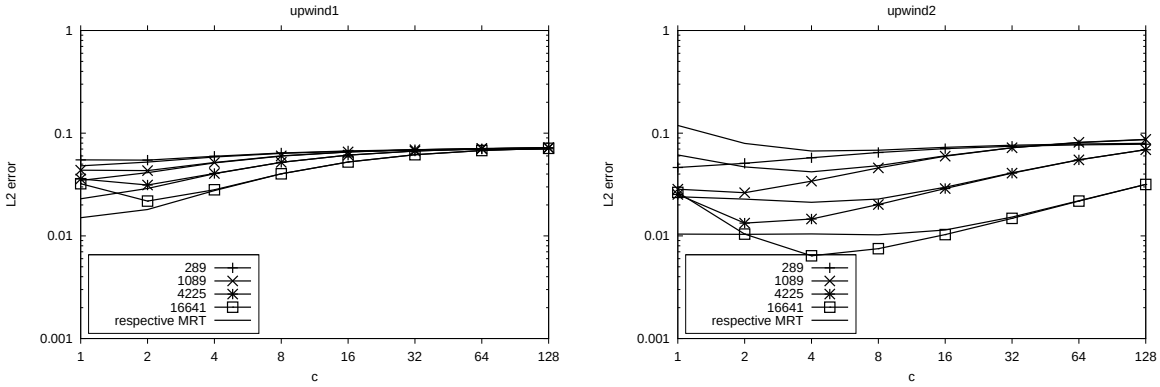
The first numerical tests using multiple relaxation times for a stationary *driven cavity* configuration were intended to verify convergence of the MRT model against Navier Stokes references. The relaxation rates $s_7,\ s_8$ (see Section 2.5) were chosen to keep a fixed Reynolds number of $\mathrm{Re} = 100$ for the whole range of parameters c, while the relaxation rates $s_1,\ s_2,\ s_4$ could be freely adjusted. In literature concerning the MRT model for the LBM these are usually fixed around 1, see [28] for reasons of stability. We should keep in mind that the LBM is a special time-discretisation which introduces a time-stepping with small Δt included in a modified relaxation time and that we cannot transfer the results directly to our monolithic model. In our case, similar treatment gave unphysical results and the solver did not converge. Consequently, we chose the free relaxation rates smaller than the physical relaxation time by a factor of 10, resp., 100. Effectively, the conditioning of the system was slightly improved due to the reduced rates. To evaluate the accuracy we present the obtained L_2 error in Figs. 7.4a, 7.4b in comparison to the previous SRT results.
It is obvious that for small c a reduction of the relaxation time is giving worse results, it seems that the Mach-error is even increased. The results for second order upwind show that the accuracy is increased in the range of $c > c_{opt}$ but asymptotically merges with the graphs for the SRT model.
Moreover, we tested further variations of the MRT-model, but this time increasing the free relaxation rates by a factor of 10 (see Fig. 7.4c). Especially the accuracy of the first order upwind was improved, obtaining good results for $c = 1$, even improving the respective c_{opt}-result of the SRT model. The optimal results of second order upwind were not so much improved, but the valley in the graph became wider, which means that very good results were obtained with small simulation parameter c.
We attempt a final interpretation of the results based on Sec. 3.3.1. The discretization-error, which is scaled by the paramter c (see Eq. 3.14) and is dominating the right side of plots 7.4, seems not to be affected by the MRT model. In contrast, the compressibility-error on the left side of our plots is directly influenced by reduced, resp., increased relaxation rates. However, we did not obtain any strict rule which we could use, for example to finally optimize results of the nonstationary *flow around cylinder* benchmark at $\mathrm{Re} = 100$ (see Sec. 8.1). Further investigation of the MRT model in combination with our implicit approach is planned in future work, but it is not clear yet if the application of MRT is a strategy specifically for the stability of the LBM or concerns the discrete Boltzmann equation in general.



(a) MRT with reduced relaxation rate $0.1/\tau$ against SRT



(b) MRT with reduced relaxation rate $0.01/\tau$ against SRT



(c) MRT with *increased* relaxation rate $10/\tau$ against SRT

Figure 7.4: Comparison of multiple-relaxation-time simulations against the SRT L_2-error, each corresponding MRT graph is converging against the SRT result for $c \to \infty$

7.3. Stationary flow around cylider

After mainly comparing the L_2 error for the different testcases, we continue the tests for the *flow around cylinder* benchmark at $\mathrm{Re} = 20$ (see plots for the pressure and x-component of the velocity field in Fig. A1.6). Here, the choice of parameter c with optimal accuracy is presumably lower than in the channel flow at $\mathrm{Re} = 2$ (see Fig. 7.2c), where the viscosity was set to $\nu = 0.01$, which is comparably easy and leads, as mentioned before, to higher c_{opt}. The $\mathrm{Re} = 20$ case has quite low viscosity of $\nu = 0.001$ so that the crucial c will range close to 1, increasing for higher refinement levels. The benchmark results focus on the forces acting on the cylinder by computing the line integral of the overall stress, crucial reference values from ([39]) for drag and lift are 5.5795 and 0.0106, respectively, and $\Delta p = 0.1175$ for the pressure difference between front and back of the circle. In the LBM, the stress is usually obtained by evaluating the nonequilibrium distributions $f_i^{neq} = f_i - f_i^{eq}$ (see Sec. A2.1). An FEM-like method would evaluate the non-primary variables $\mathbf{u}$ resulting from summation of the primary distributions f_i, but impending loss of accuracy due to differentiation and cancellation of opposing distributions in the term $\sum \mathbf{e}_i f_i$ (see [33]) is expected.
By a first look at the results in Table 7.1, the FEM like evaluation based on pointvalues of p and the gradient of $\mathbf{u}$ gives very few results within 1% of the reference drag (the range being $5.524 - 5.635$), although it is difficult to discern a convergence of the values with c or h. The corresponding 1%-range for lift being $0.0105 - 0.0107$, good results there are, in general, even more difficultly obtained. However, the LBM method yields better results for every configuration, although the values for the grid with 1092 points are insufficient. A higher refinement level and small to moderate c gives a drag within 1% of the reference, also the lift is accurately reproduced. Looking at the obtained values for the pressure difference, Δp is getting closer to the reference with mesh refinement, but higher values of c give worse results again. It seems that the parameter c is significantly affecting the solution in boundary nodes, also taking into account that the second order upwinding is not positivity preserving and can oscillate in areas of strong gradients. Alternatively, we tried to obtain improved pressure difference by extrapolation of the values from the interior flow domain, instead of taking the actual solution from the boundary. The extrapolated values for Δp given in the rightmost column are less extreme and the results are significantly improved.
Finally, in order to improve the overall results and showing the possibilities of grid-adaptivity, we constructed a highly adapted grid with a large concentration of points around the cylinder which was accomplished without difficulty using triangular elements (see Fig. A1.3). With the new mesh, we obtain good results for drag and lift, and significantly improve the results for Δp even with fewer number of unknowns.
However, the improvement of Δp due to extrapolation for the highly adapted grid is marginal, because both strategies smooth out high jumps in the solution, either by avoiding direct evaluation of the wall pressure or by a higher resolution in the critical area.

	FEM force eval.		LBM force eval.			
nodes	drag	lift	drag	lift	Δp	Δp extrap.
Results on standard benchmark grid						
			c=1			
1092	6.3801	0.0280	6.3293	0.0270	0.1766	0.1326
4264	5.8841	0.0419	5.6676	0.0413	0.1418	0.1161
16848	5.9149	0.0107	5.5287	0.0121	0.1243	0.1125
66976	6.3263	0.0071	5.5398	0.0102	0.1196	0.1128
			c=2			
1092	6.3759	0.0519	6.3800	0.0507	0.1827	0.1263
4264	5.6531	0.0451	5.6490	0.0448	0.1546	0.1139
16848	5.5755	0.0120	5.5403	0.0123	0.1270	0.1122
66976	5.8075	0.0090	5.5863	0.0103	0.1205	0.1145
			c=4			
1092	7.9013	0.1660	7.9062	0.1642	0.2269	0.1463
4264	5.4817	0.0579	5.5508	0.0594	0.1744	0.1068
16848	5.3840	0.0143	5.4923	0.0143	0.1316	0.1097
66976	5.5902	0.0100	5.6014	0.0106	0.1219	0.1148
Results on highly adapted grid						
			c=1			
762	4.4639	0.0494	4.3681	0.0513	0.1409	0.1206
2948	5.9034	0.0451	5.6422	0.0466	0.1240	0.1140
11592	5.9151	0.0209	5.5191	0.0226	0.1188	0.1124
45968	6.1069	0.0105	5.5228	0.0128	0.1177	0.1127
183072	6.3378	0.0075	5.5284	0.0105	0.1174	0.1130
			c=2			
762	4.9233	0.1439	4.9295	0.1450	0.1596	0.1316
2948	6.0262	0.0937	5.9606	0.0938	0.1314	0.1201
11592	5.7718	0.0310	5.6453	0.0315	0.1209	0.1158
45968	5.7835	0.0134	5.5801	0.0143	0.1184	0.1154
183072	5.8641	0.0097	5.5695	0.0109	0.1178	0.1156
			c=4			
762	5.4634	0.1463	5.4949	0.1472	0.1830	0.1412
2948	6.2414	0.1266	6.2388	0.1267	0.1398	0.1243
11592	5.7711	0.0404	5.7388	0.0405	0.1231	0.1178
45968	5.6735	0.0160	5.6080	0.0163	0.1188	0.1166
183072	5.6857	0.0110	5.5812	0.0114	0.1178	0.1166

Table 7.1: Evaluation of drag and lift coefficients, resp., pressure difference (with extrapolation) on standard and highly adapted grid, references: drag= 5.5795, lift= 0.0106, $\Delta p = 0.1175$

8

Extended results for nonstationary flow

8.1. Time/space accuracy for nonstationary flow

Finally, the nonstationary *flow around cylinder* benchmark was used to verify the accuracy of the applied time-discretisation schemes, starting with the basic implicit Euler time-stepping (4.3) and expecting improved results with the second order Crank-Nicholson scheme (4.4). The maximum inflow velocity is increased to $u_0 = 1.5$ with the same viscosity $\nu = 0.001$ as before which results in $\mathrm{Re} = 100$ and a transient flow regime.
This benchmark configuration is thoroughly analysed by many important CFD codes. It shows strong vortices forming behind the cylinder and the flow is oscillating with a specific frequency f which defines the Strouhal number $St = \frac{Df}{U}$ with reference $St \sim 0.300$ (see [39]). With the given period length $T = 1/f$ the lift value shifts above and below zero with an amplitude of about 2 while the drag and Δp exhibit a double period for the same length. The most recent simulations performed by FEATFLOW produced improved reference values compared to [39], giving for the lift $C_{Lmin} = -1.023$ and $C_{Lmax} = 0.987$, resp., for the drag $C_{Dmin} = 3.168$, $C_{Dmax} = 3.230$ which are supposed to be correct for at least 2 digits. The aim, besides accurately reproducing this widely accepted CFD benchmark, was to analyse the combined space- and time-discretisation in view of accuracy and stability of the low and high order schemes. By obtaining good results using large Δt, our implicit approach was supposed to distinguish itself from standard Lattice Boltzmann methods which have to use micro time-steps for this challenging problem.
We omit the results for first order upwind, as the simulated flow remained stationary. Only on the refined grid with 66 976 (denoted 66K) nodes a very small oscillation evolves inside the channel. In contrast, the second order upwinding clearly showed vortices developing behind the cylinder even on a low refinement level with 4 264 (denoted 4K) or 16 848 (16K) grid-points, where it was possible to obtain nice pictures and movies at low computational cost. We show a sequence of pictures representing one full period in Fig. 8.1, obtained on a refined grid with 66K nodes and for high choice of $c = 20$ which, as we will see, gives highly accurate results. We present the x-component of velocity on the left and a 3D view of the pressure on the right, starting at $t_0 = 0$ where the lift reaches the peak C_{Lmax} and continuing with snapshots every $0.055s$ which exactly divides the period length $T = 0.330$ into 6 parts. The pictures are presented in grayscale with 10 contour levels. The bright shades of grey mean high velocity (up to 2.15), while the darker shades show descreasing values reaching even below -0.5 for the black contour level. This means we have a main vortex just behind the cylinder which appears in the pictures as an oscillating 'black tail'. In the first step, the tail is just in the middle of the downswing, in its wake we see develop a patch of low pressure, a similar depression is forming in the third picture below the tail which is then on an upswing. The depressions are then transported down the channel, along with the vortices of the velocity $\mathbf{u}$. Obviously, in front of the cylinder we have a large pressure gradient, the pressure difference Δp between the extreme points of the object is oscillating around 2.5 which is significantly larger than for the stationary benchmark.

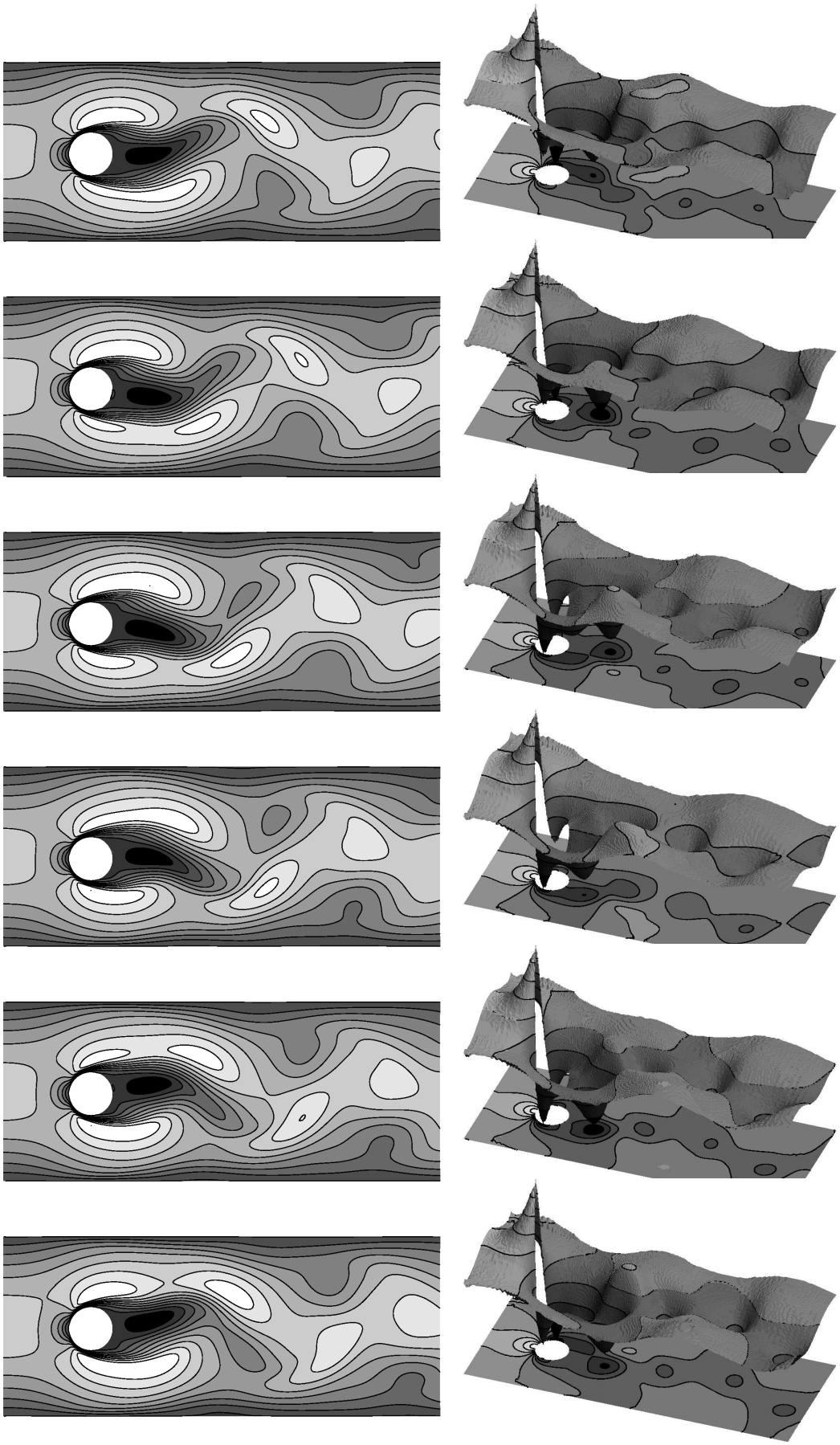


Figure 8.1: *Flow around cylinder* visualization of u_1 and p representing one full period of length $T = 0.330$ on first part of channel ($x \in [0,1]$), snapshots taken from times $t_0 = 0$ (C_{Lmax}), $t_1 = 0.055$, $t_2 = 0.11$, $t_3 = 0.165$ (C_{Lmin}), $t_4 = 0.22$, $t_5 = 0.275$, 66K grid with $c = 20$, $\Delta t = 0.001$

However, pictures give only a slack understanding of scientific results, so have a strict look at the tabulated numbers for maximum lift and drag and Strouhal number obtained on the standard grid (see Fig. A1.2). The low time accuracy of the implicit Euler method becomes obvious in Table 8.1. Looking at the lift on two refinement levels with 4K and 16K grid points, a time-step of $\Delta t = 10^{-2}$ gives values far away from the reference. For $\Delta t = 10^{-3}$ the amplitude (in general $lift_{min} \sim -lift_{max}$) is improved. However, we see more changes in the numbers for the last $\Delta t = 10^{-4}$, so a smaller time-step still would be needed using this first order scheme. Amazingly, the results obtained by the Crank-Nicholson time-stepping at $\Delta t = 10^{-2}$ (see Table 8.2) are comparable to that of the first order scheme with time-step two orders of magnitude smaller. It means a similar accuracy with 1% of the number of timesteps, also the results of the second order scheme do not change significantly for $\Delta t < 10^{-2}$ which means a small error due to the time-discretisation. Having ensured a good time-accuracy, let us evaluate the convergence with reference to reduced grid-spacing and Mach number. We will again try to distinguish between the compressibility-error $O(\mathrm{Ma}^2)$ vanishing for large c and the discretisation-error $O(\mathrm{Ma}^{-1}h^\gamma)$ which is amplified by the simulation parameter c. Increasing the refinement level up to 66K grid points, the results in general are improving, but show a strong dependence on the choice of c. Therefore, let us have a look at the plots 8.2–8.5 which show lift, drag and Δp for about one period-length. From these plots and Tables 8.2 and 8.3 we collect the following observations:

- maximum lift is above the reference for some c from the discrete parameter set, the amplitude decreases after a turning point and is too low for large c
- maximum drag behaves similar to $lift_{max}$ (here reference oscillates between 3.2 ± 0.03) the absolute values get too low for large c but the amplitude improves
- Δp is double periodic like drag (reference oscillates between 2.45 ± 0.04) with amplitude better for small Mach number, but gradually shifted above reference with increasing c (but here again extrapolation gives very good results)
- St is continuously improving with c but in the end it is above the reference 0.300 which means the period is too short, the exact value is usually not estimable for large Δt

We can deduct that the compressibility-error obviously dominates for small c, $c = 2$ for example means a Mach number of $\mathrm{Ma} = \frac{u_0}{c_s} \sim 1$. The resulting lift/drag are below the references for a range of small c up to a certain turning point, then they decrease again, so we do not have clear information as from the L_2-error plots for stationary problems. Moreover, we cannot look at h and c as decoupled in our numerical analysis: For certain values of c depending on the refinement level ($c = 4$ on the 4K grid, $c = 8$ on 16K grid, etc.) the maximum lift is larger than the reference and smaller for the following c. Based on the assumptions from Sec. 3.3.1, we suppose that after compressibility effects are sufficently reduced, mainly the discretization-error becomes dominant. Due to the two opposing error-terms of order $O(\mathrm{Ma}^2)$ and $O(\mathrm{Ma}^{-1}h^\gamma)$, it is feasible to choose c rather too large than too small so that the *quadratic* Mach error is surely avoided. Focusing on the highest refinement level and range around $c = 20$ (accordingly $\mathrm{Ma}^2 = 0.0075$), we obtained very accurate results as seen in Table 8.3. Remarkably, the reference values were approximated with similar accuracy for large time-step size up to $\Delta t = 1/100$ without any stability problems. In these simulations the fastest solver was a BiCG-Stab iteration of the GEF approach *without* additional preconditioning, showing the good performance of our special transport solver for nonstationary problems.

Finally, we try to apply strategies to further improve results, based on the techniques introduced

previously in this thesis. The values of Δp are easily and significantly improved by extrapolation of the pressure on the cylinder (see Fig. 8.5), but this does not affect the stress tensor as already discussed for the stationary results. In the LBM community the *flow around cylinder* benchmark is frequently approached using the MRT model (see Sec. 2.5). Accordingly, we took the developed flow obtained by the SRT model, to be precise, only the saved results for $c \geq 8$, and continued with a reduced relaxation time. The restarted simulation with changed relaxation-rates is not forcibly disturbed as for example when suddenly altering the system's speed of sound, instead, the modified MRT rates take effect steadily, yielding different results for the 'developed' flow (see Table 8.4). There is no significant change in the *St* numbers, which were already good. The lift values are slightly increased and thereby improved. The drag is most visibly improved, the divergence due to large c was obviously antagonized by shifting the values upwards, close to the reference (see plots 8.6). Also results for Δp are improved by the MRT model (see plots 8.7). A feasible application of the MRT model would be therefore, to take a developed flow and increase, resp., reduce the free relaxation rates, that is if one can guess if the Mach-error should be further decreased, or rather the conditioning improved. In any case, the flow adapts quickly to the new values, while changing the viscous rates has grave effects on the result and requires a complete restart.

	4K grid			16K grid		
c	C_{Lmax}	C_{Dmax}	St	C_{Lmax}	C_{Dmax}	St
	$\Delta t = 1/100$			$\Delta t = 1/100$		
2	0.42	2.80	0.1887	0.57	3.03	0.2041
4	0.63	2.92	0.2381	0.55	3.08	0.2500
8	0.37	2.86	0.2564	0.42	3.01	0.2778
16	0.15	2.84	0.2632	0.29	2.99	0.2941
	$\Delta t = 1/1000$			$\Delta t = 1/1000$		
2	0.63	2.89	0.1961	0.83	3.18	0.2128
4	1.06	3.14	0.2500	0.89	3.33	0.2632
8	0.80	2.99	0.2778	1.04	3.21	0.2857
16	0.62	2.92	0.2703	0.85	3.14	0.2941
	$\Delta t = 1/10000$			$\Delta t = 1/10000$		
2	0.67	2.89	0.1991	0.84	3.21	0.2152
4	1.09	3.17	0.2515	0.90	3.36	0.2670
8	0.85	3.00	0.2777	1.09	3.25	0.2917
16	0.67	2.93	0.2761	0.92	3.17	0.3012

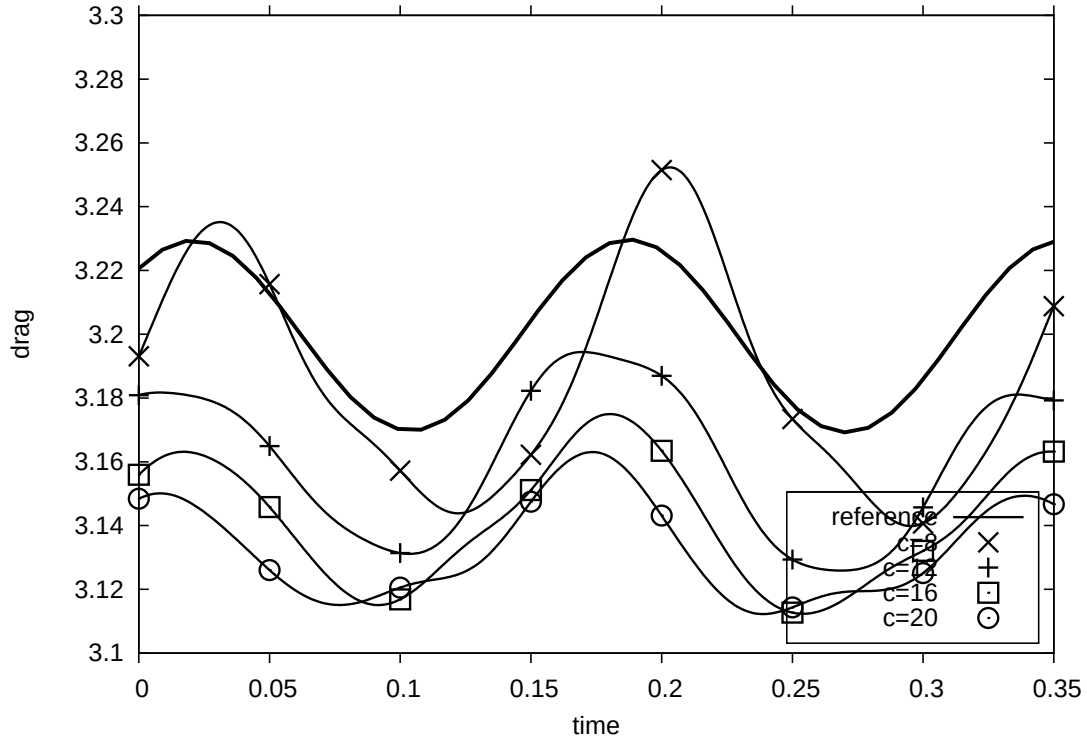
Table 8.1: Benchmark results: Implicit Euler, references: $C_{Lmax} = 0.987$, $C_{Dmax} = 3.230$, $St = 0.300$

	4K gridpoints			16K gridpoints		
c	C_{Lmax}	C_{Dmax}	St	C_{Lmax}	C_{Dmax}	St
	$\Delta t = 1/100$			$\Delta t = 1/100$		
2	0.66	2.90	0.2041	0.85	3.21	0.2174
4	1.08	3.19	0.2500	0.92	3.37	0.2632
8	0.84	3.01	0.2778	1.10	3.24	0.2941
16	0.68	2.93	0.2703	0.90	3.16	0.3030
	$\Delta t = 1/1000$			$\Delta t = 1/1000$		
2	0.67	2.89	0.2000	0.85	3.22	0.2151
4	1.11	3.19	0.2519	0.91	3.37	0.2674
8	0.85	3.01	0.2786	1.11	3.25	0.2915
16	0.68	2.93	0.2762	0.93	3.18	0.3012
	$\Delta t = 1/10000$			$\Delta t = 1/10000$		
2	0.67	2.89	0.1992	0.84	3.21	0.2155
4	1.10	3.17	0.2516	0.90	3.37	0.2672
8	0.85	3.01	0.2779	1.10	3.26	0.2920
16	0.68	2.94	0.2762	0.93	3.18	0.3014

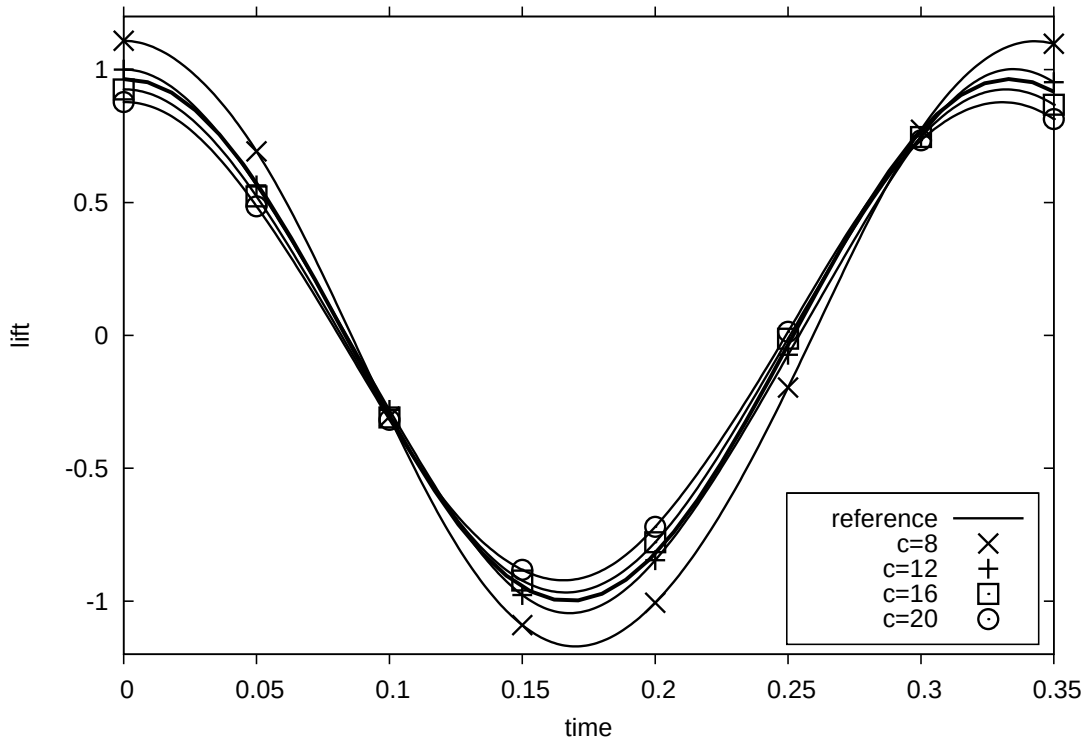
Table 8.2: Benchmark results: Crank-Nicholson, references: $C_{Lmax} = 0.987$, $C_{Dmax} = 3.230$, $St = 0.300$

	66K gridpoints				
c	C_{Lmin}	C_{Lmax}	C_{Dmin}	C_{Dmax}	St
		$\Delta t = 1/1000$			
4	-0.925	0.908	3.082	3.373	0.2710
8	-1.211	1.165	3.163	3.278	0.2924
12	-1.117	1.081	3.138	3.228	0.2985
16	-1.054	1.019	3.119	3.202	0.3003
20	-1.027	0.992	3.110	3.189	0.3021
24	-0.998	0.968	3.104	3.170	0.3021
		$\Delta t = 1/200$			
16	-1.055	1.022	3.119	3.203	0.3030
20	-1.029	0.994	3.111	3.188	0.3030
24	-1.004	0.973	3.105	3.170	0.3030
		$\Delta t = 1/100$			
16	-1.054	1.016	3.117	3.200	0.3030
20	-1.028	0.994	3.109	3.187	0.3030
24	-1.010	0.979	3.106	3.171	0.3030

Table 8.3: Benchmark results: Crank-Nicholson, high refinement level, references: $C_{Lmin} = -1.023$, $C_{Lmax} = 0.987$, $C_{Dmin} = 3.168$, $C_{Dmax} = 3.230$, $St = 0.300$

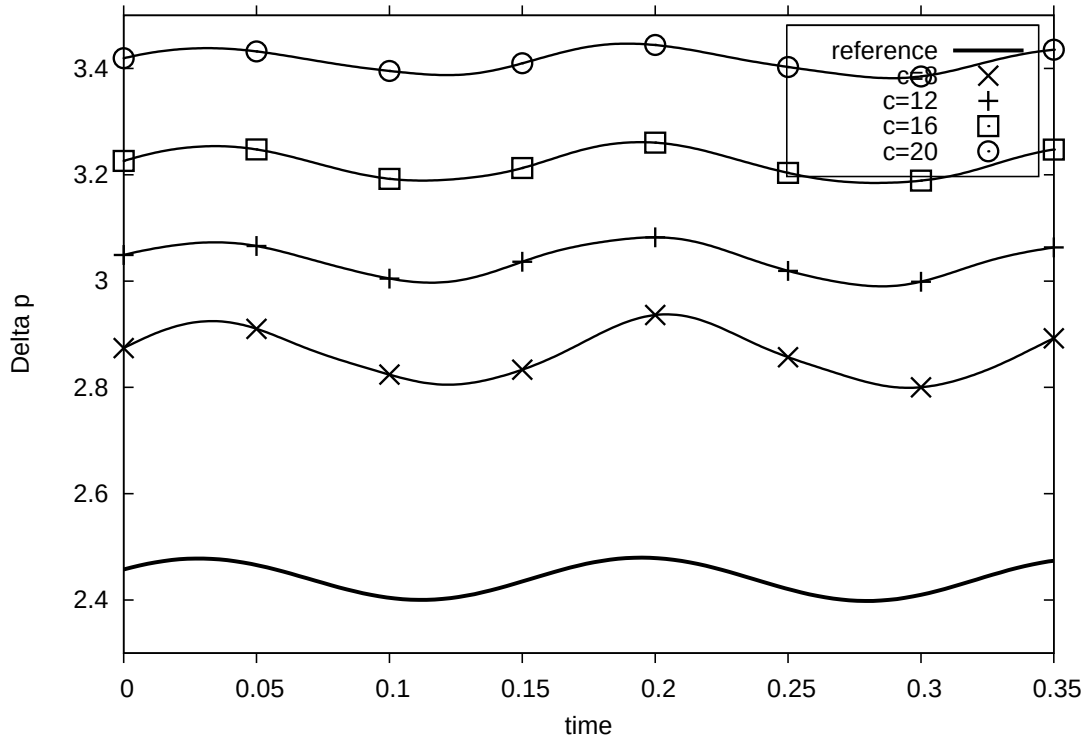


(a) drag

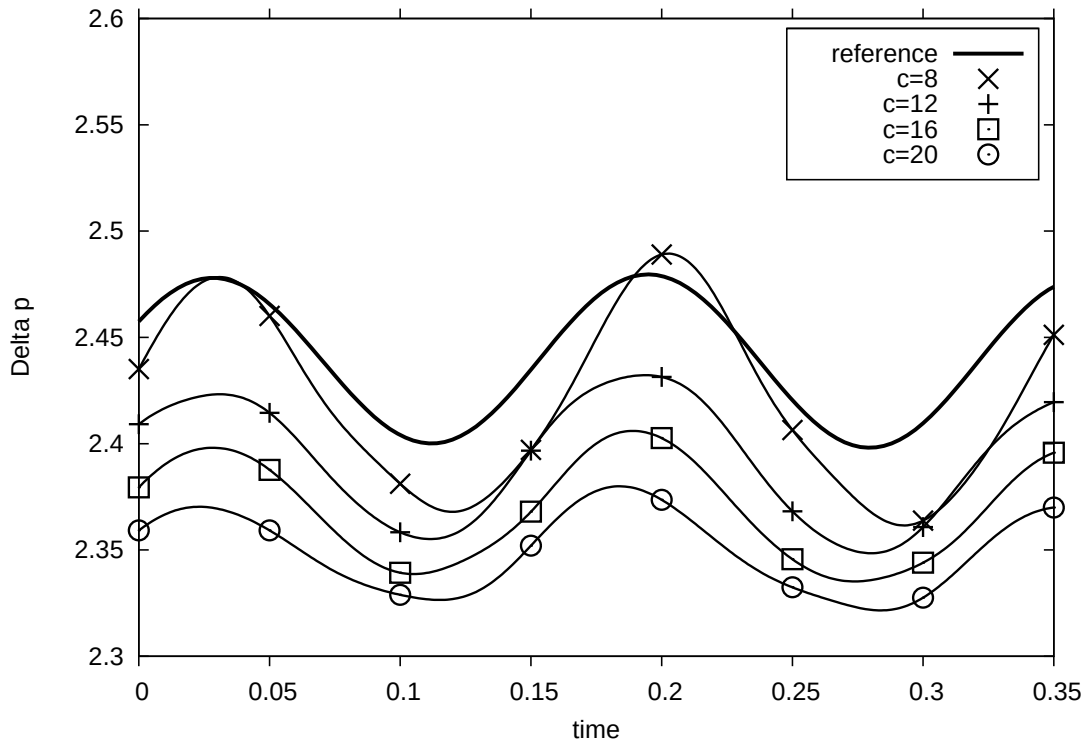


(b) lift

Figure 8.2: Forces on cylinder for various c, 16k grid

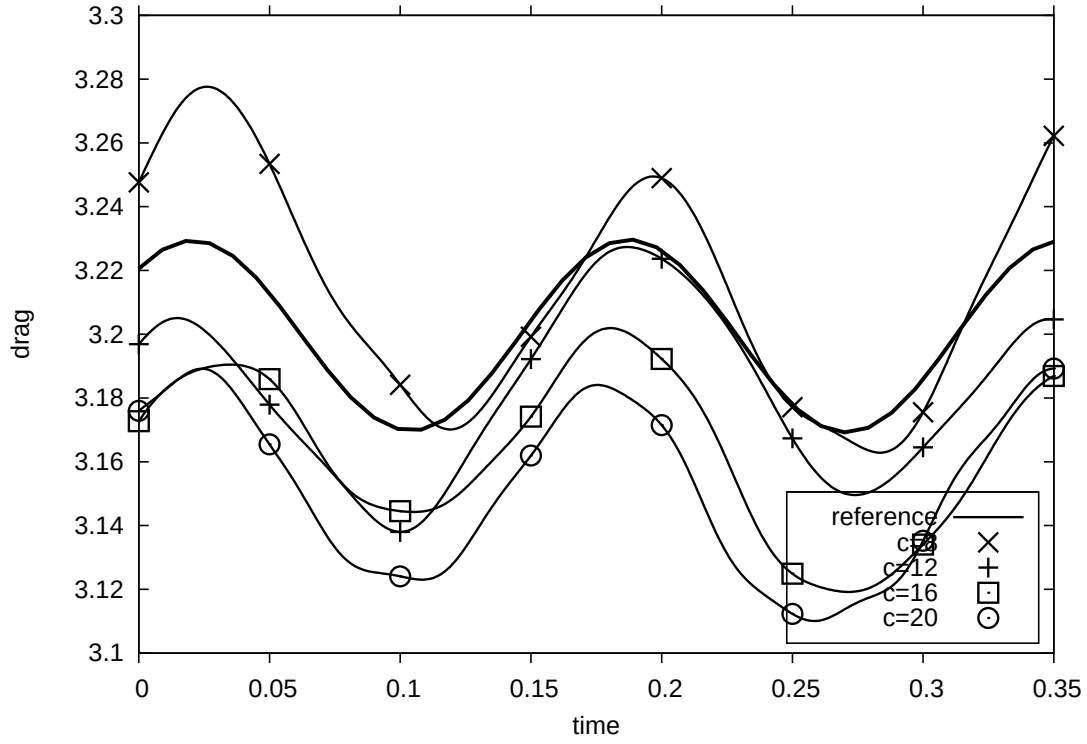


(a) Δp

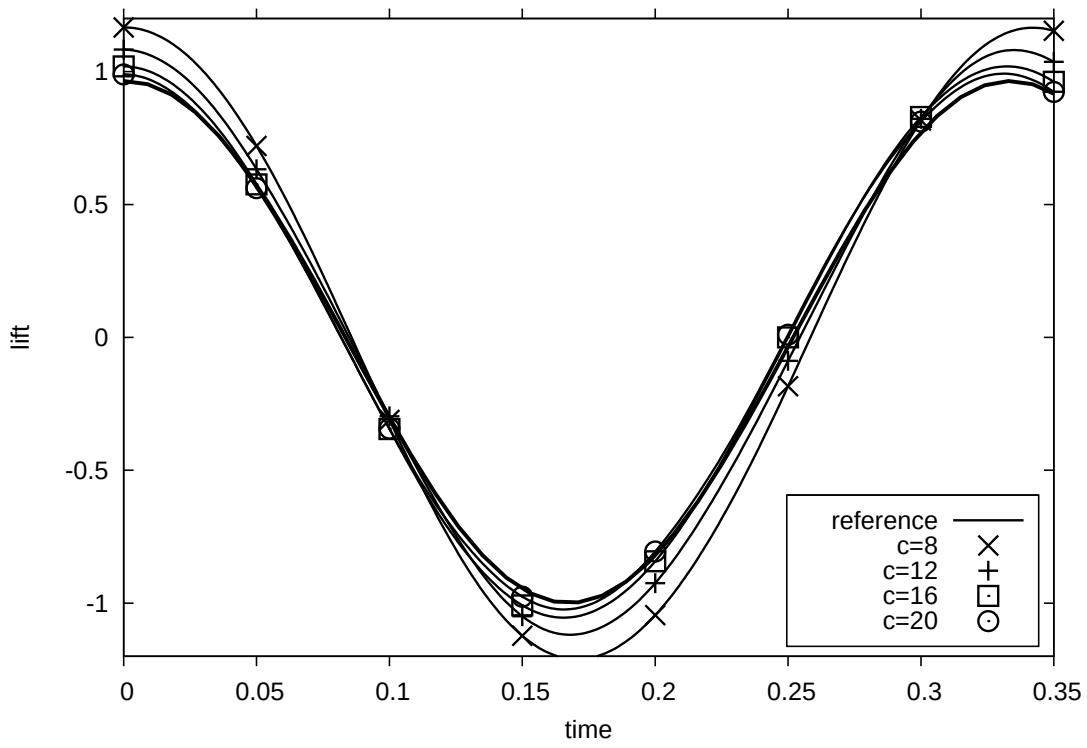


(b) extrapolated Δp

Figure 8.3: Pressure difference on cylinder and with extrapolation, 16k grid

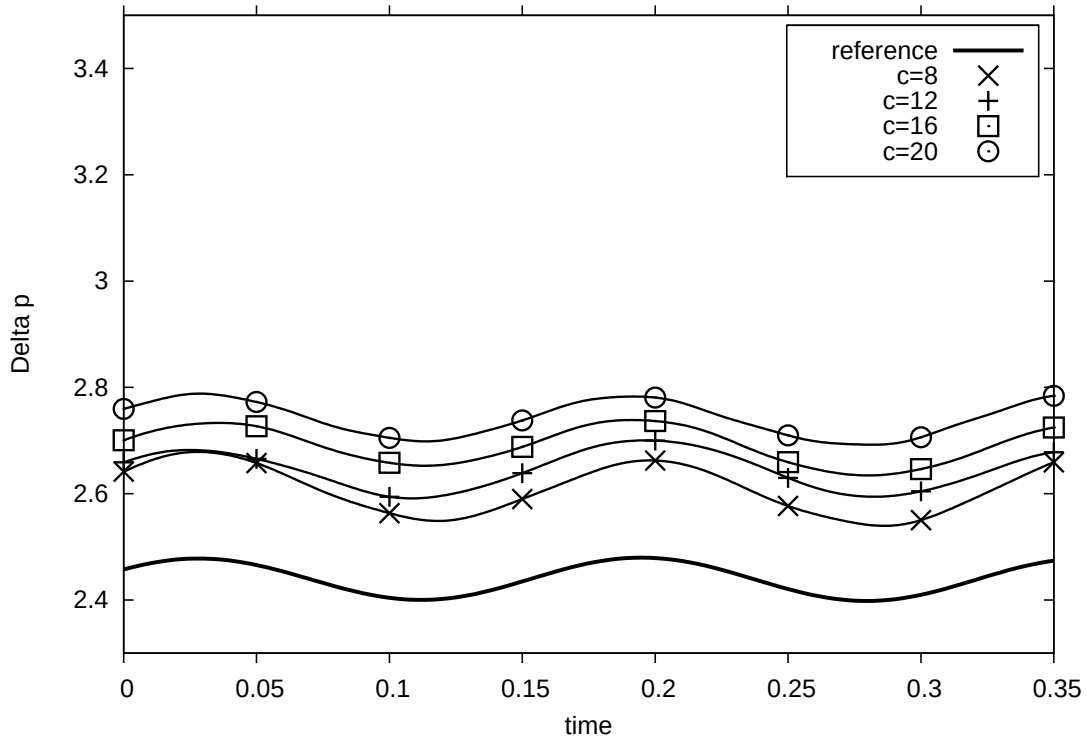


(a) drag

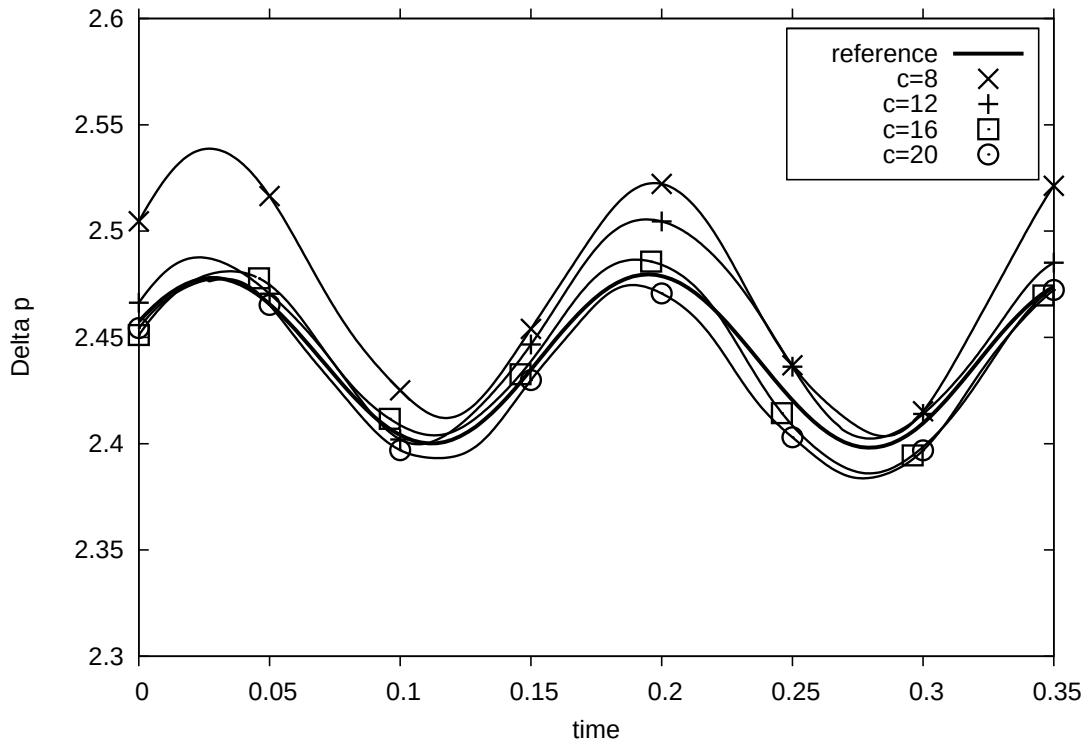


(b) lift

Figure 8.4: Forces on cylinder for various c, 66k grid

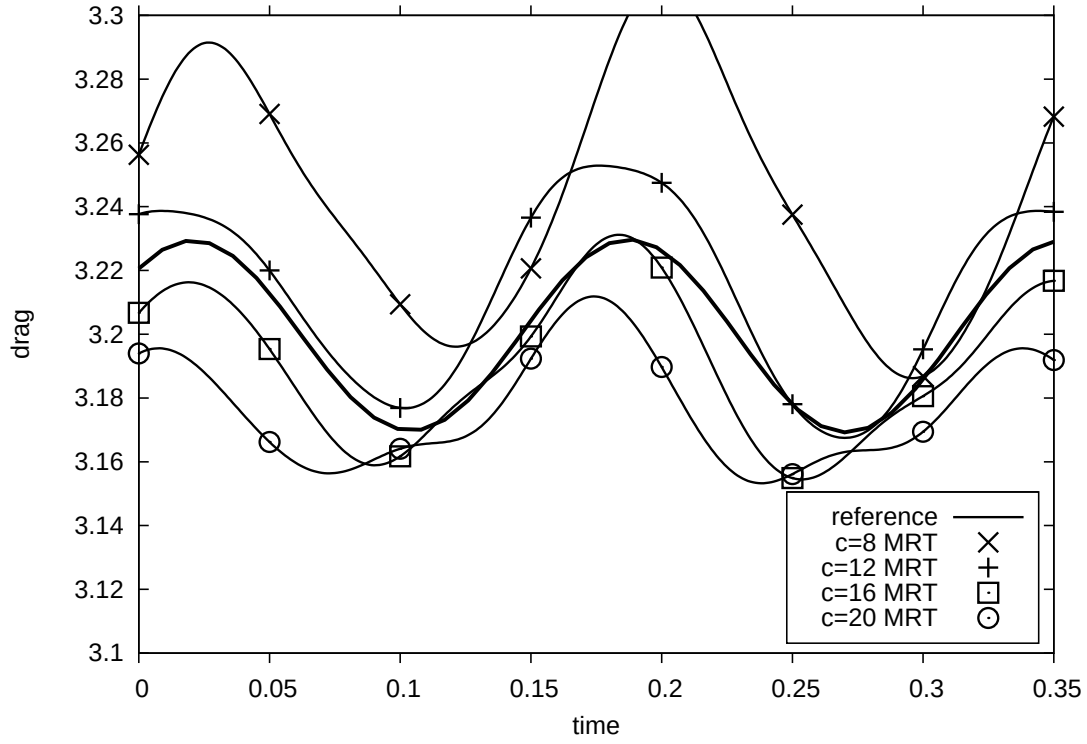


(a) Δp

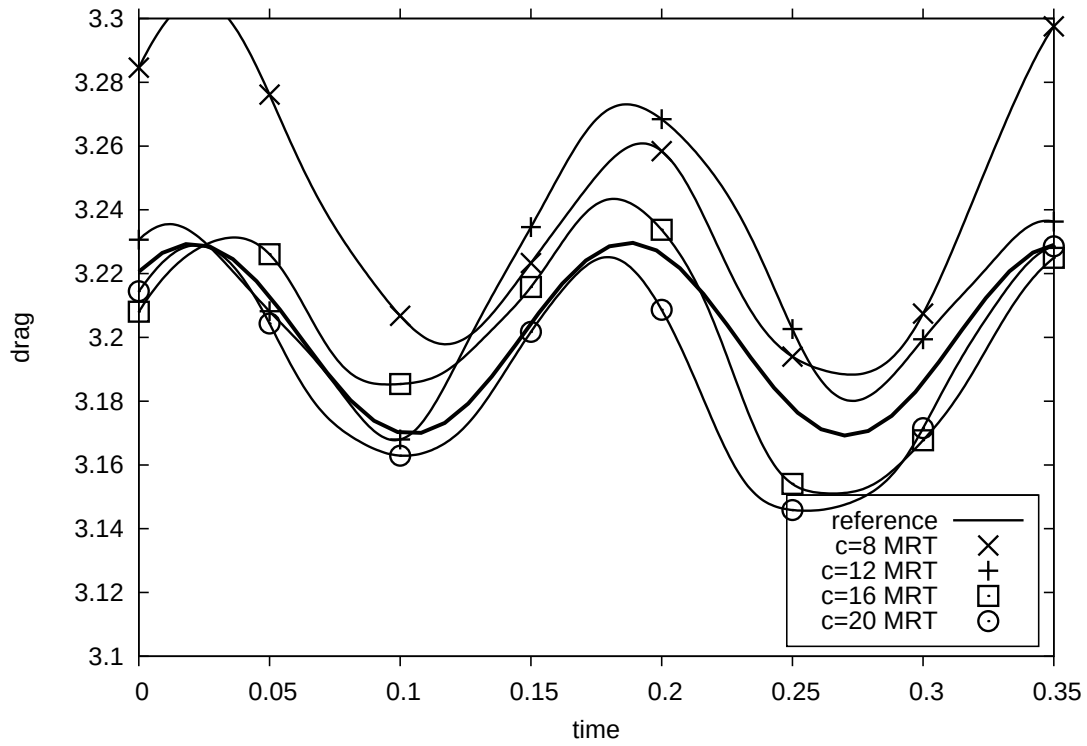


(b) extrapolated Δp

Figure 8.5: Pressure difference on cylinder and with extrapolation, 66k grid

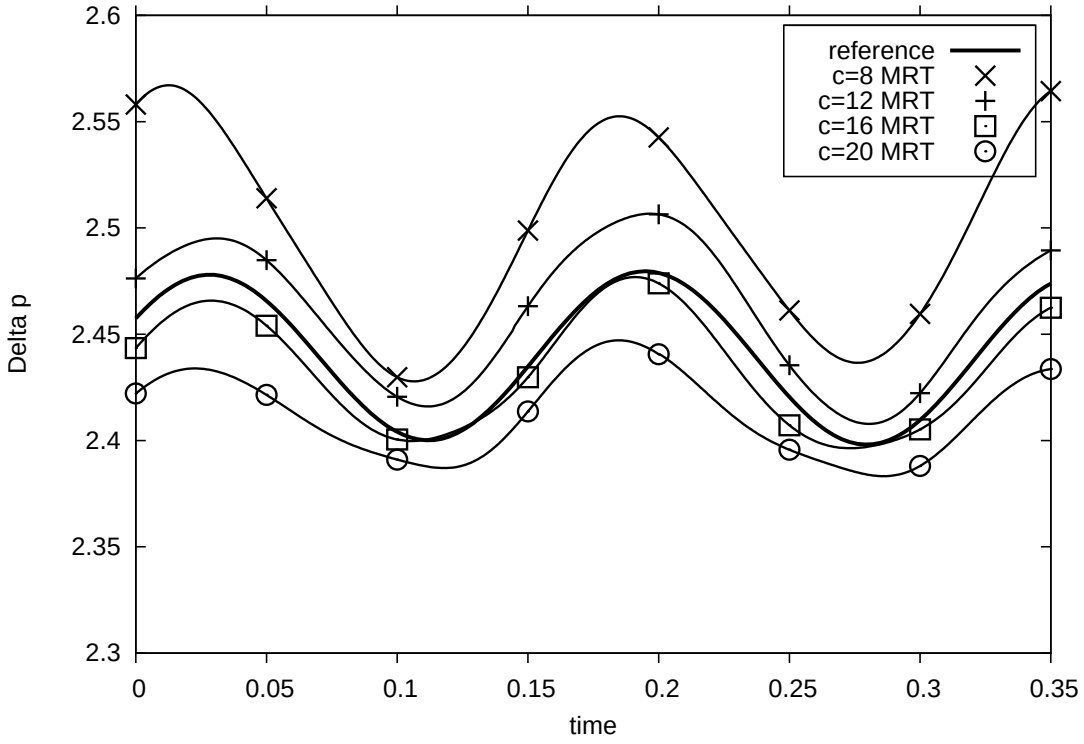


(a) 16k grid

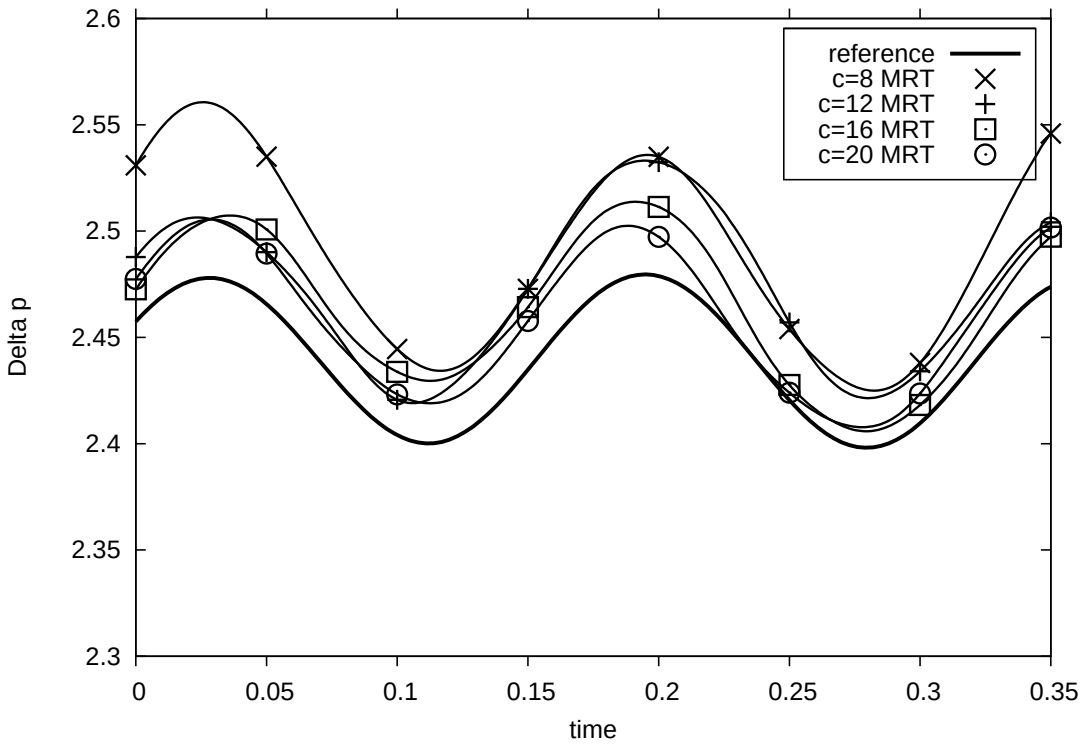


(b) 66k grid

Figure 8.6: Drag results obtained with MRT



(a) 16k grid



(b) 66k grid

Figure 8.7: Pressure difference with mrt and extrapolation

	c=20, $\Delta t = 1/1000$									
	SRT					MRT				
grid	C_{Lmin}	C_{Lmax}	C_{Dmin}	C_{Dmax}	St	C_{Lmin}	C_{Lmax}	C_{Dmin}	C_{Dmax}	St
4k	-0.475	0.616	2.849	2.885	0.2710	-0.540	0.680	**3.039**	**3.077**	0.2732
16k	-0.922	0.877	3.115	3.163	0.3021	-0.948	0.899	**3.156**	**3.212**	0.3021
66k	-1.027	0.992	3.110	3.189	0.3021	-1.044	1.004	**3.146**	**3.229**	0.3021

Table 8.4: Benchmark results: SRT vs. MRT with relaxation $0.1/\tau$, references: $C_{Lmin} = -1.023$, $C_{Lmax} = 0.987$, $C_{Dmin} = 3.168$, $C_{Dmax} = 3.230$, $St = 0.300$

8.2. Remarks

In this chapter we focused on important questions which arose initially and during the work on the thesis, they are:

- Does (our discretisation of) the discrete velocity model in practice yield the same results as Navier-Stokes-based CFD tools?
- Are higher order (spatial and temporal) schemes stable enough and accurate?
- How does the Mach number, resp., free simulation parameter c influence the accuracy/behaviour of the results?

By calculating different steady state CFD problems and showing convergence of the velocity profiles against analytical and FEATFLOW ([45]) results, we answered the first question positively. The results of the nonstationary *flow around cylinder* benchmark showed that higher order discretisations in space and time are not only stable and yield better results, but that they are absolutely necessary to solve advanced flow-problems. Only the second order upwind was able to show vortex shedding in the channel and reproduce crucial reference numbers for forces acting on the cylinder, while the Crank-Nicholson scheme approached the periodically oscillating flow accurately with large steps in time. It is interesting to approach (higher-order) semi-implicit schemes as an alternative to the Newton method, although the stability should be affected. Corresponding results will be included in a forthcoming paper ([22]).
With mesh refinement a proportional increase of the sound parameter c is necessary, on the other hand one should be careful to avoid dominating error-terms of the order $O(\mathrm{Ma}^{-1})$. It is not yet clear in how far nonstationary results can be improved by using multiple relaxation times. The use of extrapolation in certain areas is a simple strategy to improve the results, mostly the pressure along the inner boundary.
Low viscosity and Mach number lead to high values of the relaxation time $\frac{1}{\tau}$ which again means ill-conditioning of the system matrix. By using our *generalized equilibrium formulation* with special preconditioning we hope to possess an efficient and robust solver for stiff systems and large number of unknowns, steady state results should be otained by a monolithic approach combined with the multigrid method. The MG overhead can be avoided in time-dependent simulations, their condition number is improved by the additional identity term in the equation, so in each time-step few iterations of the BiCG-Stab solver (with GEF) reduce the linear defect. We refer to detailed results in the following chapter.

9

Solver efficiency and robustness

In this section we discuss the behaviour and efficiency of our nonlinear and linear solvers. Regarding the latter, we will verify mainly the special abilities of our preconditioners and the *generalized equilibrium formulation*. The analysis in Section 6.2 showed a significant influence of the sound speed c onto the condition numbers, which could be improved by modifying the system. We expect the proposed transport preconditioner — consisting of lower triangular blocks due to our special renumbering — to improve the condition especially for systems with dominating advection, independent of the refinement level. Convergence rates of the Krylov-space iterative solvers should improve significantly due to strong clustering of eigenvalues. The matrix allocation (see Fig. 6.1) already showed that the collision can become dominant (with a negative sign) on the off-diagonals. We expect to gain robustness against any large c with our collision preconditioner — obtaining a Block Jacobian by an alternative numbering — at the cost of giving up the transport oriented approach, loosing independence of the system size. Significant improvement of convergence rates and robustness is supposed to result from the GEF with special preconditioning, *at least* preserving the advantages of each special preconditioner. However, this stand-alone algorithm cannot be expected to give both, level- and c-independent convergence rates. An close look at the implicit system matrix reveals that the special preconditioner for our GEF 'only' takes into account the diagonals of the inverse transport operators weighted mainly by $1/\tau$. It leaves out of the account a significant part of the remaining entries of the advection discretisation, especially with increasing the system size. A Multigrid aproach is supposed to give satisfactory results in this case, finally removing the level-dependency of the convergence behaviour. We will analyse the efficiency of the proposed MG algorithms especially in combination with the GEF approach, which could not be determined a priori by looking at the eigenvalues. We try to find the most efficient solver/smoother combination used inside a MG cycle, and try the promising approach of using MG as precondtioner in Krylow-space iterative solvers, mainly for the *generalized equilibrium formulation*.

9.1. Nonlinear results

In this section we present shortly the results for the nonlinear solvers, comparing the performance of the fixed point iteration with the Newton scheme. The first tests are carried out for the monolithic approach which is naturally more difficult than time-stepping schemes with different Δt.

In Table 9.1 we provide the number of nonlinear iterations for *driven cavity* at $\mathrm{Re} = 100$ and $\mathrm{Re} = 600$, due to $\nu = \frac{1}{100}$, resp., $\nu = \frac{1}{600}$. We use a direct linear solver in each step, optimal for reducing the linear defect, and consider the second order upwinding. In Table 9.2 we show also results for the *flow around cylinder* benchmark (see [39]) on an unstructured grid at $\mathrm{Re} = 2$ and $\mathrm{Re} = 20$, due to $\nu = \frac{1}{100}$, resp., $\nu = \frac{1}{1000}$. Latter viscosity is the most demanding configuration among our testcases. We find that the fixed point iteration is only able to cope with small nonlinearities (low Re and number of unknowns). For higher values of Re, an appropriate damping has to be applied to ensure convergence. Also, increasing the number of grid points N is obviously affecting the fixed point solver. Remarkably, the choice of bigger c results in throughout less nonlinear iteration steps, while we showed that the linear problems then become more difficult due to bad condition number.
In contrast, the Newton scheme performs well in all cases and needs only few iterations without damping to effectively solve the nonlinear defect (similar to [44]). It means our monolithic approach can become a powerful tool for steady-state problems as, in case the linear solvers perform well, even starting from zero solution we are mostly done after 5 nonlinear iterations.

In Table 9.3 we show the convergence behaviour of both solvers for *driven cavity* at $\mathrm{Re} = 1000$, now with implicit Euler time-stepping. The initial defect is given for a zero starting vector and different time-step sizes. In the first iteration the defect is rising equally in the fixed point and Newton step, then the defects are starting to decrease quite moderately. But once the Newton scheme gets close to the solution, the quadratic convergence takes over reducing the defect to machine accuracy, while the fixed point solver needs more iterations, especially for the largest $\Delta t = 1$. Of course in the following time-steps, the Newton would perform even better, taking the solution from the previous time-step as initial guess, for example in the non-stationary *flow around cylinder* simulation.

Remark:

In practice it is easy to prolongate the obtained solution from the previous level and use it as starting guess on the refined mesh. The so gained initial defect is not zero, but in the first nonlinear step it does not grow so much like shown in Table 9.3 and will get sooner below the crucial bounds for the Newton scheme. On the other hand, moving from one sound parameter c to another, no matter if making c smaller or bigger, is not as trivial as increasing the level. Using the primary distributions without modification results for the new Mach regime in a flow profile some factor away from the actual solution. More feasible is taking the obtained macroscopic solution in ρ and $\mathbf{u}$ and either use the approximation $f_i \sim f_i^{eq}(\rho, \mathbf{u})$ as initial guess, or add the nonequilibrium part as a better approximation in the Chapman-Enskog expansion (see Appendix A2.2). However, in solving the discrete Boltzmann equation, we found that restart algorithms do not significantly reduce simulation times, as mostly not more than one nonlinear Newton step could be saved.

	Re 100		Re 600		
grid	fixed point	Newton	fixed point	fp (ω)	Newton
		c=1			
289	15	4	23	27	6
1089	20	5	81	34	6
4225	24	5	> 300	59	8
		c=10			
289	8	4	9	21	4
1089	12	4	17	21	5
4225	17	4	88	36	6
		c=100			
289	5	3	5	22	3
1089	7	3	7	22	3
4225	10	4	14	22	4

Table 9.1: *Driven cavity*: No. of iterations to reduce the nonlinear defect by 10^{-6} starting from zero, Newton against fixed point method (damping by parameter ω is required for small c)

	Re 2		Re 20		
grid	fixed point	Newton	fixed point	fp (ω)	Newton
		c=1			
572	13	4	180	22	5
2184	15	4	> 300	23	5
8528	16	4	> 300	23	5
		c=10			
572	11	4	30	20	4
2184	14	4	211	26	5
8528	16	4	> 300	46	5
		c=100			
572	6	3	8	20	3
2184	11	4	23	21	4
8528	15	4	> 300	118	6

Table 9.2: Same as above for *flow around cylinder*

	$\Delta t = 0.01$		$\Delta t = 0.1$		$\Delta t = 1$	
step	Newton	fixed point	Newton	fixed point	Newton	fixed point
0	1.63E-1	1.63E-1	1.63E-1	1.63E-1	1.63E-1	1.63E-1
1	6.03E+1	6.03E+1	6.53E+2	6.53E+2	7.25E+3	7.25E+3
2	2.97E-1	1.61E+0	4.21E+0	2.31E+1	2.50E+2	7.44E+2
3	9.05E-4	3.52E-1	2.42E-2	4.65E+0	2.28E+0	1.26E+2
4	1.22E-8	7.02E-2	1.60E-6	1.12E+0	1.56E-4	5.22E+1
5	9.43E-13	1.68E-2	8.36E-12	3.27E-1	7.66E-11	1.41E+1
6		4.07E-3		9.90E-2		4.01E+0
7		1.02E-3		3.08E-2		1.50E+0
8		2.60E-4		9.72E-3		4.02E-1
9		6.66E-5		3.08E-3		1.40E-1
10		1.72E-5		9.79E-4		4.76E-2

Table 9.3: Convergence behaviour of nonlinear schemes for *driven cavity* at Re $= 1000$, $c = 10$ and different Δt, reduction of nonlinear defect starting from zero, using a direct linear solver for 4225 grid points

9.2. Linear, single-grid results

We tested the proposed linear solvers and preconditioners for all our configurations at various Reynolds and Mach numbers. The direct stationary approach (4.17) for interesting configurations proved to be too difficult using a simple Richardson solver. In this section we will focus therefore on presenting results for the more advanced Krylov-space methods, starting with the *driven cavity* configuration at $\mathrm{Re} = 10$. First we provide the number of iterations to gain 6 digits for the linear defect using a preconditioned BiCG-Stab ([48]) solver (see Table 9.4). It gives very good results in those fields our preconditioners were made for: First, at $c = 1$ with "tr-pre", the very mild dependence of h for transport dominated problems due to the almost exact preconditioner is shown on all levels. Second, for small numbers of grid points N with "bl-jac" preconditioning, this preconditioner is stable with increasing c, however, the results are highly level-dependent. Unfortunately the convergence behaviour can fail unexpectedly, even for moderate configurations which is known for the BiCG-Stab scheme. Therefore, we also implemented a GMRES (see [38]) solver, taking advantage of its more monotone convergence behaviour. It still provides good results for transport dominated configurations, resp., for small N. Additionally, we get useful information about moderate and extreme configurations. While it is obvious that the separate use of "tr-pre" and "bl-jac" is not satisfactory, because it does not combine the advantages of both, the results (see Table 9.5) show the expected advantageous behaviour of the (preconditioned) GEF approach. For $c = 1$, the GEF without preconditioning still yields very good convergence on all levels. With our diagonal preconditioning GEF($\backslash$) we gain additionally robustness w.r.t. high values of c. This means that our preconditioned GEF combines two positive effects, which is due to our special discretisation technique. Nevertheless it remains the dependence on h, resp., N, for moderate and large values of c. That is why we use multigrid methods to overcome this last drawback.

	upw1				upw2			
grid	tr-pre	bl-jac	GEF	GEF(\)	tr-pre	bl-jac	GEF	GEF(\)
				c=1				
81	28	71	21	18	26	182	20	17
289	30	135	20	21	36	1000	23	20
1089	44	1000	24	30	43	1000	30	26
4225	52	1000	39	39	54	1000	41	36
				c=10				
81	110	56	82	51	95	62	110	44
289	100	229	161	88	96	179	148	164
1089	111	716	153	133	141	1000	1000	1000
4225	137	1000	1000	351	143	1000	199	937
				c=100				
81	1000	75	1000	60	1000	76	1000	92
289	1000	1000	1000	194	1000	349	1000	550
1089	1000	1000	1000	833	1000	1000	1000	1000
4225	1000	1000	1000	1000	1000	1000	1000	1000

Table 9.4: *Driven cavity* at Re = 10, No. of iterations to gain 6 digits, using BiCGStab

	upw1				upw2			
grid	tr-pre	bl-jac	GEF	GEF(\)	tr-pre	bl-jac	GEF	GEF(\)
				c=1				
81	42	91	33	28	40	146	34	28
289	48	178	36	31	49	334	37	32
1089	56	336	40	45	63	716	44	42
4225	67	718	49	53	83	1000	56	53
				c=10				
81	107	66	96	51	109	81	106	55
289	130	154	128	88	138	180	144	90
1089	158	351	158	129	164	423	182	147
4225	190	771	188	174	197	935	209	194
				c=100				
81	235	67	211	67	227	78	246	77
289	360	170	407	155	409	203	502	182
1089	524	400	744	340	619	539	1000	461
4225	714	950	1000	689	838	1000	1000	1000

Table 9.5: *Driven cavity* at Re = 10: No. of iterations to gain 6 digits, using GMRES

		upw1				upw2			
		unskaliert		skaliert		unskaliert		skaliert	
		plain	tr-pre	plain	tr-pre	plain	tr-pre	plain	tr-pre
c=1	N=81	108	28	108	28	131	26	131	26
	N=289	256	30	256	30	400	36	400	36
	N=1089	661	44	661	44	1000	43	1000	43
	N=4225	1000	52	1000	52	1000	54	1000	54
c=10	N=81	180	110	92	55	179	95	102	48
	N=289	345	100	172	57	346	96	217	58
	N=1089	711	111	398	66	895	141	483	72
	N=4225	1000	137	889	85	1000	143	1000	87
c=100	N=81	1000	1000	263	207	1000	1000	362	333
	N=289	1000	1000	360	258	1000	1000	497	292
	N=1089	1000	1000	652	329	1000	1000	1000	390
	N=4225	1000	1000	1000	519	1000	1000	1000	482

Table 9.6: *Driven cavity* at Re $= 10$, No. of iterations to gain 6 digits, using BiCG-Stab

		upw1				upw2			
		unskaliert		skaliert		unskaliert		skaliert	
		plain	tr-pre	plain	tr-pre	plain	tr-pre	plain	tr-pre
c=1	N=81	165	42	165	42	199	40	199	40
	N=289	353	48	353	48	482	49	482	49
	N=1089	810	56	810	56	1000	63	1000	63
	N=4225	1000	67	1000	67	1000	83	1000	83
c=10	N=81	204	107	136	86	223	109	144	86
	N=289	362	130	240	101	429	138	281	106
	N=1089	764	158	493	122	989	164	637	129
	N=4225	1000	190	1000	146	1000	197	1000	152
c=100	N=81	311	235	218	198	350	227	239	199
	N=289	478	360	399	294	788	409	489	332
	N=1089	1000	524	747	411	1000	619	1000	500
	N=4225	1000	714	1000	558	1000	838	1000	645

Table 9.7: *Driven cavity* at Re $= 10$, No. of iterations to gain 6 digits, using GMRES

9.3. Linear, twogrid results

We showed already that the GEF solver performs well for configurations with dominating transport operator, independent of the number of unknowns. Additionally, it is robust against large values of c with the block-diagonal preconditioning, but shows level-dependent convergence rates for the relevant, intermediate c-range and above. For this reason we extended our solver tool-box with the multigrid method, adapting our code for the use in a mesh-hierarchy and implementing grid-transfer operators. We apply linear and quadratic prolongation for the 1st and 2nd order upwind discretisation, respectively. However, the influence of restriction is not yet completely resolved. We obtain slightly better results using linear restriction for upw1, but for upw2 a constant (canonical) injection gave more stable results than a quadratic operator using the transpose of the saved prolongation matrix. So an optimization of effiency using second order upwind is feasible in this context.
In order to compare with single grid results from the previous Section 9.2, we will use a similar configuration. The first tests were performed for the *driven cavity* problem again at $\mathrm{Re} = 10$ to as in Table 9.5. As basic linear solver we used GMRES iterations due to the sustained monotone behaviour. Moreover, we consider only the *generalized equilibrium formulation* with optional additional preconditioning as superior to the basic discretisation where one has to decide *exclusively* between transport and collision preconditioning. Initial tests showed that the GEF has most potential in combination with MG, simply using iterative solvers based on the new formulation as smoothers. However, a special advantage of the GEF is that the optional collision preconditioning can be replaced by a multigrid preconditioner, possibly resulting in an improved algorithm. Therefore, we implemented several multigrid variants which differ significantly regarding efficiency. As mentioned before, the use of 'basic' multigrid cycles, simply iterating restriction, correction, prolongation and post-smoothing, gave unsatisfactory results. The idea to use multigrid as a preconditioner came up, this variant which we denote as MGPREC was very promising, even in an early stage of implementation. In the following, it is sufficient to give only the linear contraction rates of the respective twogrid algorithms, these can be extrapolated for a whole multigrid hierarchy. To avoid confusion, we enclose results for different multigrid variants in separate subsections. Subsection 9.3.1 contains tables for the basic MG iteration while in subsection 9.3.2 we show improved results using MG as preconditioner for the same testcase, both for our monolithic approach. Finally, in subsection 9.3.3 we show that the rates of MGPREC improve for the time-dependent system with decreasing time-step size.

9.3.1. Results for basic multigrid algorithm

As in Section 9.2, the starting guess is already close to the solution and we solve the linear defect for one final nonlinear iteration. We list the twogrid iteration with the main steps and practical specifications as

A compute initial defect

B1 restrict defect

B2 gain 4 digits on level-1 using GMRES with $GEF(\backslash)$

B3 prolongate and assemble solution

C perform s smoothing steps using GMRES with GEF or $GEF(\backslash)$

D if new defect is above 10^{-6} of initial defect, continue iteration with B1

So we have relative stopping criteria of $\varepsilon_1 = 10^{-4}$ for the coarse grid solver and $\varepsilon_2 = 10^{-6}$ for the whole linear iteration. In practice, higher accuracy of the defect correction on the coarse grid did not improve the final result, even a criterion of $\varepsilon_1 \sim 10^{-2}$ is feasible.
We obtain single grid results for comparison simply by omitting steps B1-B3 and only repeating $s = 16$ steps of the solver on the fine level, until the relative stopping criterion is reached. In Tables 9.8–9.10 we mainly compare the convergence depending on the number of smoothing steps $s = 4, 8, 16$ and the influence of collision preconditioning in the smooothing step C. We consider the upwinding of first and second order, observe the convergence bahaviour on different levels and especially for three sound parameters $c = 1, 10, 100$ which cover all configurations from transport dominated, through an intermediate case to a collision dominated configuration with extreme ill-conditioning.
For the final numbers we computed the average linear contraction between the second and last MG iteration. Remarkably, the best performance occurs always in the first iteration, easily gaining 2-3 digits on the initial defect. By not taking into account this initial jump, we obtain a faithful account on the effective performance throughout the whole linear iteration. We will now take a look at the actual results:

Starting with the first columns, we see that the single grid iteration performs well enough for the transport dominated case, but on the third level and for higher values than $c = 1$ the convergence rates degrade, especially since the problem is solved monolithically. For $c = 10$ and $c = 100$ the collision preconditioned $GEF(\backslash)$ is better than the plain GEF, but anyway the single grid iteration stalls at 0.9999. This is because the slope of linear convergence of the GMRES method starts almost horizontally when dealing with stiff configurations (compare Table 6.16). More than 16 iterations each would be needed to acquire a sufficient basis and drive the defect towards zero. A remedy is given by switching on the coarse grid correction, the rates improve significantly even using only $s = 4$ smoothing steps. Only the case $c = 1$ seems not suited for the multigrid method, here also $s = 16$ smooothing steps are necessary to have potential gain over the single grid solver. Although Table 9.8 shows that this multigrid approach is far from optimized, on the other hand it demonstrates the efficiency of our transport discretisation due to the special numbering technique. The whole MG overhead can be avoided and transport dominated problems can be solved with the GEF approach on a single grid.
To continue with the twogrid results, we observe general improvement of the results with higher refinement level up to 4225 grid points in the current tables. If only few smoothing steps are used, $GEF(\backslash)$ with block-diagonal preconditioning is first choice, while the plain GEF is continuously

catching up with the better rates up to $s = 16$. Another implication is that GEF may fail convergence, like in the case of $c = 100$ for the second order upwind. This lack of stability, although it does not disqualify completely the proposed method, was to be corrected in an alternative approach.
In total, looking superficially at the separate tables, the rates are improving from the top-left to bottom-right corner. More smoothing is obviously better and higher refinement has positive influence, but due to several runaway values which we account to lack of stability of the algorithm, we are not satisfied with the results. Therefore we will switch now to the multigrid preconditioned algorithm, which gives not only improved convergence rates but also shows more definite behaviour. The basic advantage of the first order upwind or the dependence on the number of smoothing steps are two aspects which will become more obvious in the following section.

		single grid		multigrid $s = 4$		multigrid $s = 8$		multigrid $s = 16$	
		GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1	289	0.1331	0.0557	0.54	0.48	0.52	0.47	0.18	0.05
	1089	0.5145	0.5512	0.54	0.56	0.52	0.47	0.44	0.34
	4225	0.7289	0.7729	0.61	0.92	0.51	0.50	0.47	0.44
upw2	289	0.0497	0.0573	0.56	0.52	0.53	0.49	0.14	0.05
	1089	0.1881	0.3422	0.49	0.48	0.51	0.50	0.33	0.29
	4225	0.7988	0.7774	0.43	0.46	0.45	0.44	0.46	0.45

Table 9.8: Comparison of convergence rates using GMRES smoother, Re=10, c=1

		single grid		multigrid $s = 4$		multigrid $s = 8$		multigrid $s = 16$	
		GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1	289	0.9617	0.9178	0.57	0.52	0.54	0.53	0.55	0.55
	1089	0.9829	0.9621	0.53	0.51	0.53	0.53	0.54	0.54
	4225	0.9999	0.9892	0.46	0.45	0.47	0.46	0.47	0.47
upw2	289	0.9414	0.8718	0.52	0.48	0.41	0.51	0.46	0.53
	1089	0.9747	0.9404	0.45	0.40	0.43	0.39	0.42	0.45
	4225	0.9874	0.9782	0.45	0.35	0.43	0.43	0.42	0.44

Table 9.9: Comparison of convergence rates using GMRES smoother, Re=10, c=10

		single grid		multigrid $s = 4$		multigrid $s = 8$		multigrid $s = 16$	
		GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1	289	0.9903	0.9669	0.81	0.52	0.63	0.50	0.54	0.50
	1089	0.9970	0.9882	0.70	0.53	0.59	0.51	0.53	0.51
	4225	0.9990	0.9969	0.64	0.54	0.57	0.53	0.54	0.53
upw2	289	0.9999	0.9399	–	0.68	–	0.65	0.56	0.60
	1089	0.9999	0.9843	–	0.67	0.61	0.66	0.57	0.64
	4225	0.9999	0.9999	–	0.57	0.50	0.57	0.48	0.55

Table 9.10: Comparison of convergence rates using GMRES smoother, Re=10, c=100

9.3.2. Results for multigrid as preconditioner

Although the last drawback of our linear solvers was removed, with on the whole level-independent results given in the previous section, the method obviously lacked stability. Next, we tried to combine the special properties of the GEF approach with modern numerical methods in a more sophisticated way. The result is a variable scheme which we denote as MGPREC and list with the following steps

- O perform outer GMRES iteration until relative stopping criterion of 10^{-6} is reached, for each matrix-vector multiplication $v = Ax$ call following preconditioning and use vector $\tilde{v} = MG^{-1}Ax$ instead
- A compute defect
- B1 restrict defect
- B2 gain 4 digits on level-1 using GMRES with $GEF(\backslash)$
- B3 prolongate and assemble solution
- C perform s smoothing steps using GMRES with GEF or $GEF(\backslash)$
- D give resulting preconditioned vector to outer iteration

How would a similar algorithm be implemented without GEF? The outer GMRES would not implicitly hold the inverse transport and would depend completely on the MG-preconditioner. For smoothing steps, a linear solver with either transport or collision preconditioning would be used, while we want to take advantage of both. Finally, in a nested algorithm, these drawbacks would be accumulated.
Now look at the results for our new scheme at first in Tables 9.11–9.13. The set-up corresponds exactly to the previous Tables 9.8–9.10, but here we give the average linear convergence between the second and last outer GMRES iteration. We notice a total improvement of the results while stability was also recovered. This behaviour is comparable to [49], [26] and shows the advantage of Krylov-space methods compared to 'simple iterative schemes' even in a multigrid framework.

Nevertheless, we want to discuss the results in detail. Compared to the previous section, the single grid results have significantly improved in the nested algorithm. It seems that the inner iterations are taking effect and no stalling takes place although s is still fixed to 16 steps. At the same time, the outer GMRES has good potential due to the 'small preconditioner'. Even the case $c = 100$ (Table 9.13) is manageable, but a level-dependence can be observed so the convergence behaviour would degrade further for higher refinement levels. At the latest when solving huge systems, for example an unstructured grid for flow-around-cylinder with $\sim 66\,000$ nodes, the linear convergence would definitely stall.

For the new MGPREC scheme, we look first at Table 9.11. The case $c = 1$ is obviously covered completely by our discretisation focusing on the transport operator. Therefore, the excellent contraction rates around 0.1 obtained on a single grid cannot be further improved by use of MG. Increasing the number s of our efficient smoother is better suited to improve the results than any coarse grid correction. However, the situation changes when comparing Tables 9.11 and 9.12. While the single grid results get worse for the sound constant $c = 10$, the excellent MG results remain and even improve in some cases. What is more, we acquire not only level-independent numbers, but mostly improvement for successive grid-refinement.

Another important aspect is that we have no divergent behaviour even for the smallest given number of smoothing steps. Now both discretisation methods are stable, while the first order upwind is continuosly better probably due to the sparser matrix. A final increase to $c = 100$ gives only slightly worse results, here the use of $GEF(\backslash)$ is advisable on lower levels and if few smoothing steps are used. However, the plain GEF smoother is at least as good for larger s. For upw2 on the highest levels it even just overtakes $GEF(\backslash)$ while it is cheaper, as no cumbersome collision blocks need to be inverted in a block-Jacobian approach. Performing a whole MG sweep, collision preconditioning should be only applied on lower levels where the effort is manageable, while the plain GEF would be considerably faster on the finest level and obviously as effective.

Comparing the results from left to right columns, in practice probably 8 smoothing steps would be applied, as the potential gain with doubling s is not sufficient. In general, increasing s does not boost the results as much as expected. The rates seem to improve with the order $O(\frac{1}{\sqrt{s}})$ rather than $O(\frac{1}{s})$. Therefore, we have some discrepancy from common multigrid theory which might be due to the non-optimal restriction.

A next test for our monolithic approach is presented in Tables 9.14–9.16, where the Reynolds number was increased to 100. While a simple MG iteration would show more instable behaviour, the rates here are still very good and can be sustained using $GEF(\backslash)$. We have to keep in mind that higher Re means a higher relaxation rate $\frac{1}{\tau}$ and to obtain optimal accuracy, according to Section 7.1, smaller values of c than for $\mathrm{Re} = 10$ are necessary. Consequently, increasing Re does not always mean that our solver will get worse. We should look at the efficiency depending on the overall optimal value $\frac{1}{\tau} = \frac{c_s^2}{\nu}$, and here our new MGPREC algorithm shows throughout excellent, level-independent convergence rates.

	single grid		multigrid $s = 4$		multigrid $s = 8$		multigrid $s = 16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1 289	0.0602	0.0475	0.16	0.12	0.05	0.04	0.11	0.03
1089	0.1228	0.0813	0.21	0.26	0.07	0.06	0.04	0.07
4225	0.1265	0.1384	0.24	0.32	0.13	0.17	0.06	0.09
upw2 289	0.0459	0.0756	0.21	0.14	0.09	0.07	0.06	0.06
1089	0.0498	0.0708	0.26	0.19	0.14	0.12	0.06	0.09
4225	0.1277	0.1590	0.31	0.31	0.14	0.17	0.13	0.13

Table 9.11: multigrid as preconditioner in GMRES, Re=10, c=1

	single grid		multigrid $s = 4$		multigrid $s = 8$		multigrid $s = 16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1 289	0.4212	0.2379	0.24	0.18	0.17	0.14	0.12	0.08
1089	0.4561	0.4310	0.19	0.16	0.15	0.14	0.14	0.11
4225	0.4825	0.5088	0.17	0.15	0.12	0.12	0.12	0.10
upw2 289	0.4374	0.3190	0.39	0.29	0.23	0.21	0.16	0.15
1089	0.5094	0.4578	0.30	0.25	0.15	0.17	0.11	0.13
4225	0.5488	0.5337	0.25	0.20	0.11	0.14	0.08	0.10

Table 9.12: multigrid as preconditioner in GMRES, Re=10, c=10

	single grid		multigrid $s = 4$		multigrid $s = 8$		multigrid $s = 16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1 289	0.7312	0.4293	0.64	0.26	0.37	0.19	0.24	0.14
1089	0.8312	0.6350	0.49	0.24	0.28	0.19	0.21	0.16
4225	0.8887	0.7908	0.36	0.22	0.23	0.18	0.19	0.16
upw2 289	0.8000	0.5129	0.83	0.47	0.61	0.33	0.40	0.28
1089	0.8880	0.7518	0.76	0.49	0.51	0.39	0.36	0.35
4225	0.9376	0.8819	0.62	0.43	0.38	0.35	0.27	0.30

Table 9.13: multigrid as preconditioner in GMRES, Re=10, c=100

		single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
		GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1	289	0.4122	0.2910	0.30	0.21	0.20	0.16	0.16	0.12
	1089	0.5090	0.4406	0.23	0.19	0.17	0.15	0.12	0.13
	4225	0.5439	0.4618	0.17	0.16	0.09	0.08	0.07	0.07
upw2	289	0.4840	0.3276	0.46	0.35	0.27	0.23	0.19	0.14
	1089	0.5532	0.4589	0.34	0.27	0.18	0.19	0.13	0.12
	4225	0.5894	0.5419	0.30	0.27	0.14	0.15	0.10	0.11

Table 9.14: multigrid as preconditioner in GMRES, Re=100, c=1

		single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
		GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1	289	0.7719	0.5002	0.63	0.29	0.43	0.23	0.30	0.20
	1089	0.8435	0.6505	0.50	0.25	0.29	0.20	0.22	0.18
	4225	0.8971	0.8145	0.36	0.23	0.24	0.19	0.20	0.17
upw2	289	0.8155	0.5104	0.84	0.48	0.61	0.35	0.42	0.28
	1089	0.9070	0.7800	0.76	0.48	0.51	0.40	0.38	0.37
	4225	0.9443	0.8924	0.63	0.46	0.42	0.39	0.33	0.35

Table 9.15: multigrid as preconditioner in GMRES, Re=100, c=10

		single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
		GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1	289	0.8887	0.6412	0.86	0.32	0.73	0.23	0.48	0.19
	1089	0.9372	0.7502	0.84	0.29	0.64	0.24	0.39	0.20
	4225	0.9685	0.8617	0.77	0.27	0.53	0.22	0.32	0.19
upw2	289	0.9119	0.7094	0.92	0.56	0.87	0.46	0.75	0.40
	1089	0.9706	0.8735	0.94	0.61	0.85	0.52	0.73	0.50
	4225	0.9860	0.9432	0.94	0.62	0.80	0.55	0.61	0.53

Table 9.16: Multigrid as preconditioner in GMRES, Re=100, c=100

9.3.3. Multigrid as preconditioner, time-dependent simulations

After showing results for the monolithic approach, we want to discuss the solution of nonstationary problems. Again we use the *driven cavity* testcase with the higher Reynolds number $\mathrm{Re} = 100$, but now we solve the nonstationary equation (4.3) with implicit Euler time-stepping. In the present Tables 9.17–9.19 we apply an initial timestep size of $\Delta t = 1$. Although the additional identity term improves the condition of the discrete system, obviously the results are not far from the monolithic case due to the large Δt. The results for $c = 1$ are very good, the convergence rates are reduced by the time-stepping and here even the single grid solver performs well. Higher values of c degenerate significantly the single grid results, so for that parameter range the multigrid-preconditioned variant is advisable. An interesting observation is that, compared to the previous tables 9.14–9.16, rather the convergence using the plain GEF smoother is improved, while the results for $GEF(\backslash)$ barely change.

For the next configuration, with corresponding results given in Tables 9.20–9.22, we reduce the time-step further to $\Delta t = 0.01$. This choice is rather suited to simulate non-steady flow problems like the *flow around cylinder* benchmark at $\mathrm{Re} = 100$, we would not use it to solve for steady-state. Starting again with $c = 1$, we explain that convergence rates lower than 10^{-6} signify the stopping criterion was reached after one iteration. Since the MG solver performs best in the first step, excellent results as in Table 9.20 are relative. However, the decision for MGPREC is not anymore clear, since the numbers obtained with the single grid solver are quite good. In simulations where Δt is as low as 0.001, usually stability is not an issue. Instead of the GMRES method one should consider therefore to use the BiCG-Stab algorithm on a single grid with the plain GEF to avoid computational overhead.
Looking at the last numbers for $c = 10$, we see some dependency on the number of unknowns for $s = 16$, but on a high level. The asymptotic is at least better for $s = 4$. Similarly, for the case $c = 100$ the multigrid results are level-independent. Considered the quite small time-step size, the final convergence is not excellent, but anyway some rates are reduced from 0.49 for single grid $GEF(\backslash)$ to 0.11 for multigrid $GEF(\backslash)$ using first order upwind, resp., from 0.84 to 0.34 for plain GEF smoothing with second order upwind.
Finally, these numbers also demonstrate the severe impact of extreme configurations due to low Mach number on the linear solvers and condition number. It is possibly even more severe than the monolithic steady approach.

	single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1 289	0.0210	0.0023	0.13	0.08	0.04	0.02	6.67E-3	1.30E-3
1089	0.0523	0.0206	0.09	0.08	0.04	0.03	8.98E-3	4.93E-3
4225	0.0845	0.0532	0.09	0.08	0.03	0.02	9.43E-3	6.32E-3
upw2 289	0.0392	0.0045	0.21	0.16	0.07	0.05	1.24E-2	3.78E-3
1089	0.0835	0.0255	0.18	0.14	0.05	0.05	1.32E-2	9.33E-3
4225	0.1400	0.0618	0.16	0.13	0.04	0.04	7.62E-3	9.85E-3

Table 9.17: Multigrid as preconditioner in GMRES, Re=100, $\Delta t = 1$, c=1

	single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1 289	0.3856	0.1414	0.41	0.22	0.25	0.15	0.16	0.07
1089	0.5530	0.3607	0.32	0.22	0.21	0.18	0.16	0.13
4225	0.6804	0.5818	0.24	0.19	0.19	0.17	0.16	0.14
upw2 289	0.5382	0.2576	0.64	0.39	0.40	0.25	0.24	0.16
1089	0.7435	0.5888	0.60	0.45	0.36	0.36	0.28	0.29
4225	0.8346	0.7580	0.46	0.42	0.30	0.36	0.26	0.29

Table 9.18: Multigrid as preconditioner in GMRES, Re=100, $\Delta t = 1$, c=10

	single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
upw1 289	0.6391	0.1349	0.74	0.26	0.47	0.17	0.26	0.08
1089	0.7937	0.4524	0.66	0.26	0.40	0.21	0.24	0.16
4225	0.8958	0.7253	0.54	0.26	0.34	0.22	0.23	0.20
upw2 289	0.7772	0.3798	0.79	0.49	0.70	0.36	0.45	0.28
1089	0.9272	0.7674	0.81	0.58	0.84	0.49	0.51	0.44
4225	0.9645	0.8941	0.79	0.62	0.67	0.53	0.49	0.52

Table 9.19: Multigrid as preconditioner in GMRES, Re=100, $\Delta t = 1$, c=100

	single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
u1 289	8.81E-7	6.85E-8	1.87E-5	3.21E-8	1.19E-7	3.21E-8	1.19E-7	3.21E-8
1089	7.86E-7	2.62E-7	3.19E-7	9.20E-8	3.32E-8	9.20E-8	3.32E-8	9.20E-8
4225	5.62E-7	9.41E-8	7.24E-5	4.15E-5	3.31E-7	8.26E-7	3.31E-7	8.26E-7
u2 289	2.06E-7	5.99E-8	6.61E-5	8.07E-7	1.96E-7	4.32E-8	1.96E-7	4.32E-8
1089	6.17E-7	9.46E-7	8.96E-5	4.39E-5	6.15E-7	1.21E-7	6.15E-7	1.21E-7
4225	1.89E-7	4.91E-7	1.80E-4	1.62E-4	2.42E-7	7.00E-7	2.42E-7	7.00E-7

Table 9.20: Multigrid as preconditioner in GMRES, Re=100, $\Delta t = 0.01$, c=1

	single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
u1 289	3.81E-3	9.11E-7	1.84E-1	2.28E-2	4.17E-2	9.34E-4	1.60E-3	2.16E-7
1089	1.59E-2	1.05E-3	1.77E-1	5.79E-2	4.45E-2	8.54E-3	3.21E-3	2.04E-4
4225	4.46E-2	1.37E-2	1.48E-1	7.48E-2	4.30E-2	1.97E-2	6.55E-3	1.68E-3
u2 289	6.35E-3	3.90E-5	2.49E-1	6.29E-2	4.17E-2	5.74E-3	2.26E-3	3.84E-5
1089	2.32E-2	1.96E-3	2.63E-1	1.22E-1	4.82E-2	2.52E-2	5.47E-3	1.17E-3
4225	7.31E-2	2.34E-2	2.48E-1	1.57E-1	5.67E-2	5.43E-2	1.04E-2	7.51E-3

Table 9.21: Multigrid as preconditioner in GMRES, Re=100, $\Delta t = 0.01$, c=10

	single grid		multigrid $s=4$		multigrid $s=8$		multigrid $s=16$	
	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)	GEF	GEF(\)
u1 289	5.09E-1	5.82E-2	7.13E-1	1.86E-1	4.39E-1	8.84E-2	1.89E-1	2.26E-2
1089	6.61E-1	2.45E-1	6.40E-1	2.19E-1	3.61E-1	1.33E-1	1.90E-1	6.78E-2
4225	7.77E-1	4.89E-1	5.25E-1	2.35E-1	3.19E-1	1.69E-1	1.86E-1	1.13E-1
u2 289	5.90E-1	9.73E-2	7.37E-1	3.19E-1	6.07E-1	1.72E-1	2.80E-1	6.15E-2
1089	7.56E-1	3.58E-1	7.76E-1	4.40E-1	6.90E-1	2.72E-1	3.35E-1	1.65E-1
4225	8.41E-1	5.97E-1	7.65E-1	5.15E-1	5.90E-1	3.56E-1	3.41E-1	2.92E-1

Table 9.22: Multigrid as preconditioner in GMRES, Re=100, $\Delta t = 0.01$, c=100

9.4. CPU-time results for multigrid with W-cycle

Finally, we compare the performance of our current code (denoted FeatLBE) against FEATFLOW [45] using again the stationary *flow around cylinder* benchmark. We use in both codes the same coarse mesh, but convert in the Boltzmann simulation each quadrilateral into 4 triangular elements (see Fig. A1.2). So, the total number of unknowns on a similarly configured mesh differs, we have for level l in FeatLBE approximately the same matrix size like for level $l+1$ in FEATFLOW.
For a better interpretation of the runtime, we provide in Table 9.23 additionally numbers for accuracy which should be good for both codes due to the unstructured mesh with refinement around the cylinder. FEATFLOW yields reasonable results for drag and lift with the Upwind method, but 267072 grid points are needed to get close to the reference values. However, the edge-oriented stabilisation method (EOFEM) gives excellent results already for 16848 nodes, which shows the potential of modern numerics and the advantage after a decade of development in FEATFLOW. Our code provides results within 1% of the reference (drag between $5.524-5.635$) only from 16848 grid points. Better results require a higher refinement level and also the sound parameter c must be chosen accurately. However, too large c affects performance and amplifies the discretization-error. Our numerical tests (see Sec. 7.1) indicate in this case $c_{opt} < 5$ on the highest shown refinement level.
The computational times shown in Table 9.24 result from a monolithic steady approach for the benchmark problem. We start from the Level-1 solution (by interpolation) and use stopping criteria of 10^{-8} for the nonlinear defect, resp., 10^{-4} linear gain each. The Boltzman equation (with the GEF) is solved in a GMRES iteration with a multigrid sweep as preconditioner in each step. We implemented a W-cycle as shown in algorithm 9.4.1 and decided to perform 16 smoothing-steps. The combination of CPU times and accuracy of our code (omitting the coarse grid solver) is on a level with the Upwind method of FEATFLOW, giving on the whole similar results for a comparable number of unkowns. FeatLBE needs similarly few linear multigrid sweeps of the W-cycle. The rates for $c = 2$ are decreasing compared to $c = 1$, but remain stable for a higher number of unknowns. However, FeatLBE needs less nonlinear steps due to the continuous Newton method. In FEATFLOW the nonlinear efficiency is significantly improved due to the symmetry of the EOFEM and an unperturbed Jacobian (see [35]). The edge-oriented approach is giving superior accuracy at the cost of a slight increase in CPU times, if any. On the whole, FeatLBE cannot (yet) compete with the highly optimized CFD code, nevertheless our Boltzmann discretization solves the problem with good accuracy. At the same time we observe linear convergence independent of the refinement level due to efficient use of multigrid as preconditioner in the GEF approach.

FEATFLOW		Upwind		EOFEM	
grid points	total unkn.	drag	lift	drag	lift
16848	42016	5.7460	0.0070	5.5803	0.0101
66976	167232	5.6196	0.0103	5.5789	0.0104
267072	667264	5.5882	0.0108	5.5793	0.0106
FeatLBE		c=1		c=2	
grid points	total unkn.	drag	lift	drag	lift
4264	38376	5.6676	0.0413	5.6490	0.0448
16848	151632	5.5287	0.0121	5.5403	0.0123
66976	602784	5.5398	0.0102	5.5863	0.0103

Table 9.23: FEATFLOW vs. FeatLBE results for *flow around cylinder*: Drag and lift coefficient, references 5.5795, resp., 0.0106

FEATFLOW		Upwind		EOFEM	
grid points	total unkn.	NL/AVMG	CPU	NL/AVMG	CPU
16848	42016	7/6	47	3/7	32
66976	167232	6/5	145	3/6	125
267072	667264	5/5	443	3/7	613
FeatLBE		c=1		c=2	
grid points	total unkn.	NL/AVMG	CPU	NL/AVMG	CPU
4264	38376	4/8	31	4/11	45
16848	151632	3/7	139	3/11	225
66976	602784	3/7	774	3/12	1308

Table 9.24: FEATFLOW vs. FeatLBE results for *flow around cylinder*: Nonlinear (NL)/Average multigrid sweeps (AVMG) and CPU time, compare Table 9.23

Algorithm 9.4.1 Recursive multigrid algorithm with W-cycle

MG $(\mathbf{x},\mathbf{b},l)$
auxiliary vectors $\mathbf{d},\mathbf{v}$
if (l.eq.0) then
exact solution $\mathbf{x} = A_0^{-1}\mathbf{b}$
else
compute defect $\mathbf{d}_l = A_l\mathbf{x}-\mathbf{b}$
restriction $\mathbf{d}_{l-1} = R\mathbf{d}_l$
1. recursive MG step $\mathbf{v}_{l-1} = MG(\mathbf{v}_{l-1},\mathbf{d}_{l-1},l-1)$
prolongation $\mathbf{v}_l = P\mathbf{v}_{l-1}$
assemble solution $\mathbf{x} = \mathbf{x}-\mathbf{v}_l$
smoothing $\mathbf{x} = S_l^s(\mathbf{x},\mathbf{b})$
compute defect $\mathbf{d}_l = A_l\mathbf{x}-\mathbf{b}$
restriction $\mathbf{d}_{l-1} = R\mathbf{d}_l$
2. recursive MG step $\mathbf{v}_{l-1} = MG(\mathbf{v}_{l-1},\mathbf{d}_{l-1},l-1)$
prolongation $\mathbf{v}_l = P\mathbf{v}_{l-1}$
assemble solution $\mathbf{x} = \mathbf{x}-\mathbf{v}_l$
smoothing $\mathbf{x} = S_l^s(\mathbf{x},\mathbf{b})$

9.5. Conclusions

To estimate the results given in this chapter, we must view them in the context of basic DVM theory — especially Sec. 3.3.1 concerning the dependence of space discretisations on the Mach number — which assumptions were validated in Chapter 7. For reasons of accuracy, the second order upwind discretisation is essential, but we can only fully exploit the superior scheme by simultaneously choosing the sound parameter c large, otherwise the gain of accuracy will be canceled by compressibility errors. The same holds for systems with increased number of unknowns due to grid refinement. Consequently, mere convection dominated (steady-state) configurations, that we can solve efficiently due to our special transport preconditioning, play only a minor role in practice.
The convergence behaviour exhibited by our iterative solution methods confirms that higher c immediately affects the condition number of the system matrix. Focusing on the dominant collisions in constructing a block Jacobian preconditioner was a remedy only for small systems. We could always wear out the efficiency of our basic solvers, mostly by a combination of high grid refinement and large c. Against such objective testing even the GEF approach could not make a stand, although it gave some robust results, combining the advantages of our preconditioning techniques. A critical look reveals that the first order upwind gave mostly better linear convergence rates than the second order scheme on a similar mesh. This is surely due to the sparser matrix; we needed only three variables for the difference quotient (with interpolation) for upw1, while upw2 needs twice the number of entries. This is a fundamental observation for the GEF with special preconditioning which takes into account only the diagonals of the submatrices (see Sec. 6.4.3). A less sparse matrix means more entries not 'covered' by the preconditioner, the condition number of the system $C^{-1}A$ unlikely being close to unity and consequently more work for the proposed Krylow-space methods resulting in more iteration steps.
The use of a coarse grid correction removed most level-dependent behaviour in the first implementation. However, a basic MG iteration lacked stability, we obtained best results with the new MG-PREC solver. In this variant the additional block-Jacobian preconditioning in $GEF(\backslash)$ is replaced by one MG cycle, which gives a more complete preconditioner than the inverse collision-part on a single grid. Anyway, this sophisticated algorithm is only possible due to our special *generalized equilibrium formulation* and the use of $GEF(\backslash)$ is still important as an efficient smoother. The new convergence rates were significantly better, because in each preconditioning step the MG cycle gains 1-3 digits so a couple of Krylow-space iterations is enough to reduce the linear defect even in the monolithic approach. Excellent rates around 0.1 are obtained for the range of c between 1 and 10. Beyond, a contraction rate below 0.5 is usually obtained with 8-16 smoothing steps, also improving with higher refinement level, while for the single grid solver it is increasing with the number of unknowns, reaching 0.98 and worse.
On the whole, surely our numerical results cannnot challenge high-end codes which deal easily with poisson-problems on a unit square with zero boundary. Established CFD tools have years of advance to develop better grid-transfer operators, efficient coarse grid solvers and sophisticated stabilisation methods. This is not (yet) our pretence, we have a prototypical discretisation, a new GEF approach for the discrete Boltzmann equation and a multigrid algorithm which still needs verification and optimization. The initial results are interesting and significant and, in view of developing numerical alternatives for the Lattice Boltzmann method, the efficiency shown in this chapter is promising for the extreme case of monolithically obtaining steady state solutions. A first implication which directly presents itself is an heuristic approach for a whole MG sweep with maximum depth: As coarse grid solver one should apply a fast direct method, but also our $GEF(\backslash)$ can deal with small systems due to collision preconditioning. Similarly, smoothing steps on the coarser levels should be performed with $GEF(\backslash)$, where the effort is manageable, while on the higher levels smoothing with the plain GEF would be similarly effective but faster. Especially

on the finest refinement-level, additional work makes a difference for CPU-times, so all processes there should be computationally cheap.

The results for our nonstationary simulations gave further practical conclusions. The condition number of the discrete system with small Δt can be significantly improved, as far as making the use of multigrid unnecessary. For our benchmark calculations of oscillating *flow around cylinder* we refrained therefore to plain GEF iterations of the BiCG-Stab solver, even without collision preconditioning. This combination turned out to be most efficient, usually few (around 5) iterations were effectively performed. The BiCG-Stab algorithm needs less CPU-time per iteration compared to the GMRES which has superlinear runtime with the number of accumulated basis-vectors. The efficient transport discretisation in the GEF can deal with most time-stepping simulations when applied in Krylow-space methods.
On the other hand, when the Mach number is chosen very small and the time-step size should be rather large, we can provide sufficient time-aacuracy using the second order Crank-Nicholson scheme and, if necessary, good efficiency with the MGPREC solver. However, the direct solution of steady-state problems is where we expect most gain in efficiency over the LBM which often needs micro-timestepping.

10

Summary and outloook

In this thesis, an efficient space discretisation introduced for the radiative transfer equation was applied to the Boltzmann equation with discrete velocities and the numerical methods were widely advanced. In our final summary, we review the main achievements and evaluate the ongoing research.

Spatial discretisation aspects. A finite difference upwind discretisation of first or second order was applied to the DVM. Due to a special sorting technique we obtained lower triangular matrices for the transport operator of the PDE with constant characteristics. We introduced a new, algebraic reformulation, solving the equation for the equilibrium term. The resulting system matrix implicitly holds the inverse transport and is well conditioned for convection dominated configurations. Additional preconditioning is possible for other relevant cases with dominating collisions.

Time discretisation aspects. The GEF is applicable to time-stepping discretisations and to the monolithic approach which we use primarily to solve directly for steady-state problems. We apply the second order Crank-Nicholson scheme to simulate time-dependent behaviour. Large timesteps and the direct-solution approach require efficient nonlinear and linear iterative solvers.

Nonlinear solver aspects. 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.

Linear solver aspects. The low Mach number regime which is needed to approximate an incompressible limit leads to bad conditioning. Separate preconditioners with focus on the lower triangular transport blocks or off-diagonals due to local collisions were not sufficient. The GEF approach could combine the advantages of both, solving efficiently small collision dominated systems and all transport dominated problems but lost performance for the intermediate case on high refinement levels. A sophisticated application of the GEF resulted in a prototypical algorithm where multigrid is used as a preconditioner in Krylow-space methods and achieves very good convergence rates for all configurations. Extensive numerical results were obtained for the stationary and nonstationary *flow around cylinder* benchmark on unstructured grids in accordance with CFD reference values.

The recent development of computer-systems with enhanced performance mainly resulting from clustering of compute-power has an increasing impact on the CFD community. This development not only accelerates — although the speed-up of established software due to higher FLOP/s is demonstrably stagnating — but also shifts the focus of traditional numerical methods. In short,

the use of hardware-oriented numerics is supposed to have higher potential and close the developed gap. The Lattice Boltzmann method seems predestined for large scale computing and experimental GPU clusters due to its simple arithmetic, we mention the SKALB project (*www.skalb.de*) as a recent initiative.

Nevertheless, one should not forget the importance of efficient numerics, we mainly contradicted the use of a time-stepping method to obtain the solution of steady-state problems. Moreover, it was a very interesting and challenging task to approach the discretisation of the Boltzmann equation without prejudice. Eventually, we obtained a second order discretisation in space (and time) that can be adapted to all D2QX lattice-sets on arbitrary triangular meshes. The efficient transport solver, resp., GEF approach can be in future extended to the 3D case, as the underlying node numbering is based on topological sorting and graph theory.
Another question is whether this technique can be reasonably retained for higher than second order finite differences. Such schemes would have a similar implementation, but an upwind of third, fourth etc. order makes arise some stability problems. Therefore, we should consider for example a *discontinuous Galerkin* approach as a higher order scheme to reduce the discretisation error and the number of unknonws considerably. We would obtain smaller systems while the Mach number should be chosen significantly small to reduce compressibility errors in parallel.
Also in view of future alternative discretisations, theoretical aspects were a fundamental part of our research. We took apart the aggregate error in the Boltzmann discretisation, and analysed the specific Mach number dependence. The compressibility error is hardly discussed per se in literature. Usually, the combined asymptotical behaviour of the on-lattice discretisation is given, but rarely absolute numbers which play an important role in practice. We derived methods for selective explicit or implicit treatment of the Boltzmann equation and demonstrated the high complexity of a collision/advection implicit off-lattice discretisation. However, at the latest with the monolithic solver, we showed that it is possible to overcome stability restraints which apply to the LBM.

In our numerical comparison against the incompressible Navier-Stokes solver FEATFLOW, we had to account for the higher number of variables in the Boltzmann approach and compressibility effects which would vanish only in the asymptotic limit of $\mathrm{Ma} \to 0$. Solving a steady state problem on similar grids and hardware, the tests showed for the obtained computation times some advantage for the macroscopic approach.
Another pending, important test is a numerical comparison against the LBM. However, simple CPU-time results running our serial working-code on a workstation against a supercomputer system would be less important. The interperetation should be more elaborate, involving for example the overall operations needed until a target steady-state is reached. Comparing the arithmetic operations performed mainly by our linear solvers and by the explicit time-stepping algorithm, we should obtain a measure for numerical efficiency of the two methods. Another aspect of the set-up should deal with local accuracy, testing some advanced problems on highly adapted against uniform meshes.
In the end, we expect different results for non-stationary and stationary problems, and the presented monolithic approach should prevail for the latter and for configurations where stability is too restrictive for the LBM.

Part IV

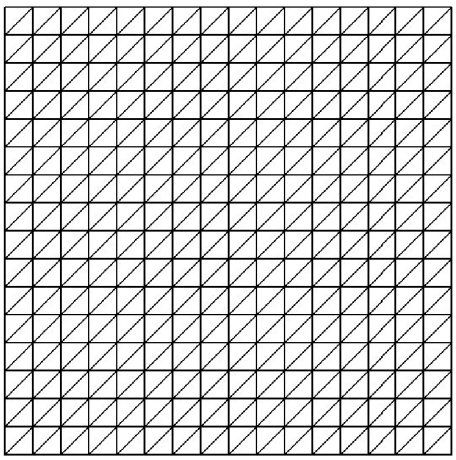
Appendix

A1

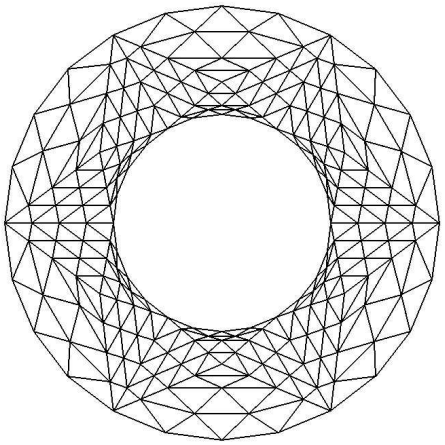
Numerical testcases on hierarchical grids

For a rigorous numerical analysis of our discretisation techniqes, resp., numerical solution methods we chose three different 'classical' problems from the field of CFD with varying difficulty level. First, we compared to the given analytical solution for *Rotating Couette flow*, also used in [11]. Second, we performed simulations of the *driven cavity* problem at various Reynolds numbers as in [31]. We took given CFD results from FEATFLOW on a highly refined mesh as reference, the same accounts for the *flow around cylinder* benchmark (see [39]). Furthermore, the latter provided a basis for the nonstationary simulations presented in the thesis, comparing different time stepping schemes.

As seen in Fig. 4.1, our spatial discretisation is designed for general (triangular) grids. Thereby, we have 3 degrees of freedom (DOF) per element for the 1st order upwinding, and 3 additional unknowns in the edge midpoints for the 2nd order upwinding. Starting from a coarse mesh we get one level of refinement - and successively a mesh hierarchy used in a multigrid algorithm - simply by connecting the edge midpoints. Thus, each element is divided into 4 new ones. In Fig. A1.1 we present a structured grid on the unit square, used for the *driven cavity* testcase, together with another less structured grid which we use for the *Rotating Couette flow*. Finally, in Fig. A1.2 a sequence of grids is shown which is locally adapted around the inner boundary component. On these grids we calculate the *flow around cylinder* benchmark. However, the upwind discretisation is carried out similarly on all three meshes.

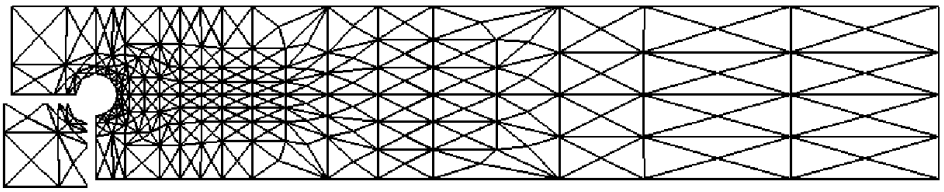


(a) Uniform square lattice for *driven cavity*

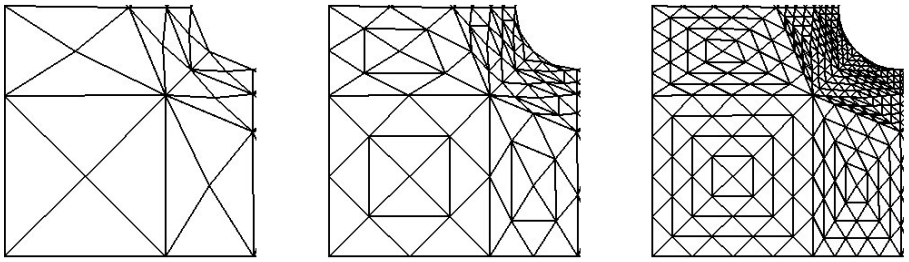


(b) Unstructured grid for *Rotating Couette flow*

Figure A1.1: Simple domains for numerical tests



(a) Channel



(b) Grid hierarchy

Figure A1.2: Standard grid for *flow around cylinder*

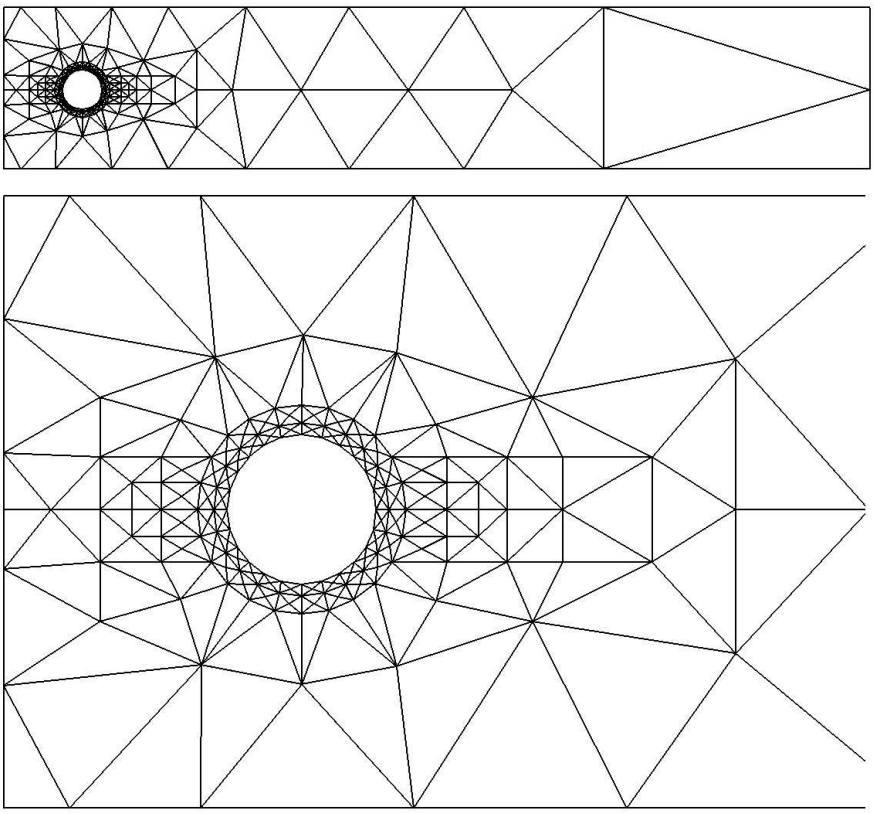


Figure A1.3: Highly adapted grid for *flow around cylinder*

	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 A1.1: *Driven cavity* mesh information

	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 A1.2: *Flow around cylinder* mesh information

configuration	ν	u_{max}	Re	regime
	lid driven cavity			
dc_{Re10}	1/10	1	10	stationary
dc_{Re100}	1/100	1	100	stationary
dc_{Re1000}	1/1000	1	1000	stationary
	flow around cylinder			
$bench_{Re2}$	1/100	0.3	2	stationary
$bench_{Re20}$	1/1000	0.3	20	stationary
$bench_{Re100}$	1/1000	1.5	100	nonstationary

Table A1.3: Main testcases (*driven cavity* and *flow around cylinder*) and their configurations

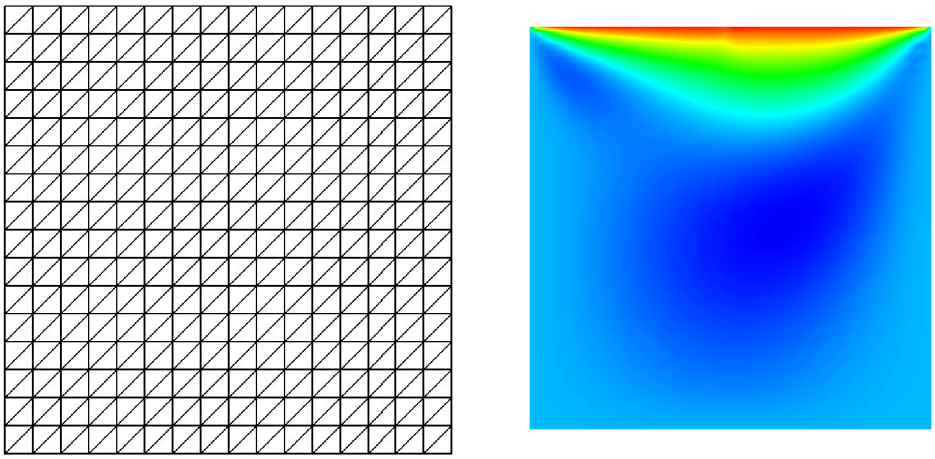


Figure A1.4: Grid and u_1 velocity for *driven cavity*

1.1. Driven cavity

The first presented configuration depicts our standard testcase, i.e. most results for linear solvers are calculated using the *lid driven cavity* model. The model is defined on a unit square by way of Dirichlet conditions for the velocity. To obtain the given macroscopic boundary-velocity we use Ladd's scheme described in Sec. 4.3. The west, south and east walls have no-slip conditions, only the north 'lid' is moving with velocity $\mathbf{u} = \binom{1}{0}$. The Reynolds number is therefore adjusted solely using the viscosity and it holds $\mathrm{Re} = \frac{1}{\nu}$ as seen in Table A1.3. Finally, we use a structured mesh as seen in Fig. A1.1, and obtained a reference solution from FEATFLOW on a highly refined mesh with 263 169 unknowns. Despite the structured configuration, the situation is not comparable to the Lattice Boltzmann method. The transport step in our case is not modeled as a simple shift to neighbouring nodes along the edges. Our finite difference upwind discretization up to second order goes back along the characteristics and on the intersection of elements usually interpolation between the nodes on an edge is necessary. So, while a matrix corresponding to on-lattice Boltzmann schemes would require one entry off the main diagonal, we have up to 6 of them in the case of second order upwinding, resp., 2 for the first order scheme. Only a structured hexagonal mesh would reduce this number and the FD scheme with interpolation would fall back to the actual nodes for every direction.

Remark: The prescribed boundary velocity is discontinuous due to the lid moving with constant speed of $u = 1$ while the walls are at rest. The question is how to define the values right in the upper corners, that means if the given discretisation defines there degrees of freedom, as in the case of our method. We chose the variant that the corners belong to the walls with velocity $\mathbf{u} = 0$. The discontinuity might be a reason why the asymptotic behaviour is slightly worse than second order in the Mach number. However, the results reproduce the interesting behaviour of the pressure tending to $+\infty$, $-\infty$ for the right, resp., left corner.

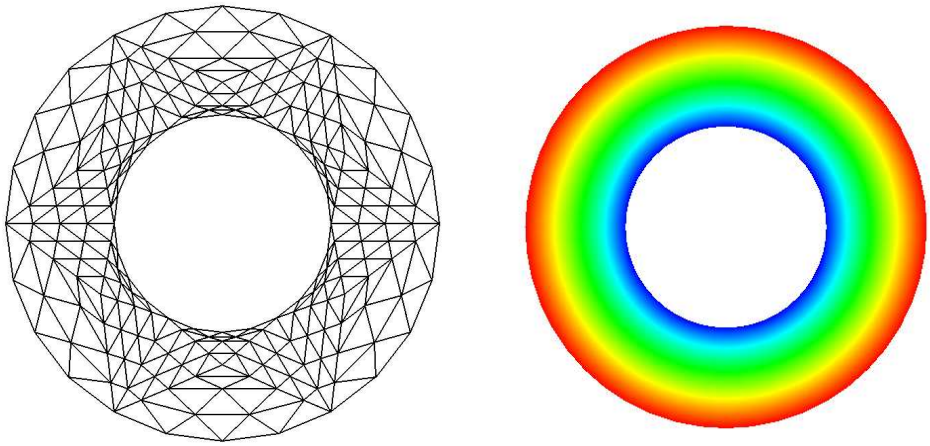


Figure A1.5: Grid and rotation velocity for *rotating couette flow*

1.2. Rotating couette flow

The setting for our second configuration is a domain bounded by two concentric circles, with inner and outer radius of $R_i = 3$, resp., $R_o = 6$. The *rotating couette flow* is driven by the outer boundary moving with a rotational speed of $U_o = \frac{1}{100R_o}$ while the inner circle is at rest, i.e. $U_i = 0$. The analytical, time-independent solution for this model is given by

$$u_{rad}(r) = \frac{1}{R_o^2 - R_i^2}\left((U_o R_o^2 - u_i R_i^2)r + (U_i - U_o)\frac{R_i^2 R_o^2}{r}\right)$$

this means it is not dependent on the viscosity ν of the simulated fluid, either. Nevertheless, we chose this model to analyse the asymptotic convergence behaviour concerning refinement level and sound speed, finding a strong influence of the parameter c on the obtained solution (see Section 7.1) As seen in Fig. A1.1 we used an unstructured mesh which has —the grid being accurately adopted to the domain — some difficult aspect ratios close to the wall, which also provided verification of our flexible finite difference space discretisation.

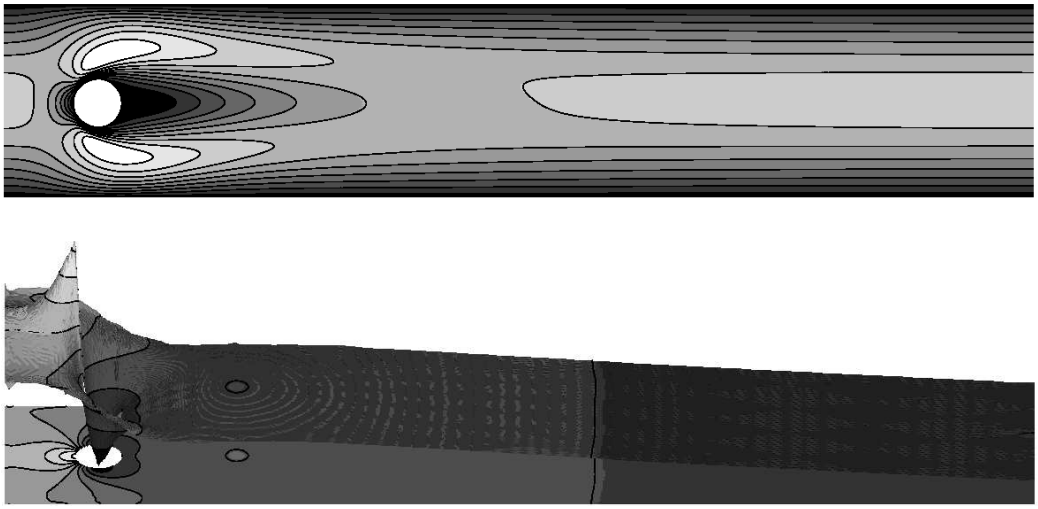


Figure A1.6: *Flow around cylinder* visualisation of u_1 and p

1.3. Flow around cylinder benchmark

The *flow around a cylinder* benchmark from [39] is generally viewed as a challenging test for any CFD tool and is modeled in the following. The flow is driven by a parabolic inflow profile in a channel of height 0.41 and length 2.4, near the entrance is fixed a circular object and free-slip boundary conditions are defined at the outflow. The Reynolds number of the flow field is defined by the cylinders diameter of $D = 0.1$, the average inflow velocity $U_0 = \frac{2}{3}u_{max}$ and the viscosity as $\mathrm{Re} = \frac{UD}{\nu}$. In Table A1.3 we present three configurations we used in our simulations, stationary flows for Reynolds numbers 2 and 20, resp., the nonstationary benchmark at Re number 100.
In course of the test one can measure forces acting on the cylinder, better results are expected using a high (local) resolution around the inner boundary. Consequently, spatial discretisations which can deal with adaptive grids are important for high computational efficiency. In general Re is chosen as 20 or 100, the latter case being in the nonstationary regime and exhibiting a periodic behaviour of the developed flow. It is relatively easy to obtain nice movies of vortex shedding behind the cylinder, but it is crucial to use higher order time — and space — discretisation to reproduce accurate numbers for the oscillating forces. In Figure A1.2 we present our locally adapted benchmark grid: To better approximate the forces acting on the circular cylinder, we have more elements gathered around the cylinder than at the channel's outflow region. This example is especially capable of showing the advantage our general FD scheme on unstructured grids has over uniform schemes which require many unknowns to obtain the needed resolution in local areas.

A2

Applications based on Chapman-Enskog analysis

In this section we will present advanced concepts developed in the thesis, applications for the DVM which are decoupled from the main discretisation, but are nevertheless important for numerical simulations. The problem with working in the Boltzmann framework is that, while in standard CFD software macroscopic conditions are quite straightforwardly implemented, here one has to use microscopic distributions and deal with approximations in the small Mach number limit. For example, in Boltzmann schemes the evaluation of forces using derivatives of velocity to obtain the viscous stress tensor is ill-conditioned. In practice, one has to evaluate the primary simulation variables to obtain accurate results.
Another example is the transformation of macroscopic moments (back) to the distribution functions, as in the case of prescribing initial conditions for the flow. Both applications are not trivial, but adequate methods were developed based on the Chapman-Enskog theory. We will relate the details and important implementation aspects, therefore the following sections should be considered after reading Section 2.4 and with an understanding of the Chapman-Enskog ansatz given therein.

2.1. Force Evaluation

In the Chapman-Enskog expansion we used the first order distribution $f^{(1)}$ as a final step towards the derivation of the Navier-Stokes equation. By this means we obtained the viscous stress tensor and in the same way — instead of using spatial derivatives of the macroscopic moments — it is possible to derive forces working on the microscopic level. We start from the approximation of the stress tensor up to second order in ε (without second viscosity):

$$S_{\alpha\beta} = \tau c_s^2 \rho_0 \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} \right) = -\varepsilon \Pi^{(1)}_{\alpha\beta} + O(\varepsilon^2)$$

To calculate the term $\Pi^{(1)}_{\alpha\beta}$ we use the identity up to first order from the series (2.14), namely

$$f^{(1)} = \frac{f - f^{(0)}}{\varepsilon} + O(\varepsilon).$$

$f^{(1)}$ is denoted as the nonequilibrium term which can be used to calculate the first order tensor

$$\Pi^{(1)}_{\alpha\beta} = \int \xi_\alpha \xi_\beta f^{(1)} = \frac{1}{\varepsilon} \int \xi_\alpha \xi_\beta (f - f^{(0)}) + O(\varepsilon)$$

The pressure tensor $P_{\alpha\beta} = p\delta_{\alpha\beta} + S_{\alpha\beta}$ used in CFD for force evaluation, is obtained in the continuous Boltzmann equation with $p = c_s^2 \rho$ as

$$\begin{aligned} P_{\alpha\beta} &= c_s^2 \rho \delta_{\alpha\beta} - \varepsilon \Pi^{(1)}_{\alpha\beta} + O(\varepsilon^2) \\ &= c_s^2 \rho \delta_{\alpha\beta} - \int \xi_\alpha \xi_\beta (f - f^{(0)}) + O(\varepsilon^2) \end{aligned}$$

while in the discrete velocity model we obtain $P_{\alpha\beta}$ by quadrature and use of the equilibrium term f_i^{eq} with overall second order compressibility error:

$$P_{\alpha\beta} = c_s^2 \rho \delta_{\alpha\beta} - \sum_i \xi_{i,\alpha} \xi_{i,\beta} (f_i - f_i^{eq}) + O(\varepsilon^2)$$

Remarks:

Remarkably, in the above force evaluation method the tensor $P_{\alpha\beta}$ is independent of ρ. This is because the contributions from the term $c_s^2 \rho \delta_{\alpha\beta}$ and the density contribution from f_i^{eq} just cancel out. For a proof it is sufficient to look only at the density appearing in

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

In the sum of P_{11} the term $\xi_{i,1}\xi_{i,1}$ results in 1 for $i = 1,2,4,5,6,8$, it means in total

$$\sum_i \xi_{i,1}\xi_{i,1}\omega_i \rho = (\frac{1}{9} + \frac{1}{36} + \frac{1}{36} + \frac{1}{9} + \frac{1}{36} + \frac{1}{36}) c^2 \rho = \frac{c^2}{3} \rho = c_s^2 \rho$$

Analogously, in P_{22} the term $\xi_{k,2}\xi_{k,2}$ results in 1 for $k = 2,3,4,6,7,8$ giving the same result. For P_{12} (and P_{21}) the term $\xi_{k,1}\xi_{k,2}$ results in 1 for $k = 2,6$ and -1 for $k = 4,8$ so the density anyway cancels out. Consequently, the forces calculated by the method (A2.1) are only determined by the distributions and macroscopic velocity, while the density has no influence as an independent variable. This explains why the resulting values for drag and lift in our tests for the *flow around cylinder* benchmark were not influenced by extrapolation of the pressure on the boundary.

2.2. Initial conditions

In the Boltzmann 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 in the form of a linear matrix, even the number of variables is not equal.
One possibility applied in practice is to set f_i equal to $f_i^{eq}(\rho, \mathbf{u})$. To obtain an accurate representation however, we have to fall back again on Chapman-Enskog theory. Skordos showed in [40] that it is possible to obtain accurate initial values from ρ and $\mathbf{u}$ by calculating the nonequilibrium part and applying the relation

$$f \sim f^{(0)} + \varepsilon f^{(1)}. \tag{A2.1}$$

The term $f_i^{(1)}$ has to be calculated from the given equilibrium in the following way. As previously, the basic step is inserting the expansion in the dimensionless Boltzmann equation. Then taking the terms of first order in ε, we obtain as previously

$$f^{(1)} = -\frac{L_r}{c_r}\left(\frac{\partial f^{(0)}}{\partial t} + \xi_\alpha \frac{\partial f^{(0)}}{\partial x_\alpha}\right).$$

In the next step and by substitution of $\varepsilon = \frac{c_r \tau}{L_r}$ we obtain

$$\begin{aligned} f^{(1)} &= -\tau \frac{L_r}{\tau c_r}\left(\frac{\partial f^{(0)}}{\partial t} + \xi_\alpha \frac{\partial f^{(0)}}{\partial x_\alpha}\right) \\ &= -\frac{\tau}{\varepsilon}\left(\frac{\partial f^{(0)}}{\partial t} + \xi_\alpha \frac{\partial f^{(0)}}{\partial x_\alpha}\right) \end{aligned}$$

Finally, we insert this identity in the series (A2.1) and use the equilibrium f_i^{eq} which results in the approximation of the discrete variables

$$f_i = f_i^{eq} - \tau\left(\frac{\partial f_i^{eq}}{\partial t} + \boldsymbol{\xi}_i \cdot \nabla f_i^{eq}\right) + O(\varepsilon^2).$$

This treatment gives in practice a gain of one order in Ma compared to identifying the distribution with the equilibrium. Again, this is not an *exact* representation of the macroscopic moments, even if it were possible to compute the gradient exactly. We have only an approximation which improves in the limit of $\mathrm{Ma} \to 0$ or $c \to \infty$.
We were able to use our finite difference discretisation of second order for the gradient and to compute $f_i^{(1)}$ for steady-state problems. The procedure can easily be extended to a full restart algorithm, on the other hand in our tests accurate initial conditions were not needed yet.

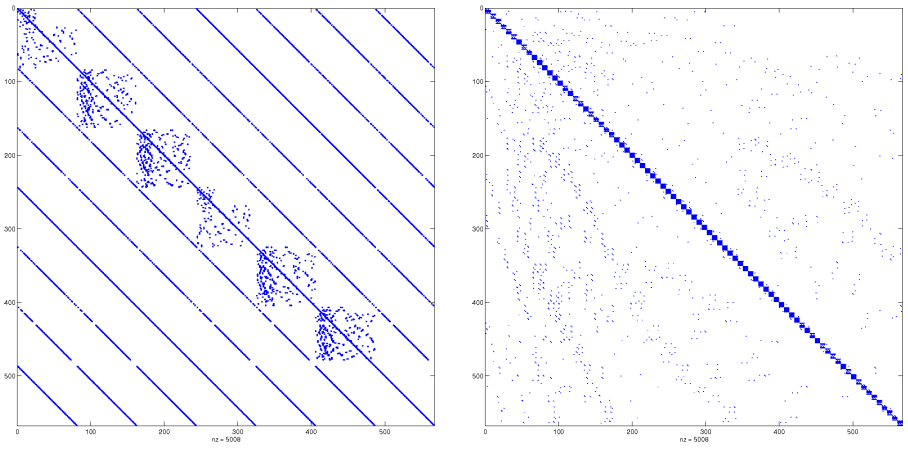
A3

D2Q7 model

In the following, we will introduce a reduced lattice set and describe the D2Q7 model. We want to apply our special discretisation techniques and monolithic solver for the 7-velocity DVM and make comparisons for efficiency and accuracy. At the same time, we will show the flexibility of our approach, not only regarding unstructured grids but also arbitrary models. Compared to the D2Q9 model, which is especially suited for Cartesian grids, the D2Q7 model has a set of lattice vectors that corresponds to a uniform triangular grid (see Fig. 2.2b). The discrete velocities include a rest particle and are defined by

$$\boldsymbol{\xi}_i = c\mathbf{e}_i = \begin{cases} (0,0)^{\mathrm{T}} & , \quad i = 0 \\ c(\cos((i-1)\pi/3, \sin((i-1)\pi/3)^{\mathrm{T}} & , \quad i = 1,\ldots,6 \end{cases}$$

This means a configuration reduced by two local variables and a matrix-size reduced by 40% compared to the standard 9-velocity model. The collisions consist only of 7×7 blocks (see Fig. A3.1). Furthermore, we have new weights ω_i resulting from quadrature restraints for f_i^{eq}, given



(a) first numbering (b) second numbering

Figure A3.1: D2Q7 matrix allocation

by the following uniform coefficients:

$$\omega_i = \begin{cases} 1/2 & , \quad \text{rest } f_0 \\ 1/12 & , \quad \text{remaining } f_i \end{cases}$$

These have to be included into the small velocity approximation of the equilibrium term, given in general form as

$$f_i^{eq} = \omega_i \rho (1 + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2}).$$

However, due to the sound speed defined in the the D2Q7 model as $c_s = c/\sqrt{4}$, we obtain for the coefficients different results (see [18]):

$$\frac{1}{c_s^2} = \begin{cases} \frac{3}{c^2} & , \quad \text{D2Q9 model} \\ \frac{4}{c^2} & , \quad \text{D2Q7 model} \end{cases} \qquad \frac{1}{2c_s^4} = \begin{cases} \frac{9}{2c^4} & , \quad \text{D2Q9 model} \\ \frac{8}{c^4} & , \quad \text{D2Q7 model} \end{cases}$$

We implemented the new (incompressible) equilibrium term as

$$f_i^{eq} = \omega_i (\rho + \rho_0 (\frac{4}{c^2}(\boldsymbol{\xi}_i \cdot \mathbf{u}) + \frac{8}{c^4}(\boldsymbol{\xi}_i \cdot \mathbf{u})^2 - \frac{2}{c^2}(u_1^2 + u_2^2)))$$

Boundary treatment

Due to the reduced number of velocities, with an angle of 60 degrees in between, we have a slightly different situation at the boundary, compared to the 9 velocity model. Looking at a rectangularly bounded domain, at south and north walls we have only 2 incoming distributions, while at east and west walls the number is still 3, but without distributions parallel to the wall. The general scheme of Ladd $f_i = f_{-i} + 2\rho_0 \cdot \omega_i \frac{\boldsymbol{\xi}_i \cdot \mathbf{u}_{bc}}{c_s^2}$ is still valid, but with modified $\omega_i = \frac{1}{12}$, $c_s^2 = \frac{c^2}{4}$ and $\boldsymbol{e}_i \in \{\binom{\pm 1}{0}, \frac{1}{2}\binom{\pm 1}{\pm\sqrt{3}}\}$ with the index $i = 1, \ldots, 6$ starting at the east cardinal point and following anticlockwise (approximately $\{E, NE, NW, W, SW, SE\}$). At the north wall for example, we get the following bounce-back scheme

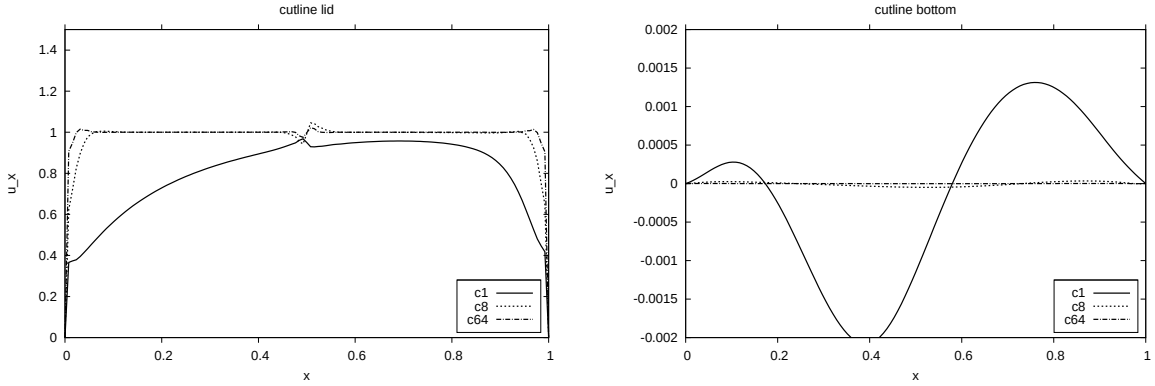
$$\begin{aligned} f_5 &= f_2 - \frac{1}{3}\frac{u_x}{c} - \frac{1}{\sqrt{3}}\frac{u_y}{c} \\ f_6 &= f_3 + \frac{1}{3}\frac{u_x}{c} - \frac{1}{\sqrt{3}}\frac{u_y}{c} \end{aligned} \tag{A3.1}$$

while Zou-He's boundary scheme is resulting in

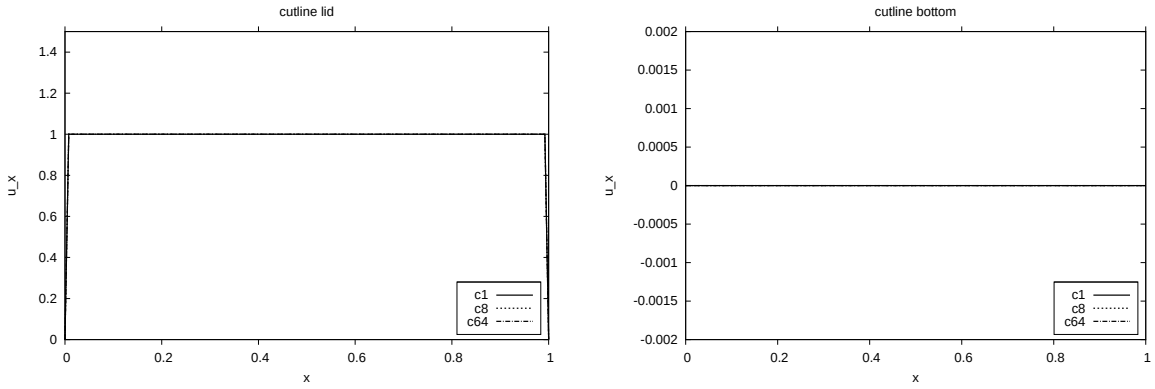
$$\begin{aligned} f_5 &= f_2 + (f_1 - f_4) - \frac{1}{3}\frac{u_x}{c} - \frac{2}{3}\frac{u_x}{c} - \frac{1}{\sqrt{3}}\frac{u_y}{c} \\ f_6 &= f_3 - (f_1 - f_4) + \frac{1}{3}\frac{u_x}{c} + \frac{2}{3}\frac{u_x}{c} - \frac{1}{\sqrt{3}}\frac{u_y}{c}. \end{aligned} \tag{A3.2}$$

One can validate scheme (A3.2) easily by summation of the distributions which results in $\sum \boldsymbol{\xi}_i f_i = (u_x, u_y)^T = \mathbf{u}_{bc}$. Numerical results (see Fig. A3.2) confirm the (locally) accurate treatment of the boundary by Zou-He's scheme. In Figure A3.2a we give the cutlines of u_x along the moving lid (north boundary with $u_x = 1$, $u_y = 0$) and the bottom (south wall with $u_x = u_y = 0$) of the cavity. In comparison, the bounce-back scheme by Ladd is not reproducing accurately the given macroscopic Dirichlet velocity of the wall. However, for increasing c the deviation from the exact values is getting significantly smaller. Also, the overall L_2-error of Ladd's scheme is comparable (even slightly better as seen in Fig. A3.2c) so one should investigate further possible advantages resulting from the advanced boundary treatment.

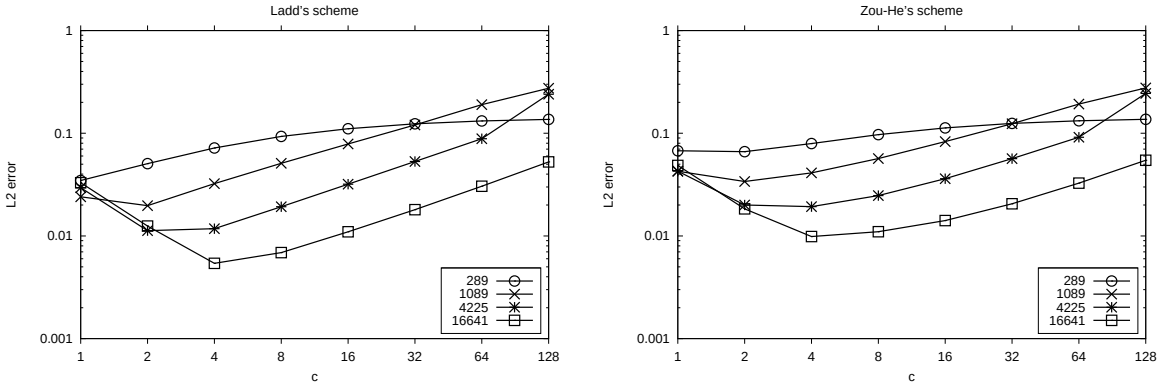
Altogether, the implementation of the 7-velocity discretisation was successfull and further tests are planned to analyse accuracy of the 'minimal' model. Especially the nonstationary *flow around cylinder* benchmark should be reproduced with accurate results for the forces at the boundary despite the few local distributions, in which case the reduced system size can lead to more efficient solvers.



(a) *Driven cavity* at Re = 100, wall u_x for Ladd's scheme



(b) *Driven cavity* at Re = 100, wall u_x for Zou-He's scheme



(c) *Driven cavity* at Re = 100, L_2-error for different boundary schemes

Figure A3.2: Results for the D2Q7 model with comparison of boundary schemes

Bibliography

[1] Bardow, A., Karlin, I.V., Gusev, A.A., *General characteristic-based algorithm for off-lattice Boltzmann simulations*, Europhys. Letters **75**, No. 3, 434–440 (2006)

[2] Bhatnagar, P.L., Gross, P.L., Krook, M., *A model for collision processes in gases*, Physical Review **94**, 511 (1954)

[3] Bouzidi, M., Firdaouss, P., Lallemand, P., *Momentum transfer of a Boltzmann-lattice fluid with boundaries*, Phys. Fluids **13**, 3452–3459 (2001)

[4] Chapman, S., Cowling, T.G., *The mathematical theory of non-uniform Gases*, Cambridge University Press (1990)

[5] Cornubert, R., d'Humières, D., Levermore, D., *A Knudsen layer theory for lattice gases*, Physica **D47**, 241–259 (1991)

[6] Dellar, P., *Incompressible limits of lattice Boltzmann equations using multiple relaxation times*, J. of Comp. Phys. **190**, 351–370 (2003)

[7] Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall International, 222–333 (1974)

[8] d'Humières, D., *Generalized lattice-Boltzmann equations*, In Rarefied Gas Dynamics: Theory and Simulations (ed. B.D. Shizgal and D.P. Weaver), Prog. Aeronaut. Astronaut. **159**, 450–458 (1992)

[9] d'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P., Luo, L.-S., *Multiple-relaxation-time lattice Boltzmann models in three-dimensions*, Philos. Trans. R. Soc. Lond. **360**, 437–451 (2002)

[10] d'Humières, D., Bouzidi, M., Lallemand, P., *Thirteen-velocity three-dimensional lattice Boltzmann model*, Phys. Rev. E **63**, 066702 (2001)

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

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

[13] Ginzburg, I., d'Humières, D., *Multireflection boundary conditions for lattice Boltzmann models*, Phys. Rev. E **68**, 066614 (2003)

[14] Guo, Z., Zhao, T. S., *Explicit finite-difference lattice Boltzmann method for curvilinear coordinates*, Phys. Rev. E **67**, 066709 (2003)

[15] Geller, S., Krafczyk, M., Tölke, J., Turek, S., Hron, J., *Benchmark computations based on Lattice-Boltzmann, Finite Element and Finite Volume Methods for laminar Flows*, Computers and Fluids **35** 8–9 (2006) 888–897

[16] He, X., Chen, S., Doolen, G., *A novel thermal model for the Lattice Boltzmann method in incompressible limit*, J. of Comp. Phys. **146**, 282–300 (1998)

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

[18] He, X., Luo, L. S., *Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation*, Phys. Rev. E **56**, 6811 (1997)

[19] Hübner, T., *Spezielle Diskretisierungs- und Lösungsmethoden für Integro-Differentialgleichungen am Beispiel der Strahlungstransportgleichung*, Diploma Thesis *www.mathematik.uni-dortmund.de/lsiii/static/showpdffile_Huebner2005.pdf*, Dortmund (2005)

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

[21] Hübner, T., Turek, S., *Efficient monolithic simulation techniques for the stationary Lattice Boltzmann equation on general meshes*, Computing and Visualization in Science **13**, No. 3, 129–143 (2010)

[22] Hübner, T., Turek, S., *Efficient monolithic Boltzmann simulations of nonstationary flow problems*, to appear (2010)

[23] Inamuro, T., Yoshino, M., Ogino, F., *Accuracy of the lattice Boltzmann method for small Knudsen number with finite Reynolds number*, Phys. Fluids **9**, 3535 (1997)

[24] Junk, M., Klar, A., *Discretisations for the Incompressible Navier-Stokes Equations based on the Lattice Boltzmann Method*, SIAM J. Sci. Comput. **22**, No. 1, 1–19 (2000)

[25] Junk, M., Klar, A., Luo, L. S., *Asymptotical analysis of the Lattice Boltzmann Equation*, J. of Comp. Phys. **210**, No. 2, 676–704 (2005)

[26] Köster, M., *Robuste Mehrgitter-Krylowraum-Techniken für FEM-Verfahren*, Diploma Thesis, Dortmund (2004)

[27] Ladd, A., *Numerical simulations of particulate suspensions via a discretised Boltzmann equation*, Journal of Fluid Mechanics **271**, 285-309 (1994)

[28] Lallemand, P., Luo, L. S., *Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability*, Phys. Rev. E **61**, No. 6, 6546–6562 (2000)

[29] Li, Y., LeBoeuf, E., Basu, P.K., *Least-squares finite-element scheme for the lattice Boltzmann method on a n unstructured mesh*, Phys. Rev. E **72**, 046711 (2005)

[30] Lee, T. Lin, C.H., *A characteristic Galerkin method for discrete Boltzmann equation*, J. of Comp. Phys. **171**, 336–356 (2001)

[31] Mavriplis, D.J., *Multigrid solution of the steady-state Lattice Boltzmann equation*, Computer and Fluids **35**, 573–591 (2006)

[32] Mei, R., Shyy, W., *On the finite difference-based lattice Boltzmann method in curvilinear Coordiates*, J. of Comp. Phys. **143**, 426–448 (1998)

[33] Mei, R., Yu, D., Shyy, W. and Luo, L.-S., *Force evaluation in the Lattice Boltzmann method involving curved geometry*, Physical Review E, **65**, 041203 (2002)

[34] Noble, D., Holdych, D., *Full Newton Lattice Boltzmann Method for Time-Steady Flows using a Direct Linear Solver*, Int. J. of Mod. Phys. C **18**, No. 4, 652–660 (2007)

[35] Ouazzi, A., *Finite Element Simulation of Nonlinear Fluids. Application to Granular Material and Powder*, Shaker, Postfach 101818, 52018 Aachen, (2006)

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

[37] Reider, M., Sterling, J., *Accuracy of discrete velocity BGK models for the simulation of the incompressible Navier Stokes equations*, Computers and Fluids **24**, 459–467 (1995)

[38] Saad, Y., Schultz, M.H., *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. and Stat. Comput. **7**, No. 3, 856–869 (1986)

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

[40] Skordos, P.A., *Initial and boundary conditions for the lattice boltzmann method*, Phys. Rev. E **48**, 4823-4842 (1993)

[41] Stiebler, M., Tölke, J., Krafczyk, M., *An Upwind Discretisation Scheme for the Finite Volume lattice Boltzmann Method*, Computers and Fluids **35**, 814–819 (2006)

[42] Tölke, J., *Gitter-Boltzmann-Verfahren zur Simulation von Zweiphasenströmungen*, Dissertation (2001)

[43] Tölke, J., Krafczyk, M., Rank, E., *A Multigrid-Solver for the Discrete Boltzmann-Equation*, J. Stat. Phys. **107**, No. 1/2, 573–591 (2002)

[44] Tölke, J., Krafczyk, M., Schulz, M., Rank, E., Berrios, R., *Implicit discretisation and non-uniform mesh refinement approaches for FD discretisations of LBGK Models*, International Journal of Modern Physics C **9**, No. 8, 1143–1157 (1998)

[45] Turek, S.: *FEATFLOW Finite Element software for the incompressible Navier-Stokes equations: User Manual (www.featflow.de)*, Release 1.2, University of Dortmund, (2000)

[46] Turek, S., *A Generalized Mean Intensity Approach for the Numerical Solution of the Radiative Transfer Equation*, Computing **54**, 27–38 (1995)

[47] Turek, S., Rivkind, L., Hron, J., Glowinski, R., *Numerical Study of a Modified Time-Stepping* θ*-Scheme for Incompressible Flow Simulations*, Journal of Scientific Computing **28** (2006) 533–547

[48] Van der Vorst, H.: *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 (1992)

[49] Wobker, H.; Turek, S.: *Numerical Studies of Vanka-type Smoothers in Computational Solid Mechanics*, Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 342, FB Mathematik, Universität Dortmund (2007)

[50] Zienkiewicz, O.C., Codina, R., *A general algorithm for compressible and incompressible flow— Part I. The split, characteristic-based scheme*, Int. J. for Num. Meth. in Fluids **20**. 869–885 (1995)

[51] Zou, Q.; He, X. *On pressure and velocity boundary conditions for the lattice Boltzmann BGK model*, Physics of Fluids **9**, No. 6, 1591–1598 (1997)