Cranfield University

Tom-Robin Teschner

Development of a Two Dimensional Fluid Solver Based on the Lattice Boltzmann Method

School of Engineering
Computational Fluid Dynamics

Msc THESIS
Academic Year 2012-2013

Supervisor: Nikolaos Asproulis
August 2013

CRANFIELD UNIVERSITY

SCHOOL OF ENGINEERING
COMPUTATIONAL FLUID DYNAMICS

MSC THESIS

ACADEMIC YEAR 2012-2013

TOM-ROBIN TESCHNER

DEVELOPMENT OF A TWO DIMENSIONAL FLUID SOLVER BASED ON THE LATTICE BOLTZMANN METHOD

SUPERVISOR: NIKOLAOS ASPROULIS
AUGUST 2013

This thesis is submitted in partial fulfilment of the requirements
for the degree of Master of Science

Abstract

The lattice Boltzmann method is a relatively new computational fluid dynamics approach. It has replaced the lattice gas cellular automata in the early 1990s which has been based on molecular dynamics. Since then, the lattice Boltzmann method has undergone a rapid growth and interest among scientists has risen fast. It offers great advantages over the Navier-Stokes equation as it is a single, linear partial differential equation which can be seen as an advection equation with a source term. Computational time is therefore reduced and it is furthermore easily parallelizable due to its locality. The main advantage is the reduction of complex physical models to simple equations. This makes the lattice Boltzmann method ideal to be used for such flows. The classic applications are multiphase flows and flows through porous media although this method has been extended by a diverse field of scientists to be used with heat transfer, reactive flow, bioengineering and quantum physics, to name only a few application areas.

A two dimensional fluid solver has been developed as part of the thesis that solvers the discrete Boltzmann equation on structured grids. The mesh reading has been done in an unstructured manner to allow further code extensions and is completely based on the CGNS format. It offers the advantage of storing not only the information of the mesh but also the flow solution, convergence history and simulation data in a binary format. The collision operator has been approximated by a single relaxation time (BGK), two relaxation time (Ginzburg) and multiple relaxation time schemes while the streaming is done using the popular D2Q9 speed model. The boundary treatment after Zou / He as well the closely related bounce back have been implemented which offers a simple and easy to understand approach. The code has been written and documented in a way which should make it easy to understand and extend.

The code validation process revealed insight into the method and its drawbacks which are little documented. Boundary conditions are prone to instabilities, especially the corner regions. Recirculation areas have been over and under predicted due to poor resolved pressure gradients normal to the primary flow direction. Curved boundaries introduced yet another possible source of errors where the pressure coefficient revealed a non-physical behaviour driven by mesh induced density fluctuations. The velocity profiles and streamlines however produced good agreement with reference data and showed that the lattice Boltzmann method is capable of simulating fluid flows. The implementation of further sophisticated models such as multi-speed models, off lattice Boltzmann method and other types of boundary conditions, to name a few, could potentially eliminate some of the currently experienced bottlenecks and increase the numerical accuracy. The true power of the lattice Boltzmann method is yet to be demonstrated by the implementation and testing of advanced physical models.

Acknowledgement

The work on this thesis has been greatly affected by friends, supporters and staff which I'd like to express my sincere appreciation to. I would like to thank *Prof. Detlev Schulze* from the Hamburg University of Applied Science for all his advices and fruitful discussions, on a professional and personal level, which greatly helped me to specialise in CFD and which ultimately resulted in applying for the MSc in CFD course at Cranfield University.
My deepest gratitude goes to *Dr. Nikolaos Asproulis* for his helpful assistant and having always his door open in times of crises. The relaxed and stress-free environment always helped greatly to stay calm and focused.
I would especially like to thank the "*Erich-Becker-Stiftung*", whose support showed that my work is recognised by an aeronautical foundation. I am greatly thankful for the support and hope to contribute to the aeronautical industry and its research through my future career steps.

I'd like to thank *Mr. André Barlow* for his helpful information exchange on the off lattice Boltzmann method. His opinion and further information beyond his publications is greatly appreciated. Furthermore I'd like to thank *Mr. Clément Carlier* for all his programming advices and answers relating to C++ problems and *Mr. Gennaro Abbruzzese* for his invaluable assistance and exchange of knowledge on the lattice Boltzmann method.

Last but not least I'd like to express my sincere love and gratitude to *Ms. Jiyoung Kim* for her support and motivation throughout the year in Cranfield and to my parents *Ms. Monica Teschner* and *Mr. Hans-Günther* who have always supported me in all my life choices for which I will be forever indebted.

Preface

Thy style of this thesis is in accordance with "The Oxford Guide to Style[1]" and the spelling with the Oxford dictionary.
The mathematical notation has been mostly followed and only altered for better legibility. Vectors and matrices are printed in bold and scalar quantities not. Variables are printed in italic but mathematical expressions are not. The internal energy for example is e while the euler number is e. The same is true for the differential, for example $\mathrm{d}t$.
Expressions that are abbreviated are written out the first time they are being used with their abbreviation given in parenthesis. Only the abbreviation will be used from then on but all abbreviations are given in the List of abbreviations.
The bibliography style is based on the Institute of Electrical and Electronics Engineers (IEEE), a format widely used engineering publication which follows the numbering system. An attempt has been made to cite all of the relevant literature and not to dismiss them as common knowledge although their results are well known, like the Reynolds number published by Osborne Reynolds in 1895. As a result, articles as old as 157 years have been tracked down and cited which should give the reader a useful source of references to start their own literature review.

[1]R.M. Ritter, *The Oxford Guide to Style.* Oxford University Press, 2 ed., 2002.

Contents

List of Figures

List of Tables

List of Symbols

Latin Letters

Symbol	**Definition**	**Unit**
A	Area	[m^2]
A_s	specific surface area	[m^2]
$\mathbf{a}$	Acceleration vector	[m s^{-2}]
$\mathbf{c}$	Particle velocity vector	[m s^{-1}]
C_p	Pressure coefficient	[1]
C_s	Coefficient in the Smagorinsky model	[1]
c_s	Speed of sound in lattice units	[m/s]
c_x, c_y, c_z	Particle velocity components	[m s^{-1}]
d_c	Diameter (channel)	[m]
E	Total energy	[J]
e	Internal energy	[J]
$\mathbf{F}$	Force vector	[N]
f	Distribution function	[1]
$\widehat{f}$	Filtered distribution function	[1]
f^a	Antisymmetric part of the distribution function	[1]
f^{eq}	Local distribution function at equilibrium	[1]
f^s	Symmetric part of the distribution function	[1]
K	Permeability	[1]
K_0	Effective permeability	[1]
k_B	Boltzmann constant	[J K^{-1}]
l	Length	[m]
l_c	Characteristic length	[m]
l_d	Development length (channel)	[m]
l_s	Lattice spacing	[m]
l_p	Physical length scale	[m]
m	Mass	[kg]
$\boldsymbol{m_v}$	Momentum vector	[kg m s^{-1}]
N	Amount of particles	[1]
$N_{lattice}$	Number of points along lattice	[1]
N_A	Avogadro number	[1]
n	Amount of particles per unit volume	[1]
$\mathcal{O}$	Truncation error	[1]
p	Pressure	[Pa]
R	Gas constant	[J K^{-1} mol^{-1}]
R_i	Residual	[1]
$\boldsymbol{\mathcal{S}}$	Relaxation matrix	[1]
r	Position vector	[m]
r_s	Sphere radius	[m]
$S_{i,j}$	Strain rate	[s^{-1}]
$\overline{S}_{i,j}$	Filtered strain rate	[s^{-1}]
T	Temperature	[K]
$\boldsymbol{\mathcal{T}}$	Transformation matrix	[1]
t	Time	[s]
$\mathbf{u}$	Fluid velocity vector	[m s^{-1}]
$\overline{\boldsymbol{u}}$	Mean velocity vector	[m/s]
$\boldsymbol{u}'$	Fluctuating velocity vector	[m/s]
u_c	Characteristic velocity	[m s^{-1}]

u_x, u_y, u_z	Fluid velocity components	[m s^{-1}]
V	Volume	[m^3]
$\dot{V}$	Volume flow rate per unit area	[m^3 s−1 m^{-2}]
$V_{kk'}$	Interaction pseudo potential between particles	[1]
w_i	Weighting factors	[1]
x, y, z	Cartesian Coordinates	[m]

Greek Letters

Symbol	**Definition**	**Unit**
α	Constant for distribution function	[1]
β	Constant for distribution function	[1]
Γ_k	Coefficients for equilibrium distribution function	[1]
γ_k	Specific heat ratio	[1]
Δ	Cut-off filter width	[m]
ε	Knudsen number	[1]
ε_D	Dissipation	[m^2 s^{-3}]
ζ_k	Energy level	[1]
η_i	Specific heat ratio control variable	[1]
θ	Domain	[1]
θ_c	Radiant around cylinder	[rad]
λ	Mean free path	[m]
μ	Kinematic viscosity	[m^2 s^{-1}]
ν	Dynamic viscosity	[kg m^{-1} s^{-1}]
ν_e	Effective (kinetic) viscosity	[kg m^{-1} s^{-1}]
ν_{SGS}	Sub grid scale (kinetic) viscosity	[kg m^{-1} s^{-1}]
$\nu_{lattice}$	Dynamic viscosity	[kg m^{-1} s^{-1}]
ν_T	Turbulent (kinematic) viscosity	[kg m^{-1} s^{-1}]
ξ	Surface tension parameter	[1]
$\mathbf{\Pi}$	Colour gradient	[1]
ρ	Density	[kg m^{-3}]
ρ'	Fluctuating density	[kg m^{-3}]
$\widehat{\rho}$	Instantaneous density	[kg m^{-3}]
Φ	Velocity correlation function	[1]
φ	Arbitrary variable	[1]
τ	Relaxation factor	[s]
$\widehat{\tau}$	Modified collision time	[s]
τ_a	Relaxation factor, antisymmetric part	[s]
τ_s	Relaxation factor, symmetric part	[s]
τ_{SGS}	Collision time at sub grid scale	[s]
χ_s	Solid fraction	[1]
$\mathbf{\Psi}$	Collision matrix in velocity space	[1]
$\widehat{\mathbf{\Psi}}$	Collision matrix in momentum space	[1]
Ω	Collision operator	[1]
ω	Collision frequency	[Hz]
ω_a	Collision frequency, antisymmetric part	[Hz]
ω_s	Collision frequency, symmetric part	[Hz]

Abbreviations

Abbreviation	**Definition**
API	Application programming interface
BGK	Bhatnagar, Gross, Krook (model)
BGKW	Bhatnagar, Gross, Krook, Welander (model)
CFL	Courant Friedrichs Lewy (number)
CGNS	CFD general notation system
DNS	Direct numerical simulation
FHP	Frisch, Hasslacher, Pomeau (model)
GCI	Grid convrgence index
HPP	Hardy, Pomeau, de Pazzis (model)
IBM	Immersed boundary method
KE	Kinetic Energy
LBGK	Lattice Bhatnagar, Gross, Krook (model)
LBE	Lattice Boltzmann equation
LBM	Lattice Boltzmann method
LES	Large eddy simulation
LGA	Lattice Gas Automata
LGCA	Lattice Gas Cellular Automata
MHD	Magnetohydrodynamics
MRT	Multiple relaxation time (scheme)
MSE	Mean square error
NSE	Navier-Stokes Equations
RANS	Reynolds averaged Navier Stokes
SGS	Sub grid scale
SIDS	Standard interface data structure
SRT	Single relaxation time (scheme)
TRT	Two relaxation time (scheme)
TUI	Text user interface
QLBM	Quantum lattice Boltzmann method

1 Introduction

The Lattice Boltzmann method (LBM) was introduced to bridge the gap between microscopic and macroscopic fluid flows. It uses a distribution function which is evaluated at each so-called site connected by links, where the streaming (advection) and collision (momentum transfer) are evaluated at discrete time steps. This method offers several advantages over the conventional Navier-Stokes equations, which are commonly used for solving fluid flows that are discussed. This section elucidates the main idea of the LBM. The collision operator Ω will be introduced and several ways shown how to approximate this operator by single and multi-relaxation time schemes.

1.1 Objectives of the Thesis

The objective of this Master thesis is to develop a fluid solver capable of simulating two dimensional flows using the lattice Boltzmann method. The code will be developed in C++.

There are two main objectives that should be achieved:

1. The solver has to be able to read and process unstructured meshes.
2. The solver has to be validated against benchmark cases.

All results are to be obtained using the Lattice-Boltzmann method. The solver must be verified and validated against experimental and numerical data. Furthermore, analytical solutions will be utilized where available. The benchmark cases should provide confidence in the results obtained by the solver. The first validation case should be a channel flow inside the laminar regime to compare the analytical velocity profile against the solver's velocity profile. The second validation case should be the lid driven cavity to compare the velocity profiles against available literature. The third validation case should be the backward facing step where the reattachment length should be investigated. Beyond that, further validation cases can be used to further demonstrate the functionality of the code.

1.2 Kinetic theory

For all subsequent sections, a basic knowledge of kinetic theory is needed which is revised in the following.
Figure 1 shows particles of mass m_k, moving in different directions x, y and z with the corresponding velocity components c_x, c_y and c_z inside the domain of θ. Newton's second law states the conservation of momentum as:

$$\boldsymbol{F} = \frac{\mathrm{d}(m_k \boldsymbol{c})}{\mathrm{d}t} \tag{1.1}$$

It is assumed that inter-particle collisions do not take place and that each particle moves at the opposite direction after colliding with the boundary θ. Then, the momentum difference for the collision with the domain can be expressed for one particle as:

$$\boldsymbol{F}\mathrm{d}t = m_{k,1}\left(\sqrt{c_x^2+c_y^2+c_z^2}\right) - \left[-m_{k,2}\left(\sqrt{c_x^2+c_y^2+c_z^2}\right)\right] = \left[m_{k,1}+m_{k,2}\right]\sqrt{c_x^2+c_y^2+c_z^2} \quad (1.2)$$

Here, the lower scripts indicate the state before $(m_{k,1})$ and after $(m_{k,2})$ the collision. The second minus sign accounts for the change in direction. The time in between collision is depending upon the speed of the particle and its distance it moves from one side of the domain to the other and back again. It can be expressed as:

$$\mathrm{d}t = \frac{2l}{\sqrt{c_x^2+c_y^2+c_z^2}} \quad (1.3)$$

Inserting eq.(1.3) into eq.(1.2) yields:

$$\begin{aligned}\frac{2\boldsymbol{F}l}{\sqrt{c_x^2+c_y^2+c_z^2}} &= \left[m_{k,1}+m_{k,2}\right]\sqrt{c_x^2+c_y^2+c_z^2} \\ \boldsymbol{F} &= \frac{\left[m_{k,1}+m_{k,2}\right]\left(c_x^2+c_y^2+c_z^2\right)}{2l} \quad (1.4)\end{aligned}$$

A perfect elastic collision is assumed, therefore $m_{k,1}+m_{k,2}=2m_k$. Furthermore, isotropic velocity distribution is assumed, hence a global velocity component, equal in all directions as $c^2 = 3c_x^2 =$

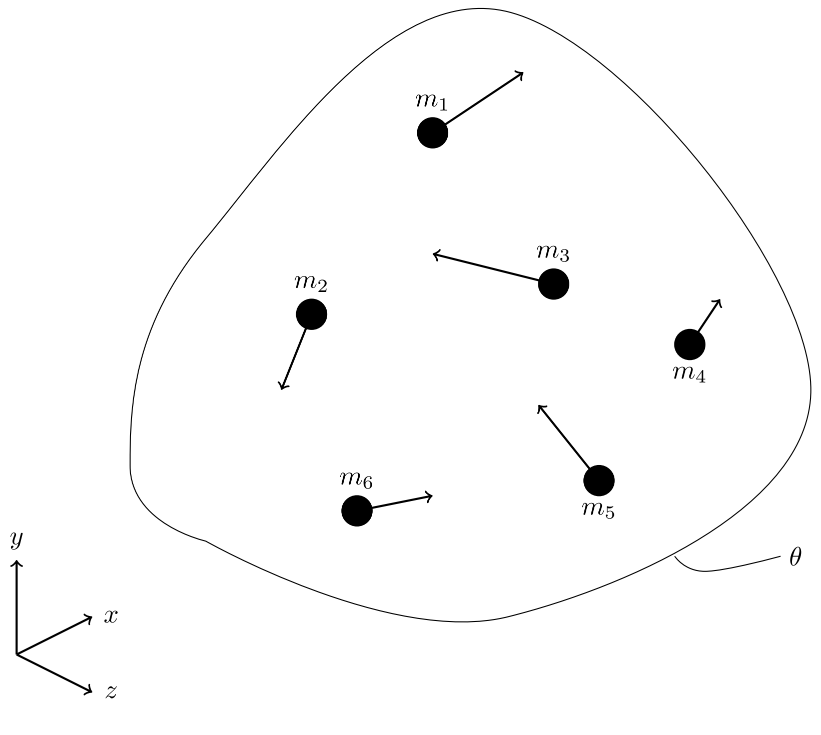


Figure 1 – Particles of mass m_k moving inside the domain of θ.

$3c_y^2 = 3c_z^2 = c^2/3$ can be introduced. Inserting into eq.(1.4):

$$\boldsymbol{F} = \frac{2m_k(c^2/3)}{2l} = \frac{m_k c^2}{3l} \tag{1.5}$$

The pressure is equal to a normal force per area. Inserting into eq.(1.5) and taking into account that a length times area is identical to volume, i.e. $lA = V$, the following form is obtained:

$$p = \frac{m_k c^2}{3lA} = \frac{m_k c^2}{3V} \tag{1.6}$$

Eq.(1.6) is the pressure exerted by one particle. To generalize this idea, N particles can be evaluated as:

$$p = \frac{m_k c^2}{3V} N \tag{1.7}$$

Rewriting the above equation yields:

$$p = \frac{m_k c^2}{3V} N = \frac{m_k c^2}{3V} \frac{2}{2} N = \frac{m_k c^2}{2} \frac{2N}{3V} = \frac{2}{3} nKE \tag{1.8}$$

where n is the number of particles per unit volume and KE[2] the kinetic energy. Eq.(1.8) can be inserted into the equation of state:

$$\frac{m_k c^2}{2} \frac{2N}{3V} V = nRT = \frac{N}{N_A} RT \tag{1.9}$$

where N_A is the Avogadro number. The Boltzmann constant is introduced as $k_B = R/N_A = 1.38 \cdot 10^{-23}\ J\ K^{-1}$. Simplifying eq.(1.9) results in:

$$T = \frac{2}{3} \frac{KE}{k_B} \tag{1.10}$$

Equation (1.8) and (1.10) show that pressure and temperature are scaled values of the kinetic energy. It is not surprising since the macroscopic value of pressure is related to the microscopic collision of particles per unit area. An increase in collision is achieved by increasing the particles velocity (i.e. kinetic energy). This is done by an increased internal energy. Internal energy is increased due to heating, so more velocity (or kinetic energy) of a particle will result at higher temperature at constant pressure.

[2]The integral value of the kinetic energy should be used rather than an averaged value for physical correctness. For simplicity however, the averaged value is herein used.

1.3 Lattice gas cellular automata

Lattice Gas Cellular Automata (LGCA) was developed before the LBM. It makes use of cellular automata, which is a rectangular, two-dimensional grid were at each node (called *site*) the state *true* or *false* can exist. In terms of fluid dynamics, this means that a particle either *can* or *cannot* exist at a node at a given time. The LGCA (sometimes also referred to as Lattice Gas automata, LGA) exists of two steps, collision and streaming, which are calculated in this exact order. This can be expressed as:

$$N(\boldsymbol{r} + \boldsymbol{c}\mathrm{d}t, t + \mathrm{d}t) - N(\boldsymbol{r}, t) = \Omega \tag{1.11}$$

The above equation states that the difference of particles N between the spatial range of $(\boldsymbol{r} + \boldsymbol{c}\mathrm{d}t)$ and the time range of $(t + \mathrm{d}t)$ with respect to the amount of particles at $(\boldsymbol{r}, t)$ is equal to the collision occurring at the site. Here, Ω is the collision operator. If no collision occurs, Ω is zero and thus the net difference of particles as well (conservation of the particle amount). Different models have been developed over the years for Ω. The first model, known as the HPP (Hardy, Pomeau, de Pazzis) model was introduced by the same authors in 1973 and 1976 [1,2]. An improved model was later proposed by Frisch, Hasslacher and Pomeau, which was introduced in 1986 [3] and is named after the authors (FHP model). Once the collision is obtained, the streaming process is performed, where each particle travels at a unit velocity. This means, that all particles are moving at the same speed and from one site to another site per time step. Since the position of each particle is exactly known due to the Boolean description at each site, no round-off errors are introduced. Furthermore, the *exclusion* principle is applied, which does not allow particles to travel on the same *link* (which is the connection of two sites) in the same direction at the same time step. This, however, does not include the streaming of two particles on the same link in opposite directions. In that situation, the particles pass each other without colliding.

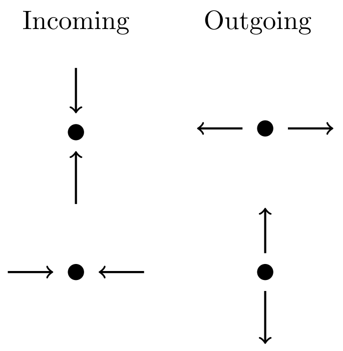


Figure 2 – Possible outgoing directions for the HPP model after collision. Filled, black circle represents a site while arrows represent particles arriving at a site on a link.

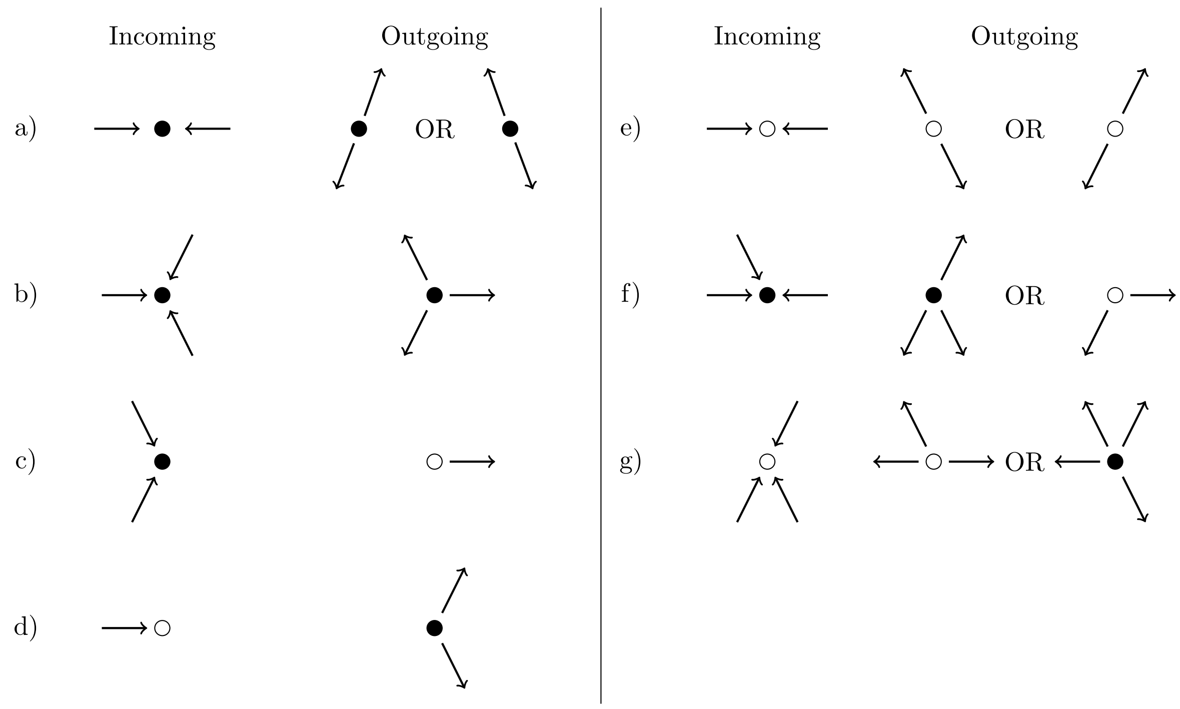


Figure 3 – Possible outgoing directions for the FHP model after collision. Filled, black circle represents a site while arrows represent particles arriving at a site on a link. Particles at rest are indicated by empty circles.

1.3.1 HPP model

The HPP model was the first model developed to predict the outgoing directions of colliding particles. This model utilizes a rectangular grid (hence, collision occurs always under a 90° angle), where each site is connected by 4 links. This means that a maximum of four particles can enter and leave the site per time step. Figure 2 shows possible incoming and outgoing combinations. If two particle approach a node perpendicularly, (e.g. from south and west), they will not collide and stream in the same direction as in the previous time step. Conservation of momentum can be easily seen from Figure 2.

1.3.2 FHP model

The FHP model is an improvement over the HPP model. In contrast to the former, it uses a hexagonal structure for the lattice (hence, collision occurs under multiples of 60° angles), i.e. one site has six links. Each site can have 6-7 particles coming in and leaving at the same time, which is achieved by particles resting at a site. Figure 3 shows possible incoming and outgoing combinations. Usually, the FHP model is divided into the FHP-I, FHP-II and FHP III model, each model offering more collision possibilities then the former one. While the FHP-I model only allows for collision option a) and b) to take place, the FHP-II and FHP-III allow particles to rest at a site (indicated by empty circles) i.e. allowing collisions of a) through e) and a) through g), respectively. By rotating the collision options a) to g) by 60°, all possible collisions are obtained.

As for the HPP model, conservation of momentum can be easily seen from Figure 3.

1.3.3 Remarks on lattice gas cellular automata

The LGA has several advantages as discussed, though it also comes with disadvantages, which are inherent in the approach. The HPP model is highly anisotropic due to its collision rules and rectangular grid. It was shown by Frisch *et al.* [4], that viscous dissipation occurs indeed anisotropically. Likewise, it lacks rotational invariance, which means that vortices occur rectangularly. Although rotational flows are better resolved in the FHP model over the HPP model, it also introduces further problems, one of the main problems being that it is not Galilean invariant, i.e. two states are possible (collision possibility a), e), f) and g) in Figure 3).
When macroscopic quantities such as density or pressure are obtained from each site, statistical noise causes inaccurate results. It has become good practice to average such quantities over a greater region to remove noise. Further noise reduction can be achieved by ensemble averaging.
Two-dimensional flows may be extended in three dimensions. If so, rectangular grids have to be used, which cause problems as seen in the HPP model. Dodecahedron and Icosahedron are the only suitable elements which can be used beside the cube which increases the complexity of the lattice structure. Here, each site has to account for 12 and 20 links, respectively, which further could account for particles resting at the site. The possible collisions at each site are therefore high and computational resources may restrict the usage of these elements. Further information on how to expand the LGA method to three dimensions can be found in [5, p.102].

1.4 The distribution function

James Clerk Maxwell realized in 1859, that even with all the formulas available to him, tracing the path of each particle was an impossible task. Instead of tracing each particle individually, Maxwell introduced the idea of a distribution function for the velocity. The exact velocity at each location of the particles is not known anymore and cannot be retrieved from the distribution function. Rather, it is stating the likelihood of how many particles are within a velocity range of $\boldsymbol{c} + \mathrm{d}\boldsymbol{c}$. Per definition, the integrated value of the distribution function is equal to unity, hence the range of $\boldsymbol{c} + \mathrm{d}\boldsymbol{c}$ represents a fraction of all particles. Mathematically speaking, the distribution function f for all three velocity components c_x, c_y and c_z can be expressed as:

$$\iiint f(c_x)f(c_y)f(c_z)\mathrm{d}c_x\mathrm{d}c_y\mathrm{d}c_z = 1 \tag{1.12}$$

Eq.(1.12) does not depend upon the space coordinates x, y and z but rather on the velocity component c_x, c_y and c_z instead. Thus, an unknown correlation function Φ can be introduced as:

$$f(c_x)f(c_y)f(c_z) = \Phi\left(c_x^2 + c_y^2 + c_z^2\right) \tag{1.13}$$

which combines the distribution function with the velocity components. Functions that satisfy eq.(1.13) are logarithmic or exponential in nature. Maxwell used an exponential approach and postulated:

$$f(c_i) = \alpha \mathrm{e}^{-\beta c_i^2} \tag{1.14}$$

This can be summed for all c_i and yields the distribution function in three-dimensional space as:

$$\sum f(c_i) = \alpha \mathrm{e}^{-\beta c_x^2} \alpha \mathrm{e}^{-\beta c_y^2} \alpha \mathrm{e}^{-\beta c_z^2} = \alpha^3 \mathrm{e}^{-\beta \boldsymbol{c}^2} = f(c) \tag{1.15}$$

Eq.(1.15) does not depend on the spatial coordinates as mentioned before. Hence, to visualize eq.(1.15), a three-dimensional space can be introduced where the velocity components c_x, c_y and c_z are assigned to the x, y and z axis. Then, a sphere would indicate a region of constant velocity magnitude, i.e. $|\boldsymbol{c}| = constant$. See Figure 4.
Since the distribution function is only capable of calculating fractions of particles within a velocity range of $\boldsymbol{c} + \mathrm{d}\boldsymbol{c}$, eq.(1.15) is only valid for same velocity ranges, hence, it can just be applied to calculate the fractions of particles inside the spheres of radii $r_{s,1} = \boldsymbol{c}$ and $r_{s,2} = \boldsymbol{c} + \mathrm{d}\boldsymbol{c}$. The volume of a spherical shell is calculated as $4\pi \boldsymbol{c}^2 \mathrm{d}c$, therefore, the distribution function with respect to the particle speed is obtained as:

$$f(\boldsymbol{c})\mathrm{d}\boldsymbol{c} = 4\pi \boldsymbol{c}^2 \alpha^3 \mathrm{e}^{-\beta \boldsymbol{c}^2} \mathrm{d}\boldsymbol{c} \tag{1.16}$$

Integrating the above equation over all velocity ranges (i.e. from zero to infinity) results in the

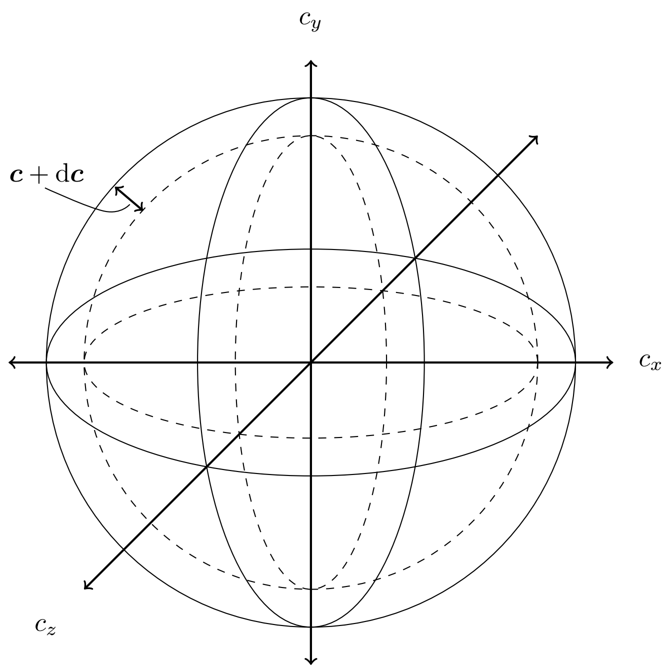


Figure 4 – Velocity space showing a sphere that indicates an ISO surface of velocity magnitude.

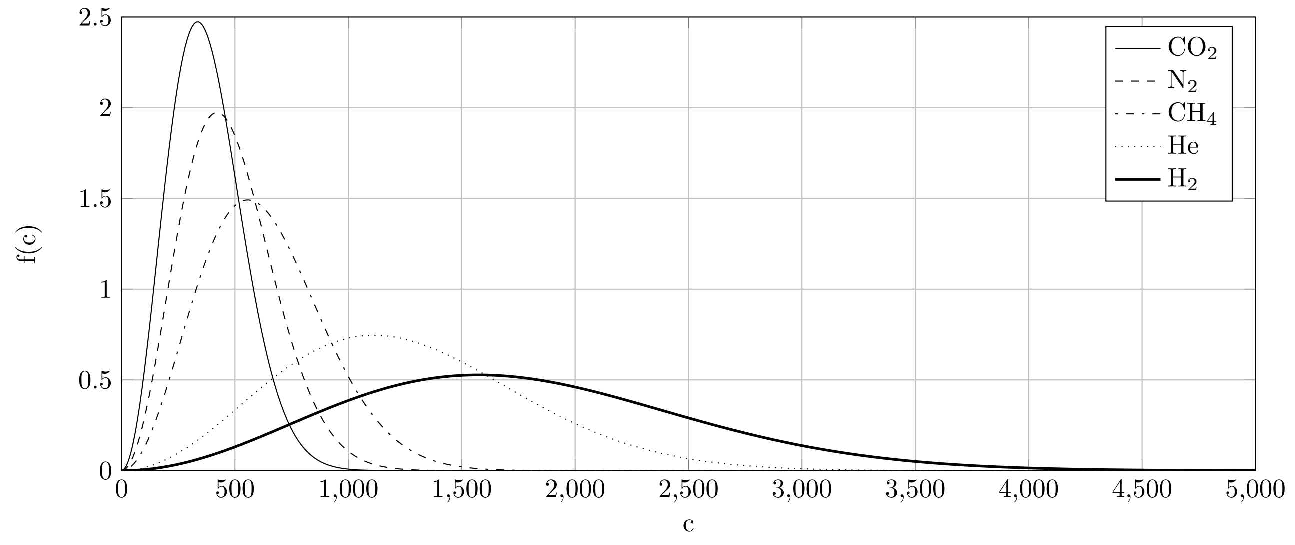


Figure 5 – Velocity distribution for CO_2, N_2, CH_4, He and H_2 at room temperature, per thousand particles.

Maxwell or Maxwell-Boltzmann distribution function as:

$$f(\boldsymbol{c}) = 4\pi \left(\frac{m}{2\pi k_B T}\right)^{\frac{3}{2}} \boldsymbol{c}^2 \mathrm{e}^{\frac{-m\boldsymbol{c}^2}{2k_B T}} \tag{1.17}$$

Eq.(1.17) is depending on the molar mass, temperature and velocity. The velocity distribution function is obtained for different chemical substances (different molar masses) at a specified temperature. Figure 5 shows the velocity distribution function for Carbon Dioxide (CO_2, 44g/mol), Nitrogen(N_2, 28g/mol), Methane (CH_4, 16g/mol), Helium (He, 4g/mol) and Hydrogen (H_2, 2g/-mol) at room temperature (T = 298.15K). The maximum velocity increases for a decrease in molar mass. This is because of Newton's first law, less molar mass means less inertia to overcome. Consequently, lighter particles travel at higher velocities.

Once the distribution function has been obtained, absolute values for maximum, averaged and root mean averaged speed can be obtained since these values are often of more interest. They are obtained as:

Maximum speed:

$$c_{max} = \sqrt{\frac{2k_B T}{m}} \tag{1.18}$$

Average speed:

$$c_{mean} = \sqrt{\frac{8k_BT}{m}} \tag{1.19}$$

Root mean average speed:

$$c_{rmas} = \frac{3k_BT}{m} \tag{1.20}$$

1.5 Distribution function at equilibrium

As will be shown in section 2, the distribution function at equilibrium is needed in order to approximate the collision operator. It is derived from a normalized Maxwell Boltzmann distribution function and given as:

$$f = \frac{3\rho}{2\pi}\mathrm{e}^{\frac{3}{2}(\boldsymbol{c}-\boldsymbol{u})^2} = \frac{3\rho}{2\pi}\mathrm{e}^{\frac{3}{2}c^2}\mathrm{e}^{\frac{3}{2}(\boldsymbol{c}\cdot\boldsymbol{u}-u^2)} \tag{1.21}$$

The Taylor series expansion around e^x is:

$$\mathrm{e}^x = 1 + x + \frac{x^2}{2!} + ... + \frac{x^n}{n!} \tag{1.22}$$

Inserting eq.(1.22) into eq.(1.21) for a truncation error of order $\mathcal{O}(u^3)$ yields:

$$f = \frac{3\rho}{2\pi}\mathrm{e}^{\frac{3}{2}c^2}\left[1 + 3(\boldsymbol{c}\cdot\boldsymbol{u}) - \frac{3}{2}u^2 + \frac{9}{2}(\boldsymbol{c}\cdot\boldsymbol{u})^2 + \mathcal{O}(u^3)\right] \tag{1.23}$$

Eq.(1.23) can be generalized for any variable φ which then becomes the universal form of the equilibrium distribution function as:

$$f^{eq} = \varphi w_i\left[\Gamma_1 + \Gamma_2(c_i\cdot\boldsymbol{u}) + \Gamma_3(c_i\cdot\boldsymbol{u})^2 + \Gamma_4 u^2 + \mathcal{O}(u^3)\right] \tag{1.24}$$

where w_i are the corresponding weights for the speed model. Eq.(1.24) is the general equilibrium distribution function where the coefficients of Γ_k need to be adapted for a given flow type. For a general, incompressible flow the coefficients of Γ_k are as follows:

$$\Gamma_1 = 1 \quad \Gamma_2 = \tfrac{1}{c_s^2} \quad \Gamma_3 = \tfrac{1}{2c_s^4} \quad \Gamma_4 = -\tfrac{1}{2c_s^2} \tag{1.25}$$

where c_s^2 is the speed of sound in lattice units $c_s = 1/\sqrt{3}$. Other coefficients of Γ_k can be derived for convection, convection-diffusion and diffusion flows. The coefficients for Γ_k given in eq.(1.25) however are universal and can be used for all kind flows types. Inserting eq.(1.25) into eq.(1.24) yields:

$$f^{eq} = \varphi w_i \left[1 + \frac{c_i \cdot \boldsymbol{u}}{c_s^2} + \frac{(c_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{u^2}{2c_s^2} \right] \tag{1.26}$$

1.6 The lattice Boltzmann method

At this point, the main concept of the Lattice-Gas Cellular Automata and the Maxwell distribution function has been introduced. The driving idea behind the LBM is the combination of both approaches. The particles at each site are replaced by the distribution function. Accordingly, the LBM is neither microscopic (particles are replaced by the distribution function) nor macroscopic (macroscopic quantities must still be obtained) but rather referred to as mesoscopic (greek: *mesos*=middle). The transport equation for the LBM will be devised in the following.

1.6.1 From lattice gas cellular automata to lattice Boltzmann method

Eq.(1.11) states that the amount of particles is changed, if a collision occurs. The collision itself can be described by using either the HPP or FHP model, though variations of these models exist and may be used. The LBM uses a similar ansatz function which later will be confirmed by transforming the transport equation of the LBM into the Navier-Stokes equation, see section 1.6.2. For the LBM, eq.(1.11) is complemented by a force acting on the particles which cause a change within the velocity range and the number of particles is replaced by the distribution function. In addition, the equation is evaluated for an infinitesimal small region of space and velocity, thus

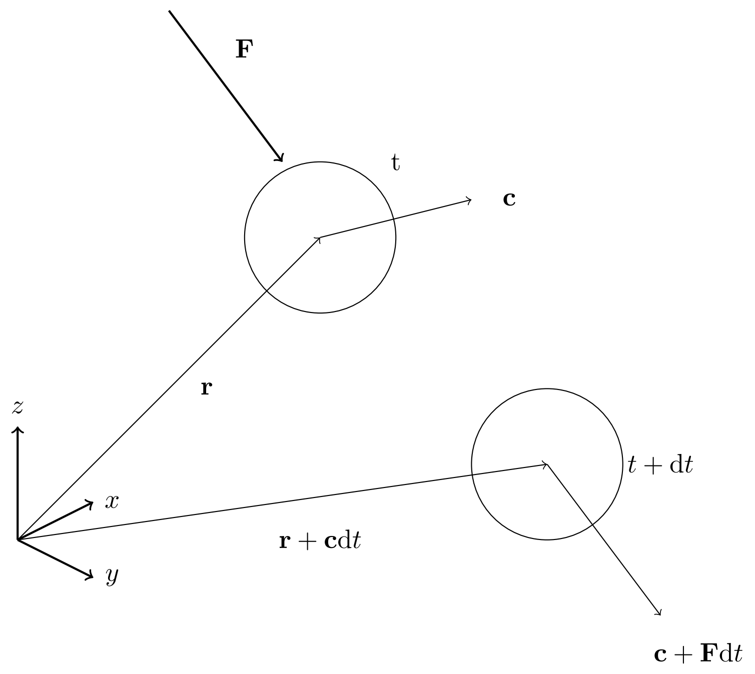

$\mathbf{F}$
t
$\mathbf{c}$
$\mathbf{r}$
z
x
y
$t + \mathrm{d}t$
$\mathbf{r} + \mathbf{c}\mathrm{d}t$
$\mathbf{c} + \mathbf{F}\mathrm{d}t$

Figure 6 – External force acting on a particle. Corresponding direction change over time is shown.

resulting in:

$$f(\boldsymbol{r}+\boldsymbol{c}\mathrm{d}t,\boldsymbol{c}+\boldsymbol{F}\mathrm{d}t,t+\mathrm{d}t)\mathrm{d}\boldsymbol{c}\mathrm{d}\boldsymbol{r}-f(\boldsymbol{r},\boldsymbol{c},t)\mathrm{d}\boldsymbol{r}\mathrm{d}\boldsymbol{c}=0 \tag{1.27}$$

Here, no collision takes place. The particles have an initial state at $(\boldsymbol{r},\boldsymbol{c},t)$ which is altered by applying an external force $\boldsymbol{F}$. This is shown in Figure 6. To account for collision, the collision operator, previously introduced as Ω can be introduced into eq.(1.27). The collision will change the velocity distribution function within the range of $\mathrm{d}\boldsymbol{c}\mathrm{d}\boldsymbol{r}$. Additionally, the collision operator alters the distribution function in the interval dt since it is the rate of change before and after the collision, hence the equation becomes:

$$f(\boldsymbol{r}+\boldsymbol{c}\mathrm{d}t,\boldsymbol{c}+\boldsymbol{F}\mathrm{d}t,t+\mathrm{d}t)\mathrm{d}\boldsymbol{c}\mathrm{d}\boldsymbol{r}-f(\boldsymbol{r},\boldsymbol{c},t)\mathrm{d}\boldsymbol{r}\mathrm{d}\boldsymbol{c}=\Omega(f)\mathrm{d}\boldsymbol{r}\mathrm{d}\boldsymbol{c}\mathrm{d}t \tag{1.28}$$

Eq.(1.28) divided by its differentials as the time tends to zero yields:

$$\frac{\mathrm{d}f}{\mathrm{d}t}=\Omega(f) \tag{1.29}$$

Since the distribution function f is still depending on space, velocity and time, the chain rule for eq.(1.29) yields:

$$\frac{\mathrm{d}f}{\mathrm{d}t}=\frac{\partial f}{\partial\boldsymbol{r}}\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t}+\frac{\partial f}{\partial\boldsymbol{c}}\frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}t}+\frac{\partial f}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}t}=\frac{\partial f}{\partial\boldsymbol{r}}\boldsymbol{c}+\frac{\partial f}{\partial\boldsymbol{c}}\boldsymbol{a}+\frac{\partial f}{\partial t}=\frac{\partial f}{\partial\boldsymbol{r}}\boldsymbol{c}+\frac{\partial f}{\partial\boldsymbol{c}}\frac{\boldsymbol{F}}{m}+\frac{\partial f}{\partial t} \tag{1.30}$$

Inserting into eq.(1.29) yields:

$$\frac{\partial f}{\partial\boldsymbol{r}}\boldsymbol{c}+\frac{\partial f}{\partial\boldsymbol{c}}\frac{\boldsymbol{F}}{m}+\frac{\partial f}{\partial t}=\Omega(f) \tag{1.31}$$

Under the assumption that no external forces are applied and simplifying the equation yields the final form of the LBM as:

$$\frac{\partial f}{\partial t}+\boldsymbol{c}\cdot\nabla f=\Omega(f) \tag{1.32}$$

This is also referred to as the transport equation for the LBM. Eq.(1.32) has the shape of a linear advection equation with a source term. Although the source term, here the collision operator Ω, is depending on the distribution function f and hence is difficult to obtain, in contrast to the classical Navier-Stokes equations (NSE), the non-linear advection term in the momentum equation has vanished. Furthermore, the LBM consists of only one transport equation while the NSE correspond to a system of partial differential equations. This makes the LBM not only computationally more efficient but also easier to parallelize. The underlying physical models in the LBM are greatly simplified and so is the mathematical representation in form of a simplified transport equation in contrast to the NSE. Multiphase flows are a great beneficiary of the LBM and commonly solved by this method since the Poisson equation does not need to be evaluated to obtain the pressure field which can take up to 80% of the computational time for NSE [6] and additionally, in LBM, no surface tracking is required. For more information see section 3.1.

1.6.2 From Lattice Boltzmann to Navier Stokes

The transformation of the LBE into the Navier Stokes equations is a cumbersome and mathematical challenging exercise and some physical intuition is needed to make appropriate assumptions. This section shows how to obtain the continuity equation of the NSE. The momentum equation is not derived because it is acquired in the same manner but requires a lot more algebra. While the LBE consists of one single equation, the NSE consist of one continuity, n momentum equations (n being the dimensions) and one energy equation. In order to obtain all NSE, the Chapman-Enskog expansion is used, which expands an arbitrary variable φ similar to a Taylor series expansion using a smallness parameter. It is named after Sydney Chapman and David Enskog who introduced this method independently in 1916 [7] and 1917 [8]. In the context of LBM, the smallness parameter is usually chosen to be the Knudsen number $\varepsilon = \lambda/l_p$ with λ being the mean free path and l_p representing a physical length scale. This parameter is then raised to a power thus separating the summands by orders of magnitudes, formally written as:

$$\varphi = \varphi^{(0)} + \varepsilon^1\varphi^{(1)} + \varepsilon^2\varphi^{(2)} + \ldots + \varepsilon^n\varphi^{(n)} \tag{1.33}$$

A Chapman-Enskog analysis is then used to investigate the different summands at different smallness parameter. Each of the investigated equations will yield one part of the NSE. For obtaining these equations, the Lattice Boltzmann evolution equation is used rather than the LBE and is defined as:

$$\Omega_i(\boldsymbol{r}, t) = f_i(\boldsymbol{r} + c_i, t + 1) - f_i(\boldsymbol{r}, t) \tag{1.34}$$

Eq.(1.34) is called the Boltzmann evolution equation and is similar to eq.(1.11) where N is replaced by f_i and dt is replaced by one since a lattice time scale is employed, i.e. particles can only be or not be at a site but never on a link at time t. Hence, t can only be a natural number. A Taylor series expansion in time around eq.(1.34) yields:

$$\begin{aligned}\Omega_i(\boldsymbol{r}, t) &= f_i(\boldsymbol{r} + c_i, t + 1) - f_i(\boldsymbol{r}, t) \\ &\approx f_i\left(\frac{\partial}{\partial t} + \nabla \cdot \boldsymbol{c}\right) + \frac{1}{2} f_i\left(\frac{\partial^2}{\partial t^2} + 2\frac{\partial}{\partial t}\nabla \cdot \boldsymbol{c} + \nabla\nabla : c_i c_i\right)\end{aligned} \tag{1.35}$$

Now two time scales t_1 and t_2 are introduced where $t_1 < t_2$. These time scales can be associated with a physical process such as advection and diffusion. Since physical phenomena occurring in diffusion are usually much faster than those occurring in advection processes, the statement above suggests that the time scale of diffusion is associated with t_1 and advection with t_2 respectively. By applying some physical intuition, the following expansions can be defined using the Chapman-Enskog expansion:

$$\frac{\partial}{\partial t} = \varepsilon \frac{\partial}{\partial t_1} + \varepsilon^2 \frac{\partial}{\partial t_2} + \mathcal{O}\left(\varepsilon^3\right) \tag{1.36}$$

$$\nabla = \varepsilon \nabla_1 + \mathcal{O}\left(\varepsilon^2\right) \tag{1.37}$$

$$f_i = f_i^0 + \varepsilon f_i^1 + \mathcal{O}\left(\varepsilon^2\right) \tag{1.38}$$

$$\Omega_i = \Omega_i^{(0)} + \varepsilon \Omega_i^{(1)} + \varepsilon^2 \Omega_i^{(2)} + \mathcal{O}\left(\varepsilon^3\right) \tag{1.39}$$

Here, $f_i^{(0)} = f_i^{eq}$ and is written in that way for legibility. Inserting eq.(1.36) to (1.39) into eq.(1.35) yields:

$$\begin{aligned} \Omega_i &= \left(\varepsilon \frac{\partial}{\partial t_1} + \varepsilon^2 \frac{\partial}{\partial t_2} + \varepsilon \nabla_1 \cdot c_i + \frac{\varepsilon^2}{2} \frac{\partial}{\partial t_1} + \varepsilon^2 \frac{\partial}{\partial t_1} \nabla_1 \cdot c_i + \frac{\varepsilon^2}{2} \nabla_1 \nabla_1 : c_i c_i\right) \left(f_i^{(0)} + \varepsilon f_i^{(0)}\right) \\ &+ \mathcal{O}\left(\varepsilon^3\right) \end{aligned} \tag{1.40}$$

Rearranging eq.(1.40) in accordance with the Knudsen number the equation becomes:

$$\begin{aligned} \Omega_i &= \varepsilon \left[\left(\frac{\partial}{\partial t_1} + \nabla_1 \cdot c_i\right) f_i^{(0)}\right] \\ &+ \varepsilon^2 \left[\left(\frac{\partial}{\partial t_1} + \nabla_1 \cdot c_i\right) f_i^{(1)} + \left(\frac{\partial}{\partial t_2} + \frac{1}{2} \frac{\partial^2}{\partial t_1^2} + \frac{\partial}{\partial t_1} \nabla_1 \cdot c_i + \frac{1}{2} \nabla_1 \nabla_1 : c_i c_i\right) f_i^{(0)}\right] \\ &+ \mathcal{O}\left(\varepsilon^3\right) \\ &= \Omega_i^{(0)} + \varepsilon \Omega_i^{(1)} + \varepsilon^2 \Omega_i^{(2)} + \mathcal{O}\left(\varepsilon^3\right) \end{aligned} \tag{1.41}$$

At this point, eq.(1.41) can be split into two different equations based on the Knudsen number ε and ε^2 and the method of comparing coefficients. Thus:

$$\Omega_i^{(1)} = \left(\frac{\partial}{\partial t_1} + \nabla_1 \cdot c_i\right) f_i^{(0)} \tag{1.42}$$

$$\Omega_i^{(2)} = \left(\frac{\partial}{\partial t_1} + \nabla_1 \cdot c_i\right) f_i^{(1)} + \left(\frac{\partial}{\partial t_2} + \frac{1}{2}\frac{\partial^2}{\partial t_1^2} + \frac{\partial}{\partial t_1}\nabla_1 \cdot c_i + \frac{1}{2}\nabla_1\nabla_1 : c_i c_i\right) f_i^{(0)} \tag{1.43}$$

Eq.(1.42) and eq.(1.43) do indeed represent the continuity and momentum equation of the NSE, although that is not evident. Additional definitions are needed and given as:

$$\sum \Omega_i = 0 \tag{1.44}$$

$$\rho = \sum f_i \tag{1.45}$$

$$\rho \boldsymbol{u} = \sum c_i f_i \tag{1.46}$$

where eq.(1.45) and eq.(1.46) are the discrete form of eq.(2.20) and eq.(2.23). Eq.(1.44) can be derived from the L collision operator but a rather physical explanation, more explanatory would be that the sum of all collisions over a control volume θ has to be conserved and thus must be zero. Another, more theoretical explanation would be to think of the LBE, i.e. eq.(1.32), as an advection equation with a source term as mentioned earlier. This means, if the sum of Ω_i would not be zero, then it would act as a sink or source for particles. To visualize that, this means that particles could be created or destroyed during a collision process which would violate the conservation of mass[3]. Eq.(1.45) and eq.(1.46) are almost self-explanatory. The sum of f_i yields the macroscopic density while the sum over all particle speeds c_i will result in the macroscopic velocity, denoted by $\boldsymbol{u}$. Inserting eq.(1.44) to (1.46) into eq.(1.42):

$$\begin{aligned} \sum \Omega_i &= \frac{\partial}{\partial t_1}\sum f_i + \nabla_1 \cdot \sum c_i f_i && (1.47) \\ 0 &= \frac{\partial}{\partial t}\rho + \nabla \cdot \rho \boldsymbol{u} && (1.48) \end{aligned}$$

Eq.(1.48) takes the well-known form of the continuity equation. The derivations can be done in the same manner for eq.(1.43) yielding the momentum equation. A good derivation of the full NSE can be found in [9–12].

This concludes the first chapter but before proceeding, the most important points discussed shall be reiterated. By using kinetic energy, the macroscopic effects of pressure and temperature have been derived from the microscopic molecular motion of particles. It has been shown that both quantities are scaled values of the kinetic energy. Using the idea of moving particles, the LGCA

[3]unless two smaller particles are created from one greater particle or on greater particle is transformed into two smaller particles. Atomic effects are, however, not considered.

method was derived, which is the predecessor of the LBM. Advantages and disadvantages have been outlined, which resulted in the development of the LBM. To bridge the gap between LGCA and the LBM, the distribution function f has been introduced, which replaced the idea of single particles by a representative distribution function based on the Maxwell-Boltzmann distribution function. The LBM is said to be a mesoscopic approach in between the continuum approach and molecular dynamics. Using the Chapman-Enskog expansion and multi-scale (different time scales) analysis, the continuity equation of the NSE has been recovered and steps outlined on how to recover the momentum equation in a similar manner. In other words, this chapter showed how to derive the governing equation of fluid dynamics in macroscopic space (NSE) from the molecular motion of single moving particles.

2 Numerical lattice Boltzmann method

Bringing the LBM into computational space means that eq.(1.32) needs to be discretized. With it there are some numerical aspects that need to be introduced in order to simulate the LBE. First of all the collision operator needs to be approximated. This is done with the concept of the distribution function at equilibrium, introduced in section 1.5. There have been a variety of models developed over the years of which the most common one will be discussed. As for the NSE, the LBE needs to be discritized. This however is done in a complete different way compared to the NSE since the LBM is a local approach which does not require the approximation of derivatives. Instead, speed models are used which describe, how the streaming process distributes the particles over time. The classic speed models propagate the information to their neighbour nodes. More advanced speed models make use of nodes beyond the direct neighbour, spanning over several lattices and are therefore referred to as multi-speed models. Since the propagation is done through derivatives in the NSE and through speed models (no derivatives) in the LBM, it is not possible to give an estimate of accuracy for the LBM, since this is solely determined by the discretization scheme.
Several models have been developed to treat the boundary conditions and a brief overview is given as the treatment is completely different from that known from the NSE. The last part deals with macroscopic variables and how they are obtained in the LBM. The sections are covered in the same order a fluid solver for the LBM would work. First it lets all distribution functions collide while the streaming process propagates this information into the domain. When it encounters a boundary, the streaming has to be treated differently. The last step obtains the macroscopic quantities and the whole process starts again.

2.1 Collision approximation

As outlined before, the collision operator Ω is depending on the distribution function. Due to its complexity, an approximation is used for calculations. Different models exist of which the most common one will be discussed in the following.

2.1.1 Single-relaxation time scheme

The most popular approximation for the collision operator was introduced in 1954 by Bhatnagar, Gross and Krook [13] and is classified as a single-relaxation time scheme. It is named after their authors as the BGK model, sometimes also referred to as the Lattice BGK, or LBGK model. In 1954, Welander [14] found the same solution independently, thus the name BGKW is sometimes used throughout the literature. From now on, this collision approximation will be addresses as the LBGK model. It is expressed as:

$$\Omega(f) = \omega\left(f^{eq} - f\right) = \frac{1}{\tau}\left(f^{eq} - f\right) \tag{2.1}$$

where f^{eq} is the local equilibrium distribution function as in eq.(1.24). ω can be expressed using the relaxation factor τ. The rate of change is indicated by the collision frequency or relaxation factor, respectively. The inside of the parenthesis give the net difference of particles after the collision. f^{eq} is only depending on space, velocity and time, as discussed previously. f however is the actual distribution function present on a site. If there is only one distribution function "arriving" at a site, no collision will take place since $f^{eq} = f$. If more distribution functions are present on a site, "arriving" from different links, then $f^{eq} \neq f$ since f is now a combination of all distribution functions. It follows that there will be a net difference and thus a collision, which is exactly what eq.(2.1) states verbally.

2.1.2 Two-relaxation time scheme

Although the LBGK model is well capable of simulating complex flows, it is a known drawback for this model that as the viscosity decreases, the numerical instability increases. Ginzburg [15] introduced a revised collision operator which is based on the LBGK approximation to overcome some numerical instabilities. The distribution function in eq.(2.1) is replaced by a symmetric part f^s and an anti symmetric part f^a. The collision operator Ω is recovered as a linear combination of both parts as:

$$\Omega(f) = \frac{1}{\tau_s}\left(f^s + f^{s,eq}\right) + \frac{1}{\tau_a}\left(f^a + f^{a,eq}\right) \tag{2.2}$$

The relaxation factor changes accordingly to τ_s and τ_a to account for the symmetry and anti symmetry. If $\tau_s = \tau_a$, then the LBGK approximation is obtained. Due to its dual relaxation factor, it is referred to as a two-relaxation time scheme. The definition for the symmetric part f^s and anti symmetric part f^a is:

$$f^s = \frac{1}{2}\left(f + \widehat{f}\right) \tag{2.3}$$

and

$$f^a = \frac{1}{2}\left(f - \widehat{f}\right) \tag{2.4}$$

where $\widehat{f}$ is the distribution function traveling in the opposite direction of f. Eq.(2.3) and eq.(2.4) are the average of the opposed distribution functions f and $\widehat{f}$. The relaxation factor can be expressed in terms of collision frequency ω as in the LBGK model as $\omega_a = 1/\tau_a$ and $\omega_s = 1/\tau_s$. A correlation between the collision frequency and the kinematic viscosity ν exists as:

$$\omega_s = \frac{1}{3\nu + 0.5} \tag{2.5}$$

and

$$\omega_a = \frac{8(2-\omega_s)}{8-\omega_s} \tag{2.6}$$

2.1.3 Multi-relaxation time scheme

It is possible to further generalize the concept of relaxation time schemes. For a general description, the collision operator Ω is replaced by the collision matrix $\boldsymbol{\Psi}$, then eq.(1.28) becomes:

$$f(\boldsymbol{r}+\boldsymbol{c}\mathrm{d}t, \boldsymbol{c}+\boldsymbol{F}\mathrm{d}t, t+\mathrm{d}t) - f(\boldsymbol{r},\boldsymbol{c},t) = \boldsymbol{\Psi}\left[f^{eq}(\boldsymbol{r},\boldsymbol{c},t) - f(\boldsymbol{r},\boldsymbol{c},t)\right] \tag{2.7}$$

The right-hand side of eq.(2.7) resembles the LBGK model. The collision frequency is now replaced by the collision matrix, which has the shape of a square matrix. During a perfect elastic collision, the momentum is conserved. Hence, eq.(2.7) needs to be transformed into the momentum space which make the general description of the collision process easier. Introducing an appropriate transformation matrix $\boldsymbol{\mathcal{T}}$ which is constructed from the discrete velocities [16], the distribution function f can be replaced by the momentum vector $\boldsymbol{m_v}$ and the collision matrix $\boldsymbol{\Psi}$ becomes the collision matrix in momentum space, denoted by $\widehat{\boldsymbol{\Psi}}$. Using $\boldsymbol{\mathcal{T}}$, $\boldsymbol{m_v}$ and $\widehat{\boldsymbol{\Psi}}$ are obtained as:

$$\boldsymbol{m_v} = \boldsymbol{\mathcal{T}} f \tag{2.8}$$

and

$$\widehat{\boldsymbol{\Psi}} = \boldsymbol{\mathcal{T}}\boldsymbol{\Psi}\boldsymbol{\mathcal{T}}^{-1} = \boldsymbol{\mathcal{T}}^{-1}\boldsymbol{\mathcal{S}} \tag{2.9}$$

where $\boldsymbol{\mathcal{S}}$ is the relaxation matrix which is diagonal in shape. Inserting eq.(2.8) and eq.(2.9) into eq.(2.7) yields:

$$f(\boldsymbol{r}+\boldsymbol{c}\mathrm{d}t, \boldsymbol{c}+\boldsymbol{F}\mathrm{d}t, t+\mathrm{d}t) - f(\boldsymbol{r},\boldsymbol{c},t) = \boldsymbol{\mathcal{T}}^{-1}\boldsymbol{\mathcal{S}}\left[\boldsymbol{m_v^{eq}}(\boldsymbol{r},\boldsymbol{c},t) - \boldsymbol{m_v}(\boldsymbol{r},\boldsymbol{c},t)\right] \tag{2.10}$$

At this point, the model cannot be further simplified as not only $\boldsymbol{\mathcal{T}}$ depends on the speed model used and thus upon its discrete velocities but $\boldsymbol{\mathcal{S}}$ as well. Since the speed model determines the dimensions of the matrices, this model is dynamic in nature and thus referred to as a multi-relaxation time scheme. Further information can be found in [16–24].

2.2 Lattice arrangements

The LBM is solved on a computational grid just as any other conventional approach using the Navier-Stokes equations on a mesh based method. The word lattice is just used for historical reasons within the framework of the LBM. There are three possible dimensions with different

Table 2 – Velocity components and weighting factors for the $D1Q2$ and $D1Q3$ lattice arrangement.

	$D1Q2$		$D1Q3$	
site	c_i	w_i	c_i	w_i
0	-	-	0	4/6
1	1	1/2	1	1/6
2	-1	1/2	-1	1/6

lattice arrangements, which are commonly referred to as speed models and expressed in $DnQm$[4] notation. The capital D stands dimension, accordingly n determines the corresponding dimension. Q indicates how many neighboring sites are connected via links to the given site where m is the number of links. Particles stream on links from site to site with the so-called lattice speed which is similar to the LGCA method. However, rather than referring to one particle as in LGCA method, in LBM the sum of particles is expressed by the distribution function which then travels representatively for all particles from one site to the next site.

2.2.1 1D cases

There are two possibilities to construct a one-dimensional lattice arrangement. They are called $D1Q2$ and $D1Q3$ and are shown in Figure 7. It is possible to construct higher-order schemes that involve more sites such as the $D1Q5$ model which are however not considered in this overview. In order for the particles to travel from one site to the other within one time step, the length between two sites has to be equal to the time step, i.e. $\mathrm{d}t = \mathrm{d}x$. The velocity component c_i and weighting factors w_i for both models are given in Table 2. The only difference between these two schemes is that the $D1Q3$ scheme allows for one particles at rest at the given site.

2.2.2 2D cases

Expanding the lattice arrangement to two dimensions yields three possible schemes shown in Figure 8 and correspond to $D2Q4$, $D2Q5$ and $D2Q9$. Particles can be at rest for the $D2Q5$ and $D2Q9$ arrangement. Similar to the one-dimensional case, the velocity component and weighting factor can be found for all three schemes and are tabulated in Table 3.

2.2.3 3D cases

In a similar way, the lattice arrangement can be further expanded to three dimensions, yielding the $D3Q15$ and $D3Q19$ model as two possibilities. The arrangement can be seen in Figure 9.

[4]not to be confused with the particles per unit volume n and mass m

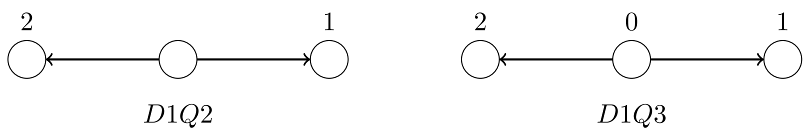


Figure 7 – $D1Q2$ and $D1Q3$ lattice arrangement shown for one dimensional cases.

Table 3 – Velocity components and weighting factors for the $D2Q4$, $D2Q5$ and $D2Q9$ lattice arrangement.

	$D2Q4$		$D2Q5$		$D2Q9$	
site	c_i	w_i	c_i	w_i	c_i	w_i
0	-	-	(0,0)	2/6	(0,0)	4/9
1	(0,1)	1/4	(0,1)	1/6	(0,1)	1/9
2	(1,0)	1/4	(1,0)	1/6	(1,1)	1/36
3	(0,-1)	1/4	(0,-1)	1/6	(1,0)	1/9
4	(-1,0)	1/4	(-1,0)	1/6	(1,-1)	1/36
5	-	-	-	-	(0,-1)	1/9
6	-	-	-	-	(-1,-1)	1/36
7	-	-	-	-	(-1,0)	1/9
8	-	-	-	-	(-1,1)	1/36

Furthermore, the velocity components and weighting factors are given in Table 4. It is possible to construct a $D3Q27$ speed model which would be essentially a combination of the $D3Q15$ and $D3Q19$ model, not counting overlapping lattices.

2.2.4 Remarks on speed models

The above described speed models are all classical speed models. As an initial starting point they are good to discretize eq.(1.32) but for more accurate simulations, multi-speed models need to be employed. The discussion of such is left to the available literature as it is a broad topic with no generalization possible. A good starting point is the article of Meng and Zhang [25] which discusses 2D speed models and gives all the needed information on weights and speeds. Further literature on 2D multi-speed models can be found in [26–28].

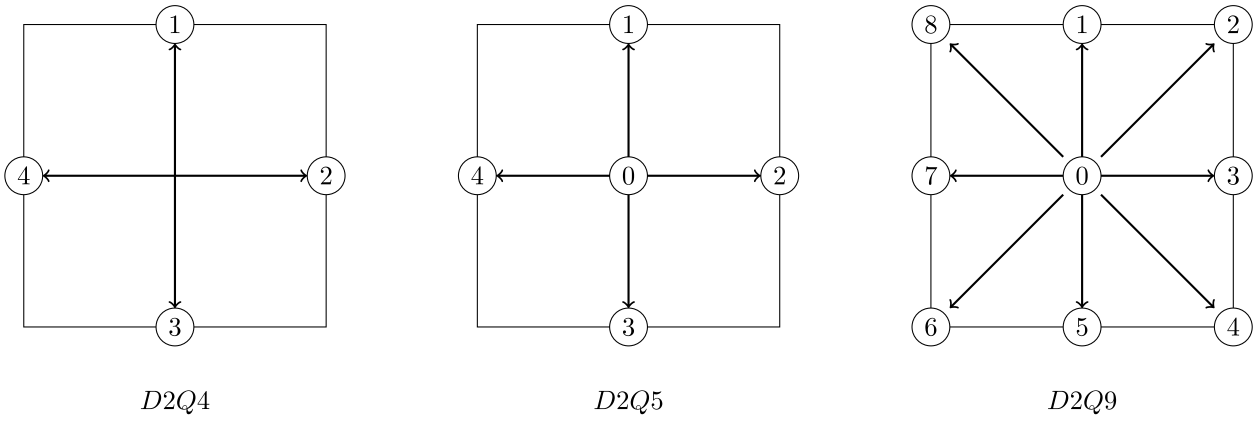


Figure 8 – $D2Q4$, $D2Q5$ and $D2Q9$ lattice arrangement shown for two dimensional cases.

Table 4 – Velocity components and weighting factors for the $D3Q15$ and $D3Q19$ lattice arrangement.

	$D3Q15$		$D3Q19$	
site	c_i	w_i	c_i	w_i
0	(0,0,0)	16/72	(0,0,0)	12/36
1	(1,0,0)	8/72	(1,0,0)	2/36
2	(0,1,0)	8/72	(0,1,0)	2/36
3	(-1,0,0)	8/72	(-1,0,0)	2/36
4	(0,-1,0)	8/72	(0,-1,0)	2/36
5	(0,0,1)	8/72	(0,0,1)	2/36
6	(0,0,-1)	8/72	(0,0,-1)	2/36
7	(1,1,1)	1/72	(1,1,0)	1/36
8	(1,1,-1)	1/72	(-1,1,0)	1/36
9	(1,-1,-1)	1/72	(-1,-1,0)	1/36
10	(1,-1,1)	1/72	(1,-1,0)	1/36
11	(-1,1,-1)	1/72	(1,0,1)	1/36
12	(-1,1,1)	1/72	(1,0,-1)	1/36
13	(-1,-1,1)	1/72	(-1,0,-1)	1/36
14	(-1,-1,-1)	1/72	(-1,0,1)	1/36
15	-	-	(0,1,1)	1/36
16	-	-	(0,1,-1)	1/36
17	-	-	(0,-1,-1)	1/36
18	-	-	(0,1,-1)	1/36

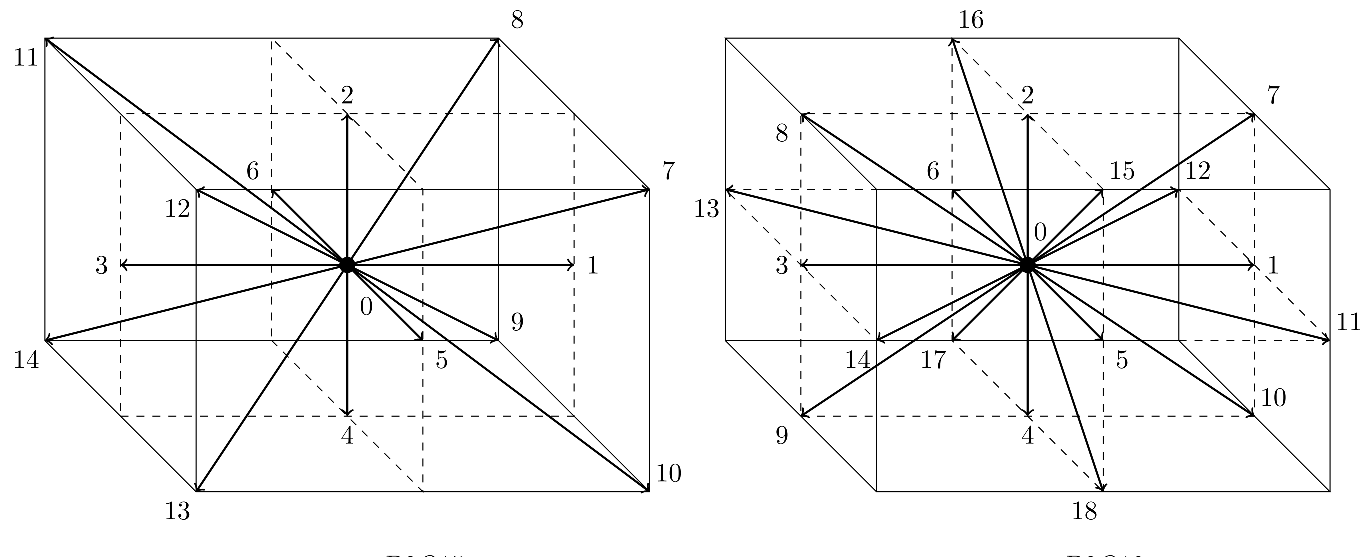


Figure 9 – $D3Q15$ and $D3Q19$ lattice arrangement shown for three dimensional cases.

2.3 Boundary treatment

As in any other approach solving the governing equations of fluid dynamics, the LBM is no different and treating the boundaries correctly is crucial to obtain the right results. The main concept of boundary treatment is summarized in the next two sections 2.3.1 and 2.3.2. While the bounce back explains the main concept of how boundaries must be treated, the boundary treatment after Zou and He gives a more physical representation but is inherently build on the bounce back.

2.3.1 Bounce Back

In contrast to the NSE, the distribution function is the only known property during the calculation. Although the macroscopic values of density and velocity are needed during calculation for the equilibrium distribution function, eq.(1.26), they are only derived quantities fromt he distribution function and not independently calculated. This means that the distribution function is the only physical quantity present at the boundary and needs to be treated accordingly.
The bounce back does simply what the name states; it bounces the distribution function back to where it comes from. Since no elastic work is taken into account, conservation of momentum is automatically given. Following the lattice arrangement in Figure 8 for the D2Q9, the bounce back at the south boundary for example can be seen in Figure 10.
The distribution functions outside of the domain are shown as dashed lines. Against intuition, the distribution function does not bounce back according to its angle that it arrives at the wall but rather bounces back in the exact same direction where it comes from. For the south boundary, this means that the following distribution functions are equal after updating the boundary conditions:

$$\begin{aligned} f_1 &= f_5 \\ f_2 &= f_6 \\ f_8 &= f_4 \end{aligned} \tag{2.11}$$

It is evident that this boundary treatment can only be applied at no slip walls, for inlet, outlet and moving walls, a more sophisticated boundary treatment is needed to account for moving fluid.

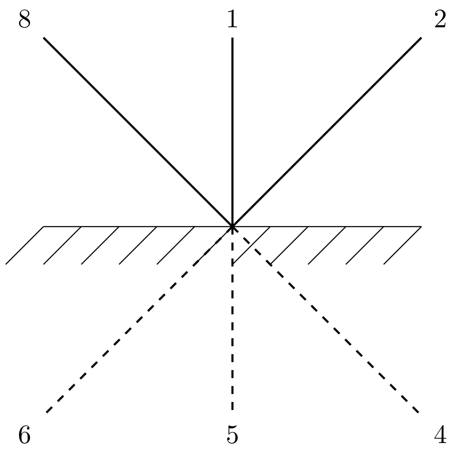


Figure 10 – Bounce back schematically shown at the south boundary.

2.3.2 Zou / He

Zou and He [29] extended the concept of the bounce back and applied their methodology to the LBGK model. This boundary treatment is however universal and can be used with any collision operator and in one, two and three dimensions. This makes this model very popular as it is very flexible, though not the best choice for all situations, more accurate models have been developed (see [30–40]) which however are more complicated to implement or lack the universal description.

The Zou/He model is best understood by expanding eq.(2.20) and eq. (2.23) from the next section. For simplicity, this is only done in two dimensions and applied to the south boundary. A complete overview for all north, east, south and west boundaries can be found in [18, p.75-81]. The lattice numbering is again the same as in Figure 8 and the missing distribution function f_3 and f_7 in Figure 10 are parallel and on top of the wall.

Expanding eq.(2.20) and eq. (2.23) yields:

$$\sum f(\boldsymbol{r},t) = \rho(\boldsymbol{r},t) = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 \tag{2.12}$$

$$\sum c(\boldsymbol{r},t)f(\boldsymbol{r},t) = \rho(\boldsymbol{r},t)u_x(\boldsymbol{r},t) = f_2 + f_3 + f_4 - f_6 - f_7 - f_8 \tag{2.13}$$

$$\sum c(\boldsymbol{r},t)f(\boldsymbol{r},t) = \rho(\boldsymbol{r},t)u_y(\boldsymbol{r},t) = f_8 + f_1 + f_2 - f_4 - f_5 - f_6 \tag{2.14}$$

The velocity components are known at the boundaries, i.e. for stationary walls they are zero, for moving walls and inlets they have a user defined value while the outlet velocity has to be extrapolated from the fluid domain. This leaves three equations for four unknowns, namely density and the three distribution functions f_8, f_1 and f_2 pointing into the domain. The simple bounce back cannot be applied since the velocity is now present. Zou and He constructed a fourth equation, relating the non-equilibrium to each other. They stated that

$$f_1 - f_1^{eq} = f_5 - f^{eq} \tag{2.15}$$

has to be satisfied at the boundaries. In words, the non-equilibrium (defined as $f_i - f_i^{eq}$) at the boundaries must be equal. Introducing the equilibrium distribution function into the problem, the system of equations can be solved for the unknowns. Inserting the equilibrium distribution function, eq.(1.26), into eq.(2.15) yields:

$$f_1 = f_5 + \frac{2}{3}\rho(\boldsymbol{r},t)u_y(\boldsymbol{r},t) \tag{2.16}$$

Through inserting (2.16) into eq.(2.12) to eq.(2.14), the rest of the unknowns are obtained as:

$$f_2 = f_6 + \frac{1}{2}(f_7 - f_3) + \frac{1}{2}\rho(\boldsymbol{r},t)u_x(\boldsymbol{r},t) + \frac{1}{6}\rho(\boldsymbol{r},t)u_y(\boldsymbol{r},t) \tag{2.17}$$

$$f_8 = f_4 + \frac{1}{2}(f_3 - f_7) - \frac{1}{2}\rho(\boldsymbol{r},t)u_x(\boldsymbol{r},t) + \frac{1}{6}\rho(\boldsymbol{r},t)u_y(\boldsymbol{r},t) \tag{2.18}$$

$$\rho(\boldsymbol{r},t) = \frac{1}{1 - u_y(\boldsymbol{r},t)}\left[f_0 + f_3 + f_7 + 2(f_4 + f_5 + f_6)\right] \tag{2.19}$$

This is the full set of equations at the south boundary to obtain all the unknowns. By applying these conditions at the boundaries, the streaming process can now be done throughout the whole domain.

2.3.3 Remarks on corner treatment

Treating corners have always been of great concern to code developers. There is simply no literature available on corner treatment for the LBM. If corners are not treated correctly, results may be still obtained while the main physics will be captured but the small scale tends to differ and especially in cases like the lid driven cavity, where the corners are all connected via the two diagonals, the error propagates along these links and influences the small scale phenomena. In the case of the lid driven cavity, the secondary vortices are insufficiently captured and vanish completely for low Reynolds numbers although experiments confirm that vortices should exist. The demonstrate the problematic nature, the corner of the lid driven cavity is examined in detail and given in Figure 11 in the problematic junction of north and east wall where two different kind of boundary condition are present, namely stationary and moving wall.
While the east boundary updates the distribution function f_6, f_7 and f_8, the north boundary updates the distribution functions f_4, f_5 and f_6. Thus, f_6 can be treated by either the north or east boundary. The problem arises when two different boundary conditions are present, in this case the east wall is stationary while the north boundary is moving. Hence, f_6 could take two different values, the question is which one is the right one. As stated before, the literature is not very clear on that problem and most of the research available does not deal with this problem

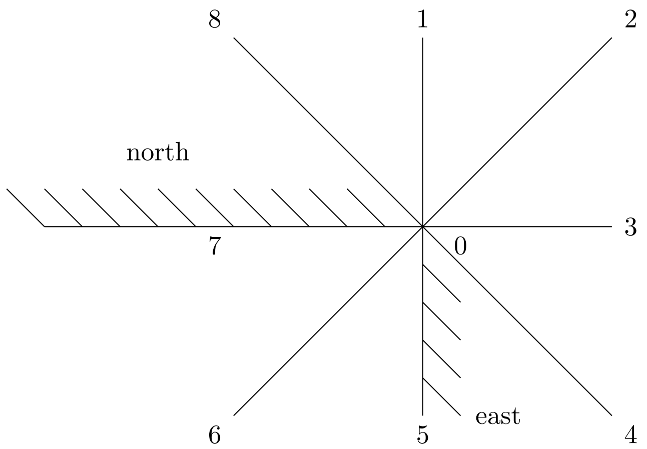


Figure 11 – Distribution function present at corner of the lid driven cavity.

separately. Due to the unique nature of the LBM (in contrast to NSE based solvers where flux direction and normal direction are solely of interest), the diagonal component has to be solved in the same manner in all flows. This leaves the only option to treat the corners with zero velocities at all times. This means that two stationary walls can be easily connected to each other. Moving walls must return to a stationary wall in the corner. For inlets and outlets this means that they too can only span from the points next to the corner which limits this approach in such regions. It is not possible to put two inlets or outlets or a combination of in and outlet together. A small amount of wall has to be implemented right in between this area. This is true for the Zou / He boundary treatment and may differ for other methods, however, it is greatly suggested to follow the above made recommendations.

2.4 Obtaining macroscopic quantities

As the transport equation eq.(1.32) of the LBM is solved, the distribution function in all lattice direction at each site is obtained. This is a rather useless physical property as it tells very little about the flow field itself. This section explains the steps necessary to obtain all macroscopic quantities known from macroscopic approaches, such as the NSE.

2.4.1 Density

The density is the easiest of all properties to obtain. Once the distribution function is known, the sum at each node yields the density as:

$$\rho(\boldsymbol{r}, t) = \sum f(\boldsymbol{r}, t) \tag{2.20}$$

2.4.2 Pressure

The pressure is linked to the density through the lattice equation of state which is:

$$p = c_s^2 \rho \tag{2.21}$$

where c_s is the lattice sound of speed and equals $c_s = 1/\sqrt{3}$. This means that the pressure is only a scaled version of the density and equals:

$$p = \frac{\rho}{3} \tag{2.22}$$

2.4.3 Velocity

Once the density is known, the fluxes can be evaluated as:

$$\rho(\boldsymbol{r}, t)\boldsymbol{u}(\boldsymbol{r}, t) = \sum c(\boldsymbol{r}, t) f(\boldsymbol{r}, \boldsymbol{c}, t) \tag{2.23}$$

With the knowledge of the fluxes, the velocity can be easily extracted as:

$$\boldsymbol{u}(\boldsymbol{r},t) = \frac{\rho(\boldsymbol{r},t)\boldsymbol{u}(\boldsymbol{r},t)}{\rho(\boldsymbol{r},t)} \tag{2.24}$$

2.4.4 Temperature

A bit more modelling is needed to obtain the temperature. Since no information on the energy is available so far with the models discussed so far, the temperature cannot be obtained. For modelling compressible flows however, this information is needed and hence the discussion is left to section 3.4, where compressible flows are examined.

2.4.5 Length

The length does not change in LBM. It is only discussed here for the purpose of a complete overview.

2.4.6 Time

The LBM is an unsteady process and each iteration one step forward in time. The LBM has two velocity scales, the macroscopic one obtained from eq.(2.24) and the mesoscopic one, determined by the speed model. The mesoscopic velocity scale is the one used during the streaming, i.e. the distribution function propagates its information at each time step / iteration through the domain (from node to node) with a unit velocity. Since the length is the same as discussed previously and the velocity streams with the unit velocity, the time step is equal to the lattice spacing, which can be also derived from:

$$t = \frac{l_s(\boldsymbol{r})}{c} = \frac{l_s(\boldsymbol{r})}{1} = l_s(\boldsymbol{r}) \tag{2.25}$$

Here, l_s is the lattice spacing. This shows one particular drawback of the LBM. Since time step and length are always the same, the CFL (Courant Friedrichs Lewy) number is equal to unity for all simulations which causes slow convergence rates. Furthermore, the implementation of unstructured grids (also called "off-LBM") with non equidistant spacing into the framework of the LBM causes cells with different spacing to march in time with different time steps.

2.4.7 Reynolds number

The Reynolds number is not a physical quantity as the aforementioned ones but differs from the macroscopic description and should be therefore examined here as well.
In macroscopic space, the Reynolds number is obtained as:

$$Re = \frac{\rho l_c u_c}{\mu} = \frac{l_c u_c}{\nu} \tag{2.26}$$

where l_c is the characteristic length, u_c the characteristic speed and μ, ν the kinematic and dynamic viscosity, respectively. For the mesoscopic Reynolds number, the concept of characteristic length does not exist anymore. Since it is a mesoscopic approach, the length scale is replaced by the discretized number of points along the characteristic length direction. In other words, it is the characteristic length divided by the lattice spacing, i.e. l_c/l_s. This can be thought of replacing the characteristic length by the mean free path (the mean free path in the LBM framework). The viscosity needs to be adapted as well and takes the form of a so called lattice viscosity, $\nu_{lattice}$, while the velocity is the same as in eq.(2.27). The lattice viscosity has no relation to its physical counterpart and can be more though of a dissipation. Hence, to obtain the required Reynolds number the usual approach is to take a stable value for the lattice viscosity, usually bound by the collision approximation and then set the velocity to any arbitrary value to math the macroscopic Reynolds number as the characteristic length (number of points along characteristic length direction) is fixed due to the grid. The mesoscopic Reynolds number is thus expressed as:

$$Re = \frac{N_{lattice} u_c}{\nu_{lattice}} \tag{2.27}$$

3 Literature review

This section will elaborate on current challenges and limitations of the LBM. The most relevant literature has been grouped into their area of research activities and is shown in Table 5 in decreasing order. Their percentage value was calculated based on the total number of papers available on the LBM[5]. It has to be noted, that most of these areas overlap, flow through porous media, for example, is often simulated in conjunction with multiphase flows and compressible flows are closely linked with heat transfer. Thus, Table 5 is giving the absolute occurrence of these keywords. The sum over all percentage values is greater than 100% , which is due to the overlapping.
Based in these findings, the literature was further divided into two groups. The first group contains the relevant literature that has a major impact on the development of current lattice Boltzmann models. These are: multiphase flows, porous media, turbulent flows, compressible flows and heat transfer. Multiphase flows are easy to implement in the LBM which is the reason for its popularity within the LBM. Porous media can be modelled using single phases but often has more than one phase and is thus discussed after multiphase flows. Turbulent flows are omnipresent and need to be treated to some extend differently from turbulence in the NSE. Compressible flows are increasingly investigated for high speed applications and are shortly discussed followed by heat transfer.
The second group treats topics which are developing fast and increasing in popularity. These topics are treated shortly in the further reading section 3.6 and are aimed to give an overview of what else can be done using the LBM. These include: quantum mechanics, reactive flows, bioengineering and magnetohydrodynamics.

3.1 Multiphase flows

This section starts with one of the most important application areas of the LBM. Multiphase flows are one of the current challenges in fluid mechanics research and industry. In most cases, experiments are prohibitively expensive and measurements too inaccurate [41, p.1]. CFD has become a very powerful tool in this field, although uncertainties are still wide spread due to poor knowledge of the underlying flow phenomena. Traditionally, the modified NSE are employed, which track each phase either by an Eulerian-Eulerian approach or, if the particle motion is of particular interest, an

[5]based on www.scopus.com, 09.05.2013

Table 5 – Overview of current research activities for the LBM

Topic	Papers	Percentage
Heat Transfer	1617	24.1%
Porous media	1568	23.4%
Multiphase flows	1548	23.1%
Turbulent flows	758	11.3%
Reactive flows	672	10.0%
Bioengineering	671	10.0%
Compressible flows	580	8.6%
Magnetohydrodynamics	415	6.2%
Quantum mechanics	271	4.1%

Eulerian-Lagrangian approach. For the Eulerian-Eulerian approach, a mass fraction is introduced to the modified NSE. This fraction represents each portion of each phase within one computational cell. The sum of all fractions must be unity. For each phase and dimension, one extra momentum equation must be solved. This means one extra non-linear partial differential equation per phase and dimension and hence a huge computational cost for multiple species. The LBM on the other hand, consists just of one equation. Depending on the multiphase model used, the extra phase can be as simple as a secondary, additive distribution function superimposed with the primary phase. There are essentially three approaches to model multiphase flows.

Gunstensen *et al.* [42] derived a model in 1991, which follows the above mentioned approach. The distribution function f_i is the sum of the species distribution function. Guntensen referred to them as the distribution of red and blue particles, i.e.:

$$f_i = f_i^r + f_i^b \tag{3.1}$$

In terms of the Lattice-Boltzmann evolution equation as introduced in section 1.6.2, eq.1.34 becomes:

$$\Omega_i^k(\boldsymbol{r}, t) = f_i^k(\boldsymbol{r} + c_i, t+1) - f_i^k(\boldsymbol{r}, t) \tag{3.2}$$

where the collision operator Ω_i^k is expanded as:

$$\Omega_i^k = \left(\Omega_i^k\right)^{(1)} + \left(\Omega_i^k\right)^{(2)} \tag{3.3}$$

Here, the first term on the right-hand side $\left(\Omega_i^k\right)^{(1)}$ is equivalent to the LBGK collision operator as introduced in section 2.1.1. The second term $\left(\Omega_i^k\right)^{(2)}$ represents the added surface tension interactions. It is defined as:

$$\begin{aligned} \left(\Omega_i^k\right)^{(2)} &= \frac{\xi}{2}|\mathbf{\Pi}| \left[\frac{(c_i\mathbf{\Pi})^2}{|\mathbf{\Pi}^2|} - \frac{1}{2}\right] && (3.4) \\ \mathbf{\Pi}(\boldsymbol{r}) &= \sum c_i \left[\rho^r(\boldsymbol{r} + c_i) - \rho^b(\boldsymbol{r} + c_i)\right] && (3.5) \end{aligned}$$

ξ is a free parameter and is derived from the surface tension. $\mathbf{\Pi}$ is known as the colour gradient because the original methods compares phases of different colours.

Shan and Chen [43] as well as Shan and Doolen [44] modified the expression for $\left(\Omega_i^k\right)^{(2)}$, taking microscopic interaction into account. The revised collision operator becomes:

$$\begin{aligned}\left(\Omega_i^k\right)^{(2)} &= c_i \cdot \mathbf{\Pi}^k & (3.6)\\ \mathbf{\Pi}^k(\boldsymbol{r}) &= -\sum_{k'}\sum_i V_{kk'}(\boldsymbol{r}, \boldsymbol{r}+c_i)c_i & (3.7)\end{aligned}$$

where $V_{kk'}$ is an interaction pseudo potential between particles. This term is used for modelling inter molecular interactions and can be derived from different potential approaches, i.e. fractional or van der Waals potentials. Information on how to determine $V_{kk'}$ can be found in Qian *et al.* [45].

A third model to model multiphase flows is the so-called free energy approach and was introduced by Swift *et al.* [46, 47]. In this method, the coefficients of Γ_k in the equilibrium distribution function (eq.(1.24)) are recalibrated and a source term $\boldsymbol{G}_{i,j}$ is introduced that assures conservation of energy. The modified equilibrium distribution function thus becomes:

$$\widehat{f_i^{eq}} = f_i^{eq} + \boldsymbol{G}_{i,j}c_ic_j \tag{3.8}$$

In this context, i and j are the column and row indices. Eq.(3.8) ensures the conservation of total energy, i.e. surface, kinetic and internal energy. A comprehensive overview of the aforementioned models can be found in Chen and Doolen [48]. These are only three popular models, although a variety of models exist. A thorough discussion of all models is beyond the scope of this work. Applications and variations of these models are further explored in [49–60].

Current challenges in multiphase flows using the LBM are the treatment of the phases interface. Huang *et al.* [61] compared five different, popular model (including the Shan and Chen (SC) scheme) showing that the surface tension treatment is the distinguishing factor in all models. Rosales and Whyte [62] implemented a dual grid approach, similar to the multi grid method used in the NSE (see Martinelli *et al.* [63] for details), where the momentum is resolved on the standard grid (lattice) while the order parameter[6] is computed on a finer grid. Simulations showed that by adapting the dual grid approach, computational time could be significantly reduced over a full grid refinement, where the momentum is solved on the same (fine) grid as the order parameter, while the result for the dual grid approach is of the same order as the simulation of the full grid refinement [62, p.1076]. They concluded that the momentum based calculation can be done on a rather coarse grid and is less affected by the grid resolution than the order parameter. Li *et al.* noted that the momentum is not consistently recovered with the current approaches. By including additional forces to the system, simulations proved that for an increase in velocity (Reynolds number), the correct momentum was recovered.
As pointed out by Leveque [64], non-equilibrium gas dynamic is needed to model certain phenomena in hypersonic flows. The system of inviscid equations, due to the high Reynolds number, is represented by the conservation laws with source terms, which model the chemistry. Leveque studied a model problem which was taken by Reis and Dellar [65], who investigated the same problem

[6]The order parameter is used when dealing with changes in states, liquid to gas, for example, and gives information on the phase. For multiphase flows, the density is conveniently chosen to be the order parameter in each phase.

using the LBM. The onset of Lattice pinning could be delayed by modifying the random projection method of Bao and Jin [66] and applying it to the LBM for multiphase flows. Here, lattice pinning is the phenomenon occurring when modeling the width of the phases interface. Usually this is done by a smooth scalar function but rather than changing in time due to advection, it gets pinned to the lattice because of its resolution. The "uniformly distributed quasi-random variable" as Reis and Dellar called this random approach not only delays the onset of the lattice pinning but also conserves the mass.
He *et al.* [67] introduced a novel model, known as the He-Chen-Zhang (HCZ) model where the interface modelling is based on molecular interactions. The result is a sharp representation of the interfaces. Zhan *et al.* [68] investigated test cases using the HCZ model for various benchmark cases, stating a very high level of accuracy. Benchmark cases included the Rayleigh-Taylor instability as well as a capillary wave dispersion of which the latter produced errors less than one percent compared with theoretical models. Despite the good interface resolution, spurious oscillation and dispersions are the shortcomings of the model near the phases interface. Chao *et al.* [69] introduced a filter function to the HCZ model, based on Kim's [70] devised model for modelling surface tension, to remove the dispersive nature while conserving the favourable properties. In addition, a mass correction procedure has been implemented to counteract the long-time modelling and numerical error accumulation. The modified version can predict high density ratio flows while conserving the mass.

Multiphase flows are still under active research both in the domain of Lattice Boltzmann but also for the NSE. The aim of this section was therefore to give an overview of approaching multiphase flows using the LBM and to state the current limitations.

3.2 Porous media

The flow through porous media is of considerable complexity and ponderous modelling. Wall-fluid interactions are dominant and usually take long computational time using classical finite difference schemes. Schwarz *et al.* [71] used such a scheme, but they are usually limited to simple physics or small domains. Porous media, in its most general description, is a material including pores. For studying such materials, the permeability K as a measurement for a fluid to pass through a porous medium and the solid fraction χ_s as the fraction that occupies the domain with solid material (the matrix) is introduced. There are many natural materials that are porous such as rocks, soil, zeolites, bones, wood and cork but man-made materials such as cement and ceramic posses the same property. The interaction of fluid passing through such material is therefore of great interest.
Henry Darcy [72] was the first to discover the relationship between the linear pressure gradient and volume flow rate per unit area by investigating natural fountains in his birthplace Dijon in France as:

$$\dot{V} = -\frac{K}{\rho\nu}\frac{\partial p}{\partial x} \tag{3.9}$$

This equation was later successfully verified by Rothman [73] using a LGCA approach. Succi *et al.* [74] measured the permeability in a random three-dimensional media and were able to confirm the validity of eq.(3.9) for the LBM as well. Such results are to be expected as Darcy's law is derived from the Stokes equation, which in turn are a simplified version of the NSE. Cancelliere *et al.* [75] did a parametric study of the permeability K as a function of the solid fraction χ_s in the range of $0.02 < \chi_s < 0.98$. They reported good agreement with Brinkman's approximation [76] stated as:

$$K = K_0 \left[1 + \frac{3}{4}\chi_s \left(1 - \sqrt{\frac{8}{\chi_s} - 3} \right) \right] \qquad \text{for } \chi_s < 0.2 \tag{3.10}$$

where K_0 is the effective permeability of a single sphere. For values of $\chi_s > 0.2$, the Kozeny-Carman equation, introduced by Kozeny [77] in 1927 and later refined by Carman [78, 79], has been used and showed equally good agreement. The semi-empirical Kozeny-Carman equation is given as:

$$K = \frac{(1-\chi_s)^3}{6A_s^2} \tag{3.11}$$

where A_s is the specific surface area. The results of Cancelliere *et al.* were confirmed by Heijs and Lowe [80] who further investigated soil samples where flow occurs in specific continuous connected pores. The Kozeny-Carman equation was not able to match the results, demonstrating its limitation.

Flows through porous media need a randomized geometry to start with. Usually this is created by superimposing spheres where voids are automatically created due to the shape of spheres that represents the pores. Adler and Thovert [81] developed a tool capable of creating randomized geometries either being macroscopically homogeneous or non-homogeneous. Pan *et al.* [82] thoroughly investigated the flow through porous media using a single (LBGK) and multi-relaxation time scheme for the collision process and four different boundary conditions for the wall, namely a standard bounce back, linearly interpolated bounce-back, quadratically interpolated bounce-back and a multi reflective scheme. The investigation was looking at the viscosity dependence of the permeability and the discretization error for all four boundary conditions and two collision operators. They concluded that the multi-relaxation time scheme generally provided better results over the LBGK collision operator and that the interpolation at the wall boundary significantly improved the accuracy of the results. Ferréol and Rothman [83] simulated the fluid flow through a Fontainebleau sandstone. Both a LBM and a LGCA were used giving good agreement for both approaches. Guo and Zhao [84] added a force term to the equilibrium distribution function to account for linear and non linear drag forces. Their approach was validated against a poiseuille, couette and a lid-driven cavity flow. Koponen *et al.* [85] used large, three-dimensional random fibre webs, representing papers or non woven fabrics, and confirmed the exponentially dependence of permeability on porosity over a large range of porosity. Further investigations on porous material using a two-phase flow were done by Pan *et al.* [86] with good numerical results compared against experiments. They were specifically looking at the entry pressure, displacement slopes, irreducible saturation and residual entrapment. All simulations where done using a random geometry based on the superimposing sphere approach on two geometries. The first contained 1200 spheres while the second only 150. A significant difference in the results was reported concluding that a lower bond exists for a representative volume of elementary size.

This outlook showed mainly the analysis of single phase flows through porous media, sometimes also referred to as complex geometries [48]. Multiphase flows are increasingly treated in the domain of porous media using the LBM because the formulation of wall-fluid boundary conditions in conjunction with multiple species are overly complicated to implement using classical NSE. Not only developing time but also computational time is less with the LBM.

3.3 Turbulence modelling

Turbulence is present in the vast majority of engineering applications. One of the requirements for an algorithm is the capability of simulating turbulence at a reasonable cost and high accuracy. In the NSE, still one of the most used engineering approaches to tackle turbulent flows are the Reynolds Averaged Navier-Stokes (RANS) equations, which are based on the Reynolds decomposition that was introduced by Osborne Reynolds in 1895 [87]. Reynolds postulated that velocity can be decomposed into a mean and fluctuation velocity component, formally expressed as:

$$\widehat{\boldsymbol{u}}(t) = \overline{\boldsymbol{u}} + \boldsymbol{u}'(t) \tag{3.12}$$

where $\widehat{\boldsymbol{u}}(t)$ is the instantaneous velocity vector, $\overline{\boldsymbol{u}}$ denotes the mean velocity vector and $\boldsymbol{u}'$ the fluctuating velocity vector. Per definition, the fluctuating part has to be zero if integrated over time, i.e. $\int \boldsymbol{u}' dt = 0$. Inserting eq.(3.12) into the NSE, i.e. into the continuity, momentum and energy equation yields the RANS equations. The derivation of these equations can be found in Lynch and Smithy [88] while more information on the Reynolds decomposition and statistical approaches in turbulent flows can be found in Pope [89].
Although it is theoretically possible to devise turbulence models similar to the RANS models in the NSE, a hybrid approach is more commonly used in the framework of LBM. This approach uses a Navier-Stokes based solver in the far-field while the near wall region is modelled using a large eddy simulation (LES) technique, which is incorporated into the LBM, as for example in [90–93]. LES, on the other hand, is much more versatile than the RANS approach and thus easier to apply to the LBE. The following description is applicable to the above mentioned hybrid RANS/LES-LBE approach or for exclusive treatment of LES. Much like the LES for the NSE, the energy spectra is divided into two parts: resolved and unresolved scales. A cut-off wave number is defined up to where the spectrum is resolved. A filter then defines the transition from the resolved to the unresolved scales. The effect of different filters on the results is discussed in [94–99].
The new LBE takes the following form:

$$\overline{f_i}(\boldsymbol{r} + \boldsymbol{c}\mathrm{d}t, \boldsymbol{c} + \boldsymbol{F}\mathrm{d}t, t + \mathrm{d}t) - \overline{f_i}(\boldsymbol{r}, \boldsymbol{c}, t) = \frac{1}{\widehat{\tau}} \left(\overline{f}^{eq} - \overline{f} \right) \tag{3.13}$$

The over line indicates that the equation has been filtered. $\widehat{\tau}$ is the modified collision time, taking the sub grid scale (SGS) collisions on a time scale τ_{SGS} into account. The viscosity needed to transfer kinetic energy from larger to smaller scales (or from lower to higher wave numbers) comes from the sub grid scale model. The viscosity is calculated as:

$$\nu_e = \nu + \nu_T = \frac{1}{3}\left(\widehat{\tau} - \frac{1}{2}\right)\boldsymbol{c}\mathrm{d}\boldsymbol{r} \tag{3.14}$$

ν_T is the turbulent viscosity and also referred to as the eddy viscosity. ν_e is the effective (kinematic) viscosity. The SGS model is used to determine the turbulent viscosity such that $\nu_T = \nu_{SGS}$. Smagorinsky [100] was the first to develop a suitable SGS model which was later revised by Lilly [101] and is known as the Smagorinsky or Smagorinsky-Lilly model. For the LBE, it is formulated as:

$$\nu_T = (C_s\Delta)^2\sqrt{2\sum \overline{S}_{i,j}\overline{S}_{i,j}} \tag{3.15}$$

where C_s is the Smagorinsky coefficient which needs to be calibrated for each flow type individually. Δ is the cut-off filter width, everything smaller than Δ will be modelled by the SGS model. The strain rate is denoted by $\overline{S}_{i,j}$ where the over line once again stands for a filtered quantity. It is defined as:

$$S_{i,j} = \frac{1}{2}\left(\frac{\partial \overline{u}_j}{\partial r_i} + \frac{\partial \overline{u}_i}{\partial r_j}\right) \tag{3.16}$$

with this knowledge, τ_{SGS} can be determined as:

$$\tau_{SGS} = \frac{1}{2}\left(\sqrt{\tau^2 + 18\sqrt{2}(\rho c^2)^{-1}C_s^2\Delta\overline{S}_{i,j}} - \tau\right) \tag{3.17}$$

This only outlines the basic steps needed in order to simulate turbulent flows. More information on LES modelling within the LBM can be found in [102–107].

Another possibility to model turbulences is to employ direct numerical simulations (DNS). The underlying assumption, as in DNS for the NSE, is that the grid resolution is fine enough to capture the small-scale phenomena that occur at the Kolmogorov scales. When Kolmogorov introduced these scales in 1941 [108–110], turbulence was not an unknown occurrence but practical engineering and academic research were still trying to understand pressure drops in channel flows and drag due to body friction [111, p.590] which are well studied examples nowadays. Kolmogorov states that at these scales, viscous forces become dominant and all motions are dissipated into heat due to the viscosity. This happens at an equivalent Reynolds number of unity based on the eddy velocity and size at these scales. Therefore, DNS resolves the grid up to these scales and all filtered quantities which were introduced for the LES become unfiltered, resolved quantities without the over line.

The viscosity is calculated directly without the eddy viscosity assumption as:

$$\nu = \frac{1}{3}\left(\tau - \frac{1}{2}\right)\boldsymbol{c}\mathrm{d}\boldsymbol{r} \tag{3.18}$$

The strain rate can be further extracted from the flow field using:

$$S_{i,j} = -\frac{1}{2\rho c_s^2 \tau}\sum c_i c_j \left(f_i - f_i^{eq}\right) \tag{3.19}$$

where c_s is the speed of sound in lattice units and depending on the speed model used. The equilibrium distribution function takes a new form as:

$$f_i^{eq} = \omega_i\left[\rho' + \rho\left(\frac{3c_i\cdot\boldsymbol{u}}{c^2} + \frac{9c_i\cdot\boldsymbol{u}}{2c^4} + \frac{3u^2}{2c^2}\right)\right] \tag{3.20}$$

where ρ' is the fluctuating density and can be obtained in a similar manner as eq.(3.12):

$$\rho(t)' = \widehat{\rho}(t) - \overline{\rho} \tag{3.21}$$

the definitions are accordingly, i.e. $\widehat{\rho}(t)$ is the instantaneous density component and $\overline{\rho}$ is the mean density. The dissipation rate can be found as:

$$\varepsilon_D = 2\nu\sum S_{i,j}S_{i,j} \tag{3.22}$$

As in the NSE, since DNS resolves all the scales, no further modelling is required. Yu *et al.* [112,113] explored the capabilities of the LBM for homogeneous decaying isotropic turbulence and used a LES approach. Results were compared against similar studies using LES and the NSE and both results from the LBM and the NSE were compared against DNS results. It was observed that the LES results obtained from the LBM were closer to the DNS than the LES results from the NSE. The explanation lies in the strain rate which is directly calculated for the LBM from scalar quantities, see eq.(3.19), while it can only be approximate for the NSE using a finite difference (or finite volume) approach due to the derivatives, see eq.(3.16). Since the Smagorinsky constant is depending on the strain rate (eq.(3.19)), the determined value for C_s is therefore more accurate than for the NSE. A generally lower value of C_s was further observed for the LBM than for the NSE.
Although hybrid turbulence models using RANS in the far-field and LES in the near wall region are popular within the LBM framework, Tekeira [114] implemented a two-layer mixing-length algebraic model as well as the standard and RNG k-ε model including wall functions into the LBM. Tekeira used two benchmark cases: a straight pipe flow and the backward facing step. While good agreement for all three models for a straight pipe flow compared to experiments were achieved, the

backward facing step yielded best results with the RNG k-ε giving a 2% deviation for the reattachment point. An overall good recovery of the mean flow structure was reported for all models. Böhle and Becker [115] investigated the development of laminar boundary layers, comparing an in-house code based on the LBM with a commercial CFD package based on the NSE. The results were validated against an analytical formula. The overall agreement was good and the LBM performed equally well compared against the commercial CFD code. Reported flow properties such as the skin friction coefficient were accurately reproduced with both codes.
Fakhari and Lee [116] successfully simulated the Kelvin-Helmholtz instability using a multi-relaxation time scheme, as introduced in section 2.1.3. The multi-relaxation time scheme was able to effectively damp the high-frequency motion within the kinetic energy originating from numerical moving waves. By damping this motion, the stability could be increased. Chiappini *et al.* [117] used a model devised by Lee *et al.* [118,119] who used a high order differencing approach for the non-ideal forces. This resulted in a very sharp resolution at the fluid-fluid interface which is free of spurious oscillation. The problematic treatment of the interface has been addressed in section 3.1. Despite the good resolution, the model lacked mathematical conservation of mass, which was analytical proven by Chiappini. Even though the mass was not conserved, this adverse characteristic acted much like an artificial viscosity that produced a self-stabilization dynamic for long-term simulations. Chiappini investigated the Reyleigh-Taylor instability reporting that the Lee *et al.* model agrees well with experiments in terms of spike and bubble leading front position.

This section showed the capabilities of the LBM to handle turbulent flows. Derivations for DNS and LES have been given while an outlook on applications and current research has been given to demonstrate the impact of turbulence modeling within the LBM.

3.4 Compressible flows

Compressible flows have attracted great interest in past research and industrial applications. Discontinuities impose great mathematical challenges to be captured accurately. This is usually done using high-order schemes in conjunction with a compressible Navier-Stokes or Euler code. Recently, some attempts have been made to introduce compressible models into the LBM. Still, a lot of research is needed since compressible flows are all thermal in nature but thermal calculations using the LBM have proven to be a burdensome area. More research is needed to understand compressible flows as many of the current models used for compressible flows within the LBM are still not fully tested and understood [120, p.14]. One of the major restrictions is that the equilibrium distribution function is only valid for monatomic particles and as soon as diatomic particles are considered, the added degree of freedom will increase the internal energy and thus the temperature. An alternative formulation has been suggested by Kun [120].

There are a couple of models trying to mimic the compressible behavior, usually only suitable for a certain range of applications and thus having a restrictive rather than a global validity. A universal model does not exist to the present which is capable of simulating all kinds of compressible flows.
Yan *et al.* [121] developed a model which consists of two particles at each site, i.e. $f_{i,A}$ and $f_{i,B}$,

except for the center site. This means in the context of the $D2Q9$ speed model, that two particles exist at $i = 1 \ldots 8$ while only one particle is present at the center $i = 0$, called the static particle. Three energy levels are introduced as ζ_A, ζ_B and ζ_C where ζ_A is associated with $f_{i,A}$, ζ_B with $f_{i,B}$ and ζ_C with the static particle. The conserved, macroscopic properties are then recovered as:

$$\rho = \sum f_{i,k} \tag{3.23}$$

$$\rho \boldsymbol{u} = \sum f_{i,k} c_i \tag{3.24}$$

$$\frac{1}{2}\rho u^2 + \rho E = \sum f_{i,k} \zeta_k \tag{3.25}$$

$$\rho \boldsymbol{u}\boldsymbol{u} + p \cdot \boldsymbol{I} = \sum f_{i,k}^{eq} c_i c_i \tag{3.26}$$

$$\left(\frac{1}{2}\rho u^2 + \rho E + p\right) \boldsymbol{u} = \sum f_{i,k}^{eq} \zeta_k c_i \tag{3.27}$$

$$\tag{3.28}$$

with E being the total energy and $\boldsymbol{I}$ the identity matrix. The equilibrium distribution function is the same as in eq.(1.24). The problematic nature is that the coefficients for Γ_k are free parameters and the results are depending on its choice. Shi *et al.* [122] used a similar approach where the energy levels were associated as: ζ_A for $f_{i,A}$ in the range of $i = 1 \ldots 4$, ζ_B for $f_{i,B}$ in the range of $i = 5 \ldots 8$, and ζ_C for the static particle. Furthermore, the energy levels were now representing the extra amount of energy due to added degrees of freedom. This choice of the energy level (referred to as the rest energy) made the specific heat ratio γ adjustable and therefore, the free coefficients in the equilibrium distribution function computable. The macroscopic quantities for the Shi *et al.* model are computed as:

$$\rho = \sum f_i^{eq} \tag{3.29}$$

$$\rho \boldsymbol{u} = \sum f_i^{eq} c_i \tag{3.30}$$

$$\frac{1}{2}\rho u^2 + \rho E = \sum f_i^{eq} \left(\frac{1}{2}c_i^2 + \zeta_k\right) \tag{3.31}$$

$$\rho \boldsymbol{u}\boldsymbol{u} + p \cdot \boldsymbol{I} = \sum f_i^{eq} c_i c_i \tag{3.32}$$

$$\left(\frac{1}{2}\rho u^2 + \rho E + p\right) \boldsymbol{u} = \sum f_i^{eq} \left(\frac{1}{2}c_i^2 + \zeta_k\right) c_i \tag{3.33}$$

$$\tag{3.34}$$

Both preceding models are able to simulate compressible effects in subsonic flows but struggle to predict discontinuous behaviour. This shortcoming was removed by the model of Kataoka and Tsutahara [123, 124] which, similar to the Shi *et al.* model, allows the specific heat ratio to be adjustable. Further constraints allowed consistent macroscopic properties with the Euler equation in the limit of small Knudsen numbers. They are given as:

$$\rho = \sum f_i^{eq} \tag{3.35}$$

$$\rho \boldsymbol{u} = \sum f_i^{eq} c_i \tag{3.36}$$

$$\rho \boldsymbol{u}\boldsymbol{u} + p \cdot \boldsymbol{I} = \sum f_i^{eq} c_i c_i \tag{3.37}$$

$$\rho \left(\frac{2}{\gamma - 1} RT + u^2 \right) = \sum f_i^{eq} \left(c_i^2 + \eta_i^2 \right) \tag{3.38}$$

$$\rho \left[\left(\frac{2}{\gamma - 1} + 2 \right) RT + u^2 \right] = \sum f_i^{eq} \left(c_i^2 + \eta_i^2 \right) c_i \tag{3.39}$$

where η_i is a parameter to adjust the specific heat ratio. This model was the first model derived for simulating shock waves and contact discontinuities simultaneously. Results for the Sod shock tube problem were captured. It was reported by Kataoka and Tsutahara that for step changes in one location, the model satisfies the weak solution of the Euler equation.
Sun [125–129] introduced a novel approach which is known as the adaptive LBM. The attribute adaptive means that the velocity becomes scalable. For the standard LBM, particles travel from site to site but are never on a link at a discrete time. Sun introduced a second velocity which is not discrete but continuous and hence all properties calculated using the velocity as the kinetic energy, for example, are more accurately represented. Sun was able to simulate the Sod shock tube and concluding that this model can be used for high Mach number flows.

The Sod shock tube problem has already been mentioned and used by the various authors to validate their model. He *et al.* [130] further discussed compressible flows for the LBM and showed case studies based on a compressible model introduced by Shan *et al.* [131] including the shock tube problem, the lid driven cavity at a lid Mach number of $Ma = 0.7$, a transonic flow around a NACA 0012 profile and the flow field around a high-speed train. Zhong *et al.* [132] investigated in general the flow around airfoils for compressible conditions, giving good agreement with experiments. Zhou *et al.* [133] and Shock *et al.* [134] further noticed that the separation point is predicted in close agreement for high Reynolds numbers with experimental data. In addition, Zhuo used different turbulence models declaring best agreement with a modified Spalart-Allmaras turbulence models for the LBM.
Compressible flows are also encountered for river flows. Usually the shallow water equations have to be solved, including the Riemann problem. Thömmes *et al.* [135] applied the LBM for this kind of application and validated a shallow water based LBM against an analytical solution, without having to solve the Riemann problem. Due to good agreement, this method was then applied to the strait of Gibraltar giving the correct physical flow field and again good agreement where data was available for comparison.
Although the adversity of simulating thermal flows has been noticed, Watari and Tsutahara [136] used a three-dimensional multi-speed thermal model to investigated supersonic flows. This has been validated and verified against a couette flow, a normal shock wave problem and a supersonic nozzle flow. Attempts to simulate hypersonic flows can be found in Kamali and Frad [137] for example, where the hypersonic flow around a two-dimensional cylinder was investigated. However,

noticeable less literature is available on high Mach number flows. Only 0.9% of the current literature deals with supersonic flows and 0.3% with hypersonic flows.

This concludes the section on compressible flows for the LBM. Popular models and their approach to conserve the primitive variables have been discussed and some applications with current limitations and shortcomings elucidated. Compressibility within the LBM needs more research in order to overcome some of the limitations present in the current models.

3.5 Heat Transfer

The preceding section discussed compressible flows, which are, as described, thermal in nature. Therefore, all compressible models derived in the past section inherently include the ability to simulate thermal flows. The aim of this section is to give a general overview of what has been done in the area of heat transfer using the LBM.
The temperature is not directly calculated by the LBM, see eq.(1.32). For simplicity eq.(3.25) from the Yan *et al.* model is considered, where the total Energy E is calculated from the scaled distribution function, the scaling parameter being the energy level. The macroscopic conserved equations, i.e. the left-hand side of the equations (for all models, i.e. Yan *et al.*, Shi *et al.* and Kataoka and Tsutahara), can be obtained by the Chapman-Enskog expansion and analysis as introduced in section 1.6.2 by comparing terms of the same smallness parameter. Details can be found in Alexander *et al.* [138] who derived the equality for total energy through the Chapman-Enskog expansion as:

$$\rho E = \sum f_i \frac{(c_i - \boldsymbol{u})^2}{2} \tag{3.40}$$

$$\tag{3.41}$$

The equilibrium distribution function was further expanded up to a truncation error of $\mathcal{O}(u^4)$. Simulations of a two-dimensional couette flow between two heated plates with a temperature gradient present agreed well with the theory for small changes in temperature. Vahala *et al.* [139] demonstrated the usability for this model in a two dimensional, turbulent shear layer flow with a steep temperature gradient present. Qian [140] further extended the model for three-dimensional flow while Chen *et al.* [141] implemented a multi speed model and removed some non-physical effects. One fundamental problem in the LBM is the discrete velocity which has been elaborated on in section 3.4. The resulting effect is a constant Prandtl number since momentum and thermal modes relax at the same speed. Chen *et al.* [142] and McNamara *et al.* [143] modified the model which allowed the variation of the Prandtl number.
Bartoloni *et al.* [144] and Massaioli *et al.* [145] simulated the Rayleigh-Bérnard convection using an active scalar approach which proved to be a non favourable method because the dissipation function is difficult to be implemented correctly. Shan [146] investigated the same natural convection based on a LBM model devised by himself and Chen [147] using a passive scalar approach. This time results were generally better and the critical Rayleigh number agreed well with theoretical predictions.

Khiabani *et al.* [148] investigated the heat transfer in micro channels using cylindrical, solid particles. The parameters studied were the Reynolds number, particle to channel size ratio and the eccentricity of the particles due to heat transfer at the walls. This rather parametric study offered insight into the mechanism of heat transfer in micro channels, concluding that the suspended particles had great influence on the change in heat transfer. This was reported as well by Wu and Kumar [149] who used a double distribution function approach for velocity and temperature to simulate the heat transfer. They stated that the volume fraction played a dominant role for the heat transfer mechanism. Further studies by Yang [150], who used Al_2O_3 particles of 47nm in size, revealed a correlation between the Nusselt and Reynolds number for creeping flows.

Some of the applications and limitations of heat transfer within the LBM have been examined. It is currently still under active development [151] and 81.3% of the papers on heat transfer have only been published in the last seven years. Yet confidential thermal models are lacking. Another disadvantage of thermal models over the isothermal LBM is the fundamental non-stable computational nature. Not only physical correct models, satisfying the conservation laws, need to be developed but also high order numerical schemes need to be further investigated and improved.

3.6 Further reading: selected topics

To give a full overview of the applications of the LBM would be beyond the scope of this work. There are however emerging topics, which are worth mentioning, either due to their unique or practical nature. Some of them are attracting rapidly interest and are briefly discussed within this section. Additional literature can be found in [5], section 5.9.

3.6.1 Quantum mechanics

The migration of the LBM to the NSE was shown in section 1.6.2. In a similar manner, one can derive the Schrödinger equation from the Dirac equation. The Dirac equation comes from the field of particle physics and describes fields of spin-$1/2$ particles, such as protons, neutrons, electrons, neutrinos, and quarks. The Schrödinger equation describes the temporal evolution of a quantum state, where a state describes a position and momentum of a system. Such a system can be thought of a point mass or system of particles. The Dirac equation can be mapped to the LBE so that quantum mechanic phenomena can be simulated using the LBM, known as quantum lattice Boltzmann method (QLBM). The full derivation can be found in Succi and Benzi [152].
A classical quantum mechanics example is the Bose-Einstein condensate. This is a state of matter of dilute bosons, which are cooled close to $0°K$. At this state, a large amount of bosons have the lowest possible quantum state at which quantum effects become apparent on a macroscopic scale. Successful simulation of Bose-Einstein condensates have been performed by Palpacelli and Succi [153], Palpacelli *et al.* [154] and Succi [155].

3.6.2 Reactive flows

Reactive flows are the combination of several fluid dynamic disciplines. The basis is the LBM with extensions to allow chemical reactions to occur. In the case of a fluid-fluid interaction, multiphase capabilities need to be added and since most of the interactions of engineering interests are exothermal in nature, heat transfer needs to be added. There are, however, situations where the fluid-solid interaction is of interest. O'Brian *et al.* [156] investigated a fluid flow through porous media, allowing the fluid to react with the porous material (matrix) thus reshaping the geometry over time. Such flows are representative for magma flows in the earth crust where the rocks represent the matrix. They achieved overall good agreement with experimental data, despite the known limitations in modelling thermal flows. Gabrielli *et al.* [157] investigated micro channels with embedded barriers, for a more efficient reaction. Results showed that the barriers increased the effective reaction efficiency. Chen *et al.* [158] developed a velocity-temperature coupled LBM, which is capable of simulating steep density gradients. They validated their model against a benchmark case of a premixed propane-air flame and showed that their model gave improved results over a non-coupled LBM, where temperature and velocity are decoupled. Further stability was attested. Chen *et al.* [159] extended the LBM to allow combustion to occur. Comparison against experimental and numerical data showed excellent agreement. Novozhilov and Byrne [160] combined multiphase and compressible flows together with heat transfer to simulate explosions under the occurrence of natural convection. Successful simulations were presented, limiting the range of Rayleigh numbers where natural convection effects have a considerable influence on the explosion.

3.6.3 Bioengineering

Bioengineering has not only emerged in the field of the LBM but has in general created a research interest among scientists. The symbiosis of biology and engineering allows research to quantitatively look at biological phenomena for which experiments might either not be feasible or where the extra amount of numerical data would be a great asset for comparison and further in-depth studies. Representative studies in this field have been done by Sun and Munn [161] for example, where they investigated the blood flow of red and white blood cells, looking at the forces involved, trajectories and pressure changes. The simulations were consistent with experimental data available. Navidbakhsh and Rezazadeh [162] investigated the time evolution of infected blood cells by Plasmodium falciparum, which is responsible for the malaria illness. Typical effects observed by malaria infected cells are the adhesive properties of blood cells at the endothelial cells which are found on the inside layer of blood vessels. They used an immersed boundary method to simulate the membrane-fluid interaction, in order to capture the long time stiffening effect of the membrane. Results were conform with previous studies. Another application was studied by Hilpert [163], who investigated the travelling of bacteria bands along self-induced gradients.

3.6.4 Magnetohydrodynamics

The NSE have been successfully coupled with other physical disciplines that created a whole new area of studies, such as fluid-structure interaction, where the underlying governing equations for structural mechanics are simultaneously solved in conjunction with the NSE. A similar approach is the area of Magnetohydrodynamics (MHD), where the Maxwell equation for electric fields is solved together with the NSE. Hence, electrically conducting fluids can be analysed such as plasma, liquid metals, salt water or electrolytes. Martínez *et al.* [164] presented a simplified model, originally introduced by Chen and Matthaeus [165] and Chen *et al.* [166] for simulating MHD phenomena where the speed model and relaxation time scheme have been simplified for greater computational efficiency. Good results were obtained for the Hartmann flow test case which is a simple channel flow with a magnetic field at the walls. A more recent model can be found in Schaffenberger and Hanslmeier [167]. Chatterjee and Amiroudine [168] coupled the LBM with MHD and heat transfer capabilities where a double distribution function approach was selected to compute velocities and temperatures while the magnetic field was obtained independently through a so-called vector distribution function. The coupling of these distribution functions occurs on a macroscopic level where density, momentum, temperature and the magnetic field are coupled. Another combination of several regimes was done by Fogaccia *et al.* [169], where three dimensional electrostatic turbulence in thermonuclear plasma was investigated. Applications have been presented.

4 Development of a 2D lattice Boltzmann solver

This section is dedicated to the main purpose of this thesis; to develop a fluid solver based on the LBM capable of diverse mesh reading in two dimensions. This section does not only cover the code itself as part of the documentation but also elaborates on the CGNS format which is exclusively used in the code for storing mesh information. The mesh itself has to comply certain criteria which are highlighted in the subsequent sections.

4.1 The CGNS format

CGNS stands for "CFD General Notation System" and its purpose is to generalize and unify the way CFD data is stored. It has been created by NASA in collaboration with Boeing in 1994 and has attracted several other contributors worldwide ever since. It has been refined over the years and was passed onto a public forum in 1999, made up of international key representatives, both from government and industry, which is known as the CGNS steering committee.
This format has four key advantages which makes it highly suitable to use for own code developments:

1. It is open source
2. It is platform independent
3. It writes all information in binary, thus minimizing the file sizes and loading time
4. It stores all key information for a CFD simulation and can store auxiliary information

The fact that it is open source and platform independent makes it easy to distribute and user friendly. Being binary can be seen both as an advantage and disadvantage. It has a smaller file size but at the same time it cannot be opened and edited with a normal text editor, it needs a special binary file reader to look at the data. In the current version of CGNS 3.1.4 release 2 (31.07.2013), a binary file reader is implemented which can be installed as part of the distribution. The main advantage, however, is the capability to store information that otherwise would be spread over several files such as the mesh and output data.
The probably most important features of the CGNS format are storing the mesh coordinates and interfaces (1D/2D/3D, structured and unstructured) and the boundary conditions/position. It is further possible to store the flow solution to the file, making an extra output file obsolete while its binary nature ensures fast loading times and small file sizes. Furthermore, it can store information usually not saved in a CFD simulation such as: the used equations and units, convergence history, reference state, user defined data and some more features. The full list can be found in [170] and is a must have document when implementing the CGNS format into codes. At this point it is also worth mentioning the document "A User's Guide to CGNS" from Rumsey *et al.* [171] which gives a brief but descriptive introduction. Developers wanting to change the CGNS format in a way that suits them best will find the documentation about the SIDS (Standard interface data structure) very useful and can be found in [172]. The next section will clarify some aspects about CGNS which are necessary to know in order to understand the fluid solver's source code which has been

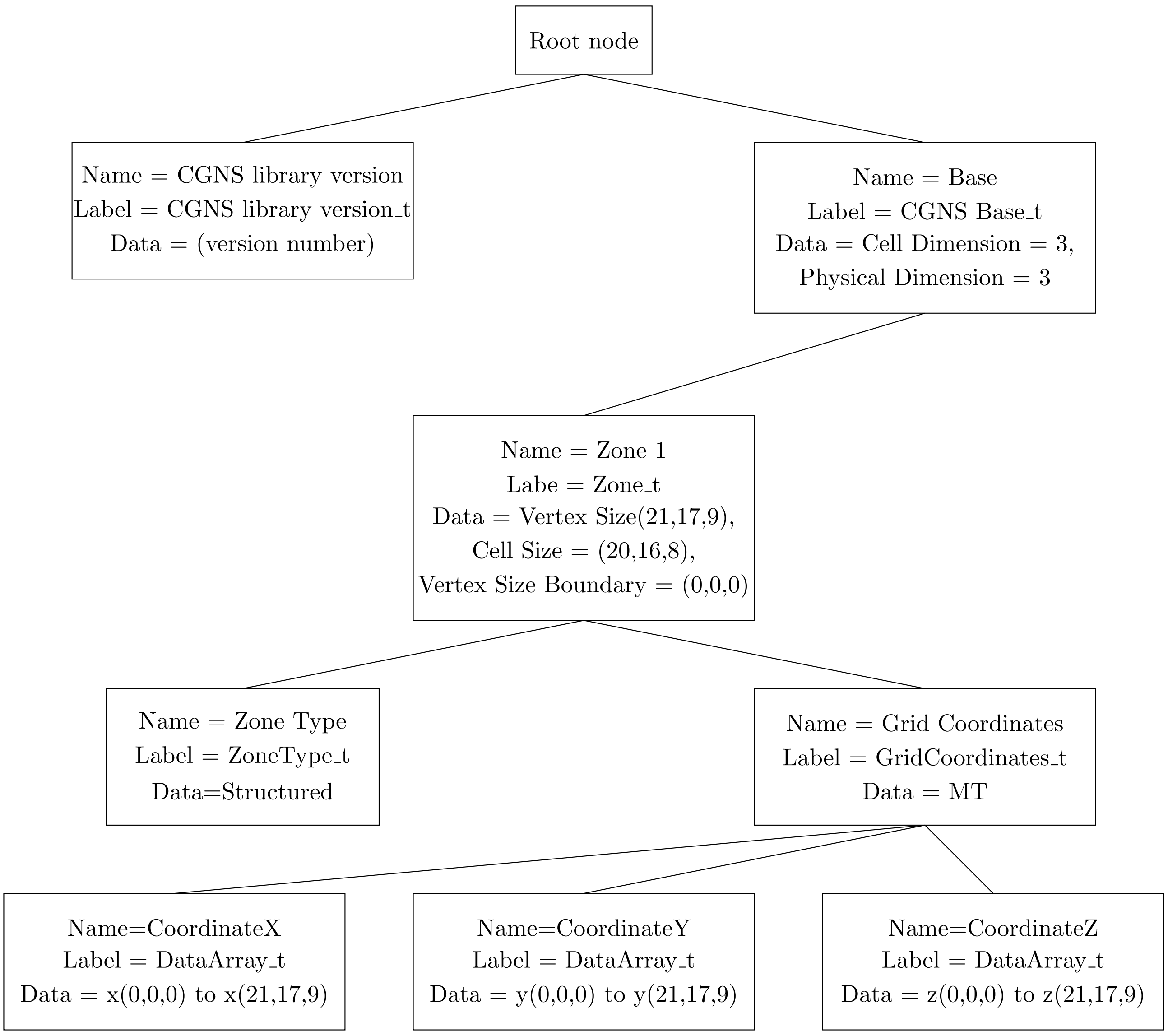


Figure 12 – Basic overview of the tree like structure of the CGNS format. Taken from [171, p. 11]

developed as part of this thesis. Appendix A holds information on how to obtain and install the library.

4.1.1 Basic CGNS structure

A typical CGNS file layout is shown in Figure 12. Each file is created in a top down hierarchy, starting with a root node. The tree structure best represents the file format as every information that is written to the file is linked to a node which is eventually linked through other nodes to the root node. The organisation of the nodes and their names is manifested in the SIDS. It is important to choose the names suggested by the SIDS when writing information to a CGNS file in order to create a "SIDS-compliant" file. This is necessary as variable names can be freely chosen.

If any other name than specified by the SIDS is used, the file may be processed by the software specifically written for this file but not by other programs. Since the whole purpose is to generate a unified file for CFD data storage, one should always create SIDS-compliant files. Although this is important, users only making use of CGNS files that have been created with an external program like a mesh generator do not need to take any precautions as the mesh generator will automatically create SIDS-compliant files. It is only important to know about this when reading the file in the code. Moreover, if a file is in accordance with the SIDS, it can be taken from platform to platform, from solver to solver and be read by any kind of program. This gives the flexibility to start a calculation with one solver which saves the intermediate results to the CGNS file and later resumes the same calculation from the point where it was left with another solver.

SIDS are low-lever calls and only need to be used when dealing with basic CGNS development. For a more user friendly environment, the mid-level calls (also referred to as application programming interface (API) calls) have been created to easily access the CGNS file. The list of all API calls are given in [170].

Returning to Figure 12, there are a few points that should be noted. First of all, every CGNS file has a root node, followed by a base and zone node. The base contains all the information needed to perform a CFD simulation such as the mesh coordinates and boundary conditions. The Zone node is only important when dealing with structured grids as each zone will be written into one zone. Using connectivity data, the zone's interfaces can be read and matched together. Under each zone node lays the core of the file information. The mesh coordinates and boundary conditions are saved here which are the minimum information required by any fluid solver.
The figure shows a file containing a structured grid of a simple rectangular domain with 21,17 and 9 points in x, y and z direction. The cell size gives the amount of cells in each direction. The coordinates are stored in CoordinateX, CoordinateY and CoordinateZ, which are all in accordance with the coordinate names specified in the SIDS. Boundary condition would be saved in the same manner and are not shown here for legibility.

4.1.2 Basic CGNS code

This sections aims to demonstrate a simple workflow with the CGNS format. The following shows a simple mesh reading in C++ using API's to communicate with the CGNS file. The API calls can be spotted by their prefix cg_, followed by its function name.

```
#include <stdio.h>
// include the CGNS library here
#include "cgnslib.h"

using namespace std;

int main()
{
   float *x, *y, *z;
   cgsize_t isize[3][1];
   int index_file,index_base,index_zone;
12    cgsize_t irmin,irmax,istart,iend;
13    int nbases, nzones;
14    int index_section,index_zone;
15    char zonename[33];
16
17    // open the CGNS file here, if it does not exist,
18    // return with error message and abort
19    if (cg_open("mesh_name.cgns",CG_MODE_READ,&index_file)) cg_error_exit();
20
21    // get number of bases and loop over them
22    // usually nbases == 1
23    cg_nbases(index_file,&nbases);
24    for (int i=0;i<nbases;i++)
25    {
26       index_base=i+1;
27
28       // get numbers of zones
29       cg_nzones(index_file,index_base,&nzones);
30       for (int j=0;j<nzones;j++)
31       {
32          index_zone = j+1;
33
34          // reading the zones and extracting its name
35          // iSize contains information as follows
36          // isize[0][0] = vertex size
37          // isize[1][0] = cell size
38          // isize[2][0] = boundary vertex size (unstructured always zero)
39          cg_zone_read(index_file,index_base,index_zone,zonename,isize[0]);
40
41          // set min and max values for coordinate array (local)
42          irmin = 1;
43          irmax = isize[0][0];
44
45          // allocate memory for coordinate arrays
46          x = new float[irmax];
47          y = new float[irmax];
48          z = new float[irmax];
49
50          // reading coordinates for each zone in x, y and z direction
51          cg_coord_read(index_file,index_base,index_zone,
52                   "CoordinateX",RealSingle,&irmin,&irmax,x);
53          cg_coord_read(index_file,index_base,index_zone,
54                   "CoordinateY",RealSingle,&irmin,&irmax,y);
55          cg_coord_read(index_file,index_base,index_zone,
56                   "CoordinateZ",RealSingle,&irmin,&irmax,z);
57       }
58    }
59    return 0;
60 }
```

In order to use the CGNS format, the library's header file has to be included as shown in line 3. Line 19 opens the actual CGNS file and checks if "mesh_name.cgns" exists. If not, the program will return an error message and abort. It will return the address of index_file if the reading is done successfully. The index_file is an ID, identifying each opened CGNS file within a code. By using the index_file, the code developer just need to open the file once and can use this information later to extract all information via this ID. Line 23 reads the amount of bases for the mesh and line 29 the amount of zones. In line 39, the actual size of the zone is read and stored into the isize array. It stores the vertex (number of grid points) size, cell size and boundary vertex size (which is always zero for unstructured meshes). This array is then used to allocate memory for the three coordinate arrays which in turn are read from line 51-56. The cg_coord_read API calls always start with the index_file, index_base and index_zone as part of the identification. The names CoordinateX, CoordinateY and CoordinateZ are the SIDS name for the coordination direction and can be read either with single or double precision. The array size is again determined by the isize array which has been used to initialize irmin and irmax. The actual coordinates are then stored into the x,y and z array.

Boundary conditions are read in a similar way, the only difference being that they are stored in sections. In its current version 3.1.4 release 2, there are several ways to write boundary conditions to the file. The effect is that different mesh generators will save it in a way that is likely to be different from another mesh generator's approach. Unfortunately that leaves the code developer with the decision to either implement subroutines for all mesh generators that should be supported which causes a lot of extra work or to choose on mesh generator which offers the greatest advantages. Clearly, the SIDS need to be more strict on this in future releases and should incorporate a best practice guideline which has to be followed to make a file SIDS-compliant[7].

4.2 A 2D lattice Boltzmann code

This section elucidates the main features of the current solver in its version 1.0 and states the current limitations. The last section gives a list of possible topics that can be easily implemented into the code based on its current state. For readers new to the LBM, a simple 2D example for the lid driven cavity using the BGK collision approximation, D2Q9 speed model and Zou/He boundary conditions has been included in Appendix B for learning purpose. This is a translated and improved version of the original Fortran code, found in [18].

First and for all, the code is based around the CGNS format as previously described. It has a fully unstructured mesh reading core which allows for unstructured domains to be read. Although this makes it possible to process unstructured meshes, the decision was made to implement the classic LBM on a structured grid to have a benchmark solver where improved versions of the LBM can be tested against. Furthermore, the unstructured LBM, also called the off-LBM, yields new challenges that will be addressed in section 4.2.3. More information on structured and unstructured grids can be found in Thompson *et al.* [173] who gives a complete overview on mesh generation techniques although the novel reader might find the books of Hoffmann ([174] structured and [175] unstructured) more appealing.
All three approximation for the collision operator described in section 2.1.1 to 2.1.3 have been implemented. While the single relaxation time and two relaxation time scheme are both speed model independent, the multiple relaxation time scheme has to be tailored to the used speed model. In this case, the D2Q9 speed model has been exclusively used.

[7]These thoughts have been forwarded to the CGNS steering committee

For the boundary treatment, the Zou/He boundary condition has been applied although the simplified version of the bounce back has been implemented as well. Since the bounce back is only applicable at the walls, the boundary treatment for the inlet and outlet are the same as in the Zou/He approach.
The results are written into an ASCII data file, readable from within Tecplot since the newest version of the CGNS distribution is only supported by the newest version of Tecplot and was unavailable during the code development. The aim is to switch at some point to the CGNS format to use the format to its full potential.
To give a user friendly entry into the program, which is based on a text user interface (TUI), a help mode has been implemented which takes the user through the menus, setting up the simulation. The TUI is shown below as it would appears when starting the solver.

```
-------------------------------------------------
   LBM V1.0 (C) 2013 by Tom-Robin Teschner
   Supervisor: Dr. Nikolaos Asproulis
   Contact: tomrobin.teschner@yahoo.de
   Master Thesis, Computational Fluid Dynamics
   Cranfield University, MK43 0AL
-------------------------------------------------

   main menu

   (1) load mesh
   (2) numerical set up
   (3) specify output
   (4) Initialize and set up domain
   (5) create parameter file
   (6) run simulation
   (7) activate help

   (8) exit
```

The workflow is from top to down. The mesh is loaded in menu 1 and then set up in menu 2. Everything from speed model to collision approximation and boundary condition is set up here. Menu 3 does only allow the output to Tecplot but is implemented for future use. Menu 4 initializes the flow field and sets up the domain while all the simulation parameters can be stored to a parameter file in menu 5. This file can be used to launch similar simulations and modification can be done using a simple text editor. Menu 6 offers either to run the simulation based on the current set up or to load a parameter file. If the current set up is used, the amount of iterations is specified and the auto save function switched on or off. Menu 7 brings up the help dialogue for each menu while menu 8 exits the program. Once the help function has been activated and the first simulation has been done, the menu structure should be familiar to the user and the workflow clear and easy to understand. Appendix C gives detailed instructions on how to create a mesh in ICEM CFD.

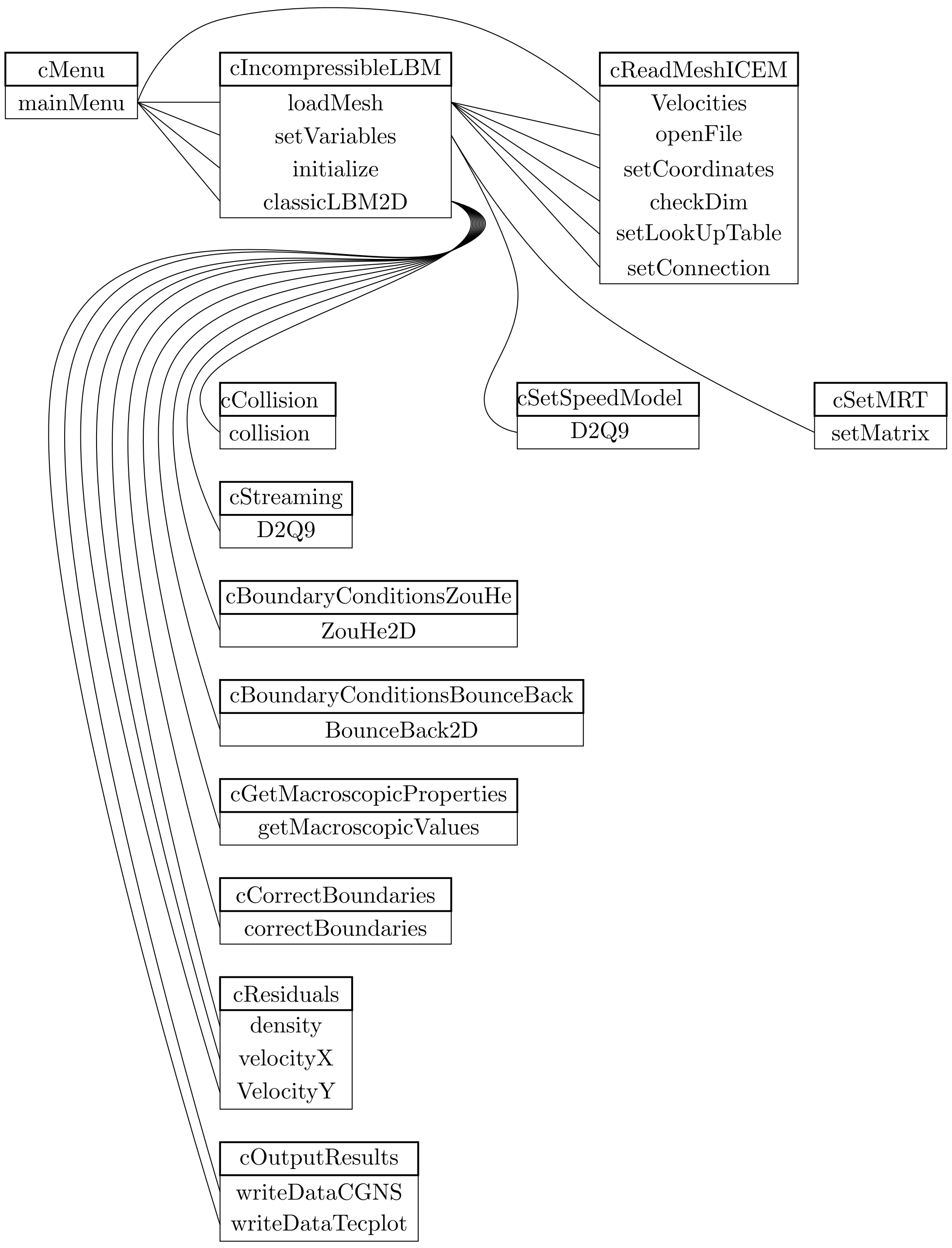


Figure 13 – Schematic drawing of the code structure. Only classes and their most important methods are shown while attributes are omitted.

4.2.1 Code structure

The schematic sketch of the code is shown in Figure 13. Against normal convention, only the classes and their methods are shown while the attributes are omitted for legibility. Furthermore the instances of the classes are not shown but rather their interaction between the classes. The code makes use of only one instance per class, hence this simplification is plausible.
The schematic only shows the interaction between classes so the main.cpp file is not shown. The main.cpp file only calls the cMenu class and the calculation is done from there. The cMenu class sets up the simulation and sends the necessary information to the cIncompressibleLBM class which in turn loads the mesh trough cReadMeshICEM and sets up the domain. cMenu accesses the cReadMeshICEM class as well as it needs to know the names of the boundaries and their type in order to correctly set up the velocities. The last method of cIncompressibleLBM, classicLBM2D performs the actual calculation and calls the classes connected to this method. The classes have been presented in the same order the code would call them, starting from the top and going downwards. The boundary conditions have been split into two different classes and depending on the setup, only one of them is going to be called. This decision was made to keep all classes short and readable. The collision class for example contains all three collision approximation (single, two and multiple relaxation time scheme) due to its short mathematical description. While the cResidual class calls all its member for each iteration, the cOutputResult class is called during auto save, after the simulation has converged or reached the maximum amount of iterations and only one method, depending on the user input in mainMenu, is going to be executed.
Although it has been tried to program all classes in a way which is self explanatory and easy to understand, some classes require further explanation which is given in Appendix D for some specially picked methods.

4.2.2 Current limitations of the code

The current release of version 1.0 comes with a fully functional fluid solver which has however some limitations which should be addressed here.
The only tested and successfully used mesh generator is ICEM CFD 14.0. A second class for reading meshes created in Pointwise has been included (source code) but has not been implemented in the code as some severe drawbacks in the output of the CGNS format did not allow meshes to have more than one zone. The details are omitted and the choice in favour of ICEM CFD was made since it stores all information to one zone.
As the only implemented method is based on the classical LBM, the only supported elements are quad4 and bar2 elements for the interior and boundary. Unstructured and 3D meshes would require at least tri3, tetra6 and hexa8 which can be easily implemented, see next section.
At this point it should be also noted, that the classical LBM uses almost exclusively Cartesian meshes in the literature. Internal walls are resolved by the immersed boundary method (IBM) which resolves curved boundaries as stair steps. The fluid solver proposed in this thesis however makes use of surface meshes and internal walls create recesses. This means that the mesh has to go around the geometry on the inside of the domain which, when using curved boundaries, causes skewed elements in the vicinity of the geometry. This curvature has to be kept to a minimum so

that mesh generation requires some further attention. In general, the less curvature encountered, the better the results. The reason for that being the weights in the equilibrium distribution function as the distant to the surrounding nodes alters due to curvature.
Another point related to curvature is that the LBM is a local approach. This means that the domain cannot change its direction as it is created. A simple example of a domain which changes direction would be a channel flow with curvature, say a S-duct. Although the flow physics are correctly reproduced, they are outputted in the wrong way. The calculated velocity has no information which (global) direction it is going and just as for the straight channel flow, the fluid assumes that the flow direction is always in positive velocity inlet direction. Simulating the S-duct or straight channel makes no difference, the results will be the same and there is no accounting for the curvature. This is the same phenomenon as with structured grids on normal NSE based solvers. The structured meshes have to be brought from physical into computational space. This means that the results from the LBM have to be brought from computational space into physical space after the calculation is done. Since this is currently not available it is best practice to have a rectangular north, east, south and west boundary as this will not give any problems.
There are currently only three boundary conditions available, the stationary and moving wall, inlet and outlet. As encountered in section 2.3.3, corners have to be modelled as stationary points. This restricts the use of inlets, outlets and moving walls at corners and only leaves the stationary wall as an option for this area. A simple channel flow with west: inlet, north=south=wall and east=outlet will not make any problems as the solver will automatically detect the wall in the corner and assign them the corner nodes. A problem will be created if two inlets or two outlets or the combination of both inlet and outlet are used together, next to each other. The singularity where both boundary conditions meet cannot be modelled and therefore should be avoided. A small fragment of a wall has to be implemented instead.
The last point has been already mentioned before and concerns the CGNS format itself. Currently there is no output to CGNS but to an ASCII Tecplot data file. It is favourable to switch from ASCII to binary when this becomes an option.

4.2.3 Possible code extensions

There are several topics that can be easily implemented based on the opinion of the author. The code has been written in a way that makes it easy to understand and extend. The next steps should be to implement one of the following features.

4.2.3.1 Three dimensions One of the obvious extensions is to bring the current version to 3D. This is one of the easier tasks as it is basically a copy of the 2D version which requires only a new speed model for 3D and a new streaming process, which however is exactly the same as for 2D and just needs some modification. The same is true for the boundary conditions which need to be updated for 3D. Furthermore the 3D element hexa8 needs to be implemented into the mesh reading class, this can be however copied from the currently existing quad4 method and needs some modification for the look up table that connects the grid information together, as it is still an unstructured mesh reader.

4.2.3.2 Off lattice-Boltzmann method The off-LBM method in 2D/3D makes it possible to simulate fluid flows on unstructured meshes. It uses a modified LBM and there are several methods available, although the off-LBM after Bardow *et al.* [176] is a very good starting point. Further literature related to off-LBM and their implementation techniques can be found in [177–179]. The advantage is that two and three dimensions are modelled by the same equation although the boundary condition and streaming still need to be updated individually. The introduction of the off-LBM requires the approximation of derivatives which means that the approach is not local anymore. There are two topics which need to be thought of carefully and how to treat them. The first being the speed models as the amount of neighbour nodes can change form node to node and the fact that the cells of greater size will march with a different time step than smaller cells. One solution could be to introduce a steady state LBM, as for example in Yan and Yan [180] or Bernaschi *et al.* [181].

4.2.3.3 Multi-speed models Multi-speed models would greatly enhance the accuracy of simulations. While the standard D2Q9 for example struggles to resolve vortical flows, multi-speed models such as D2Q25 derived by Lycett-Brown *et al.* [26] are fully Galilean invariant and therefore more suitable for such flow types. It would be better to implement the off-LBM and the 3D classical LBM first before implementing multi speed models, in order to implement suitable multi-speed models into all solvers.

4.2.3.4 High performance computing Although the LBM is very fast, eventually the need arises to bring the code into a parallel environment, especially since the code is an explicit, unsteady approach, limited to a CFL number of unity which gives a slow convergence rates. Another possibility would be to parallelize the code using CUDA. As for the multi-speed models, the same is true tor parallelizing the code. It should be done at a stage where more solvers would be affected and hence the benefit of having an parallel version would affect all solvers.

4.2.3.5 Physical models At his stage it is possible to implement further physics into the solver. The topics discussed in the literature review in section 3 should be implemented in the long run and multiphase is one of the models which is probably the most easiest to implement yet the most effective model as this demonstrates clearly the advantage of the LBM over NSE based solvers. Apart from multiphase, other physical models like porous media or turbulence can be implemented although for turbulence the mesh reading should be improved first in order to be able to resolve the boundary layer. This requires non equidistant cells which in theory could be accounted for by adapting the weights throughout the domain or by using the off-LBM which could process structured elements in the boundary layer and unstructured elements everywhere else. Information on how to adapt the weights locally can be found in Succi [182].

4.2.3.6 Boundary conditions Another extension could be the implementation of different boundary conditions. There are several popular choices of boundary conditions in the LBM of which some of the popular choices are the Zou/He, Inamuro, Regularized, FD and nonlinear FD which are discussed in Latt *et al.* [183]. The implementation should be straight forward but careful thoughts should be spent on how to implement the boundary conditions on unstructured meshes.

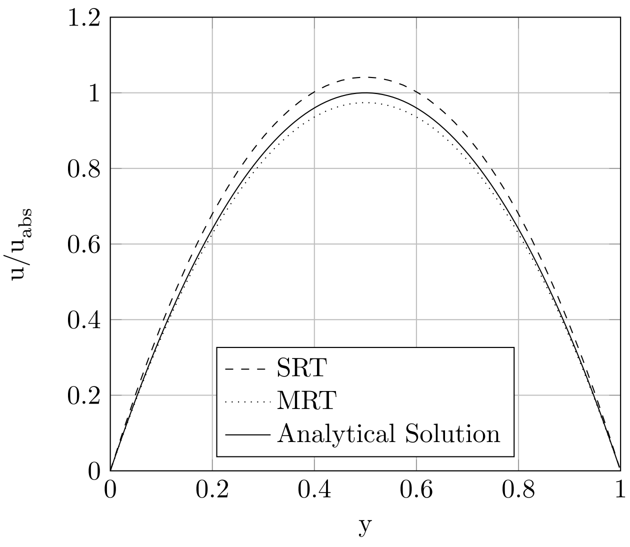
5 Code validation

The following section examines how the code developed for this thesis compares against established CFD test cases. The test cases are a channel flow, the lid driven cavity, the backward facing step, the sudden expansion and the flow around a cylinder. Each case has been tested with the present code and the results are given in the following sections.
In a high fidelity CFD simulation, a grid dependency study is mandatory to determine a size where the error induced by the grid is negligible. Ferziger [184] and Ferizger and Perić [185] give and introduction into estimation of numerical inaccuracy while Roache [186] introduced the grid convergence index, short GCI, which is an effective measurement of the grid induced error. Celik *et al.* [187] expanded this method and proposed a revised version of the GCI which is more sophisticate yet offers great stability and accuracy.
The implementation of the GCI is favourable but in the framework of the LBM not practical due to its inherent instability that comes along with changes in key flow parameters, such as viscosity and velocity. It is a well known problem for example that the viscosity is limited by a lower bound. Simulations have shown that not only a lower bound exists but an upper bound might exist alongside, which gives an operating range for viscosity that may differ for different collision models. Furthermore, the boundary conditions contribute instabilities through the choice of the inlet velocity. Looking at eq.(2.19), it is clear that a choice for the velocity of unity yields division by zero. The rule of thumb is to set the velocity as $u_i < 0.1$ although higher velocities may still work well for some cases. This is again depending on the collision approximation and different collision models tolerated different velocity ranges.
The meshes for a proper mesh dependency study however require a subsequent mesh refinement, having 1.4 - 2 times more cells then the preceding mesh. As discussed in section 2.4.7, a change in mesh size changes the Reynolds number as well. To achieve the same Reynolds number for all meshes, parameters of velocity and viscosity need to be adjusted which cause the parameters to slip out of the operating range causing instabilities. These instabilities influence the result too much in order to get a useful measurement on the grid induced error. Hence, to find the optimum grid size, one needs to adjust the mesh size, velocity and viscosity in a trial and error process to find a working mesh that is not diverging, yet not yielding any noticeable instability. This mesh will then most likely produce the most accurate results.

All the validation cases have been run using the single relaxation time (SRT) scheme, or LBGK, and the multiple relaxation time (MRT) scheme. The two relaxation time (TRT) scheme failed to converge for all cases and behaved extremely unstable. It is therefore left out of the validation process. It is believed that the instabilities arise due to the choice of the boundary conditions and hence different boundary conditions could counteract this behaviour.

5.1 Channel flow

The channel flow offers an analytical solution for the laminar, parabolic velocity profile which makes it an ideal benchmark case for validation. The investigated channel had an aspect ratio of 1:20 and simulations have been done using a Reynolds number of 82 and a uniform mesh size



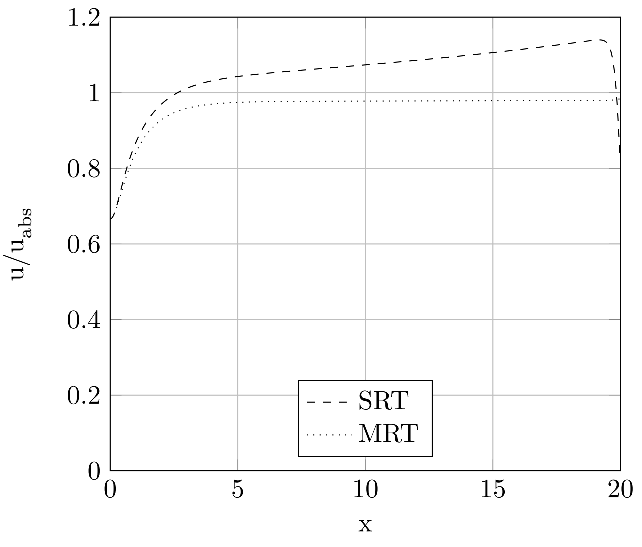


Figure 14 – Velocity profile of the fully developed channel flow (left) and the development of the maximum velocity at the channel symmetry line along the channel (right).

of 41 x 800 nodes. Durst *et al.* [188] carried out a literature survey looking at different formulae to calculate the development length of laminar channel and pipe flows and derived an empirical formula as:

$$\frac{l_d}{d_c} = \left[0.619^{1.6} + (0.0567Re)^{1.6}\right]^{\frac{1}{1.6}} \tag{5.1}$$

which has been based on previous studies of different authors. For a Reynolds number of 82, the development length is $l_d/d_c \approx 4.8$ and the velocity profile shown in Figure 14 corresponds to this section. While the SRT collision approximation yields 4.1% more maximum velocity, the MRT produces a maximum velocity which is 2.6% lower than the analytical solution. Looking at the shape, the MRT produces a much closer fit to the analytical solution which is backed up by a 4.5 times lower mean square error (MSE) compared to the SRT. The right hand side of Figure 14 shows the development of the maximum velocity in stream-wise direction at the channels symmetry line. Albeit both methods show rapid decrease in the downstream velocity gradient with its lowest value at $l_d/d_c \approx 4.8$, the SRT's maximum velocity continues to rise which causes an over prediction further downstream. If this rise in velocity would not be accounted for at the outlet region then the conservation law of mass would be violated. Hence, a sudden decrease can be seen at the channel's end which conserves mass but at the same time creates a non-physical flow field. The MRT on the other hand does not rise once it has reached its fully developed profile and stays constantly just below the analytical solution. Since no further change downstream is encountered, the fluid exits at the outlet without creating a non-physical flow field. Figure 15 shows both the outlet for the SRT (left) and MRT (right).

As can be seen for the SRT, the streamlines are deviated to the walls as if the channel would en-

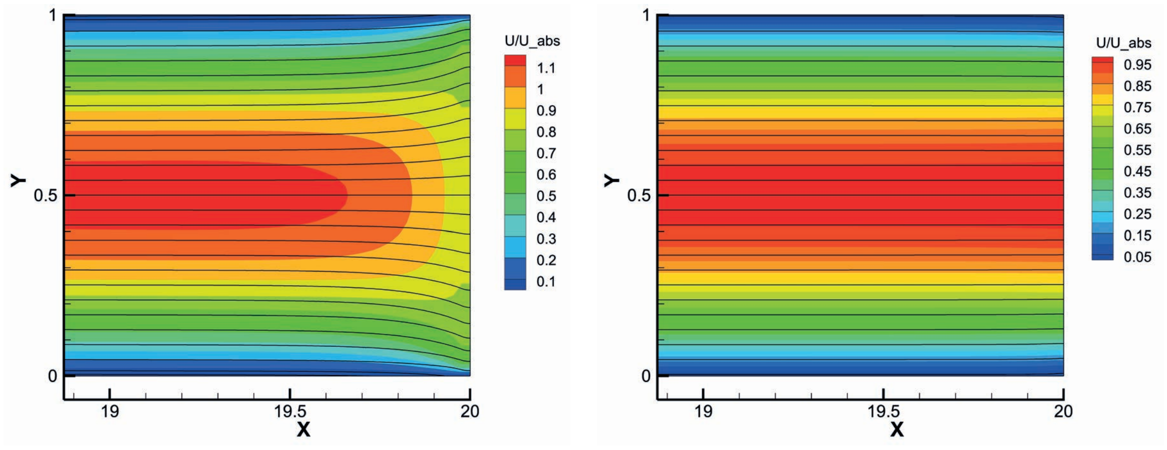


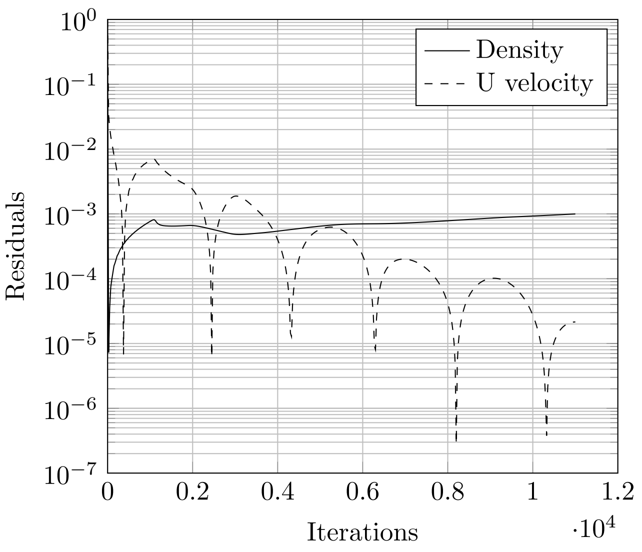
Figure 15 – Channel outlet for the SRT (left) and MRT (right) collision approximation. Velocity contours normalized to the analytical solution.

counter an expansion just before the outlet. Streamlines close enough to the wall are forced to exit the domain due to the pushing streamlines from the channel inside to ensure the conservation of mass. The MRT does not have this problem as discussed before which can be seen from Figure 15 on the right hand side. Minor streamline changes at the outlet are due to the instability created by the boundary.

Figure 16 shows the convergence history of both the SRT (left) and MRT (right) for density and stream wise velocity. The velocity component normal to the flow direction has not been included since the flow is dominated by the downstream velocity. They are calculated as:

$$R_i = \frac{\sum \varphi_i^{n+1} - \sum \varphi_i^n}{R_1} \tag{5.2}$$

where R_i is the residual for density and velocity, φ_i^n the surface integral for rho and velocity at time step n and R_1 the residual at the first time step which is used for normalization.
The SRT took about 11 000 iterations to converge below 10^{-4} while the MRT took about 100 000 iterations to reach the same level. Not only does the MRT take more iteration to converge but also more time per iteration compared to the SRT which is explained by its more complex nature. Thus, simulation times increases drastically for the MRT although qualitatively better results were obtained. The density converges below the predefined level for the MRT but fails to do so for the SRT. For simulations left running for more than 11 000 iterations, the density increases which is due to the problem at the outlet, causing unrealistic results for the density in the domain at a sufficient amount of iterations. The MRT on the other hand does not exhibit this behaviour demonstrating its robustness.
Another feature seen in the convergence history of the LBM is the wave like behaviour of the residuals which have been observed for all simulations. The nature to this behaviour could not be fully explored and hence lack of understanding prohibited implementing convergence acceleration



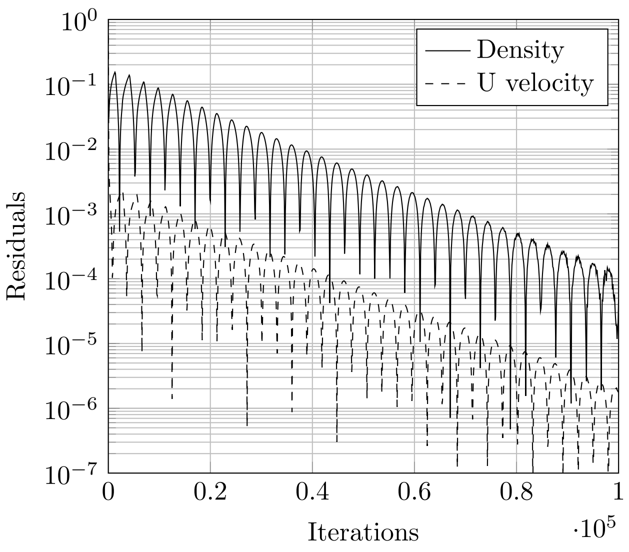


Figure 16 – Residuals for the channel flow for the SRT (left) and MRT (right) collision approximation.

methods. Since the collision operator relaxes at a certain time level with a collision frequency ω, one could think that a miss match in collision frequency and physical time step (always unity) could yield this behaviour. When changing the collision frequency to unity, this behaviour persists and hence cannot be the root cause.

The discussion on the convergence behaviour is done exemplary here for the channel flow but is applicable to all other simulations in terms of convergence rate, speed and behaviour.

5.2 Lid driven cavity

The lid driven cavity is another popular choice to validate CFD codes. Ghia *et al.* [190] performed simulations on the lid driven cavity for different Reynolds numbers using a stream function approach. Results of the velocities in the middle sections for the SRT and MRT are shown in Figure 17 and the results of Ghia *et al.* are shown for comparison. Another computation has been performed by Perumal and Dass [189], using the LBM and the SRT and MRT scheme. The speed model has been the same D2Q9 used in the current investigation while the mesh had 201 x 201 nodes and was slightly coarser than the mesh used for this calculation with 320 x 320 nodes. The results have been plotted as a reference.
As can be seen, not only does the SRT and MRT match the Navier-Stokes based results of Ghia *et al.* but almost no disagreement can be attested between the both LBM approaches. The only change is due to the different mesh size and the error of digitalizing the data.
The MSE shows excellent agreement with the reference data of Ghia *et al.*, both the SRT and MRT perform equally well for the u_y velocity component while the u_x component is calculated more accurately by the MRT by a MSE factor of two compared to the SRT. It has to be taken into account that the reference data was obtained with a stream function approach and is by no

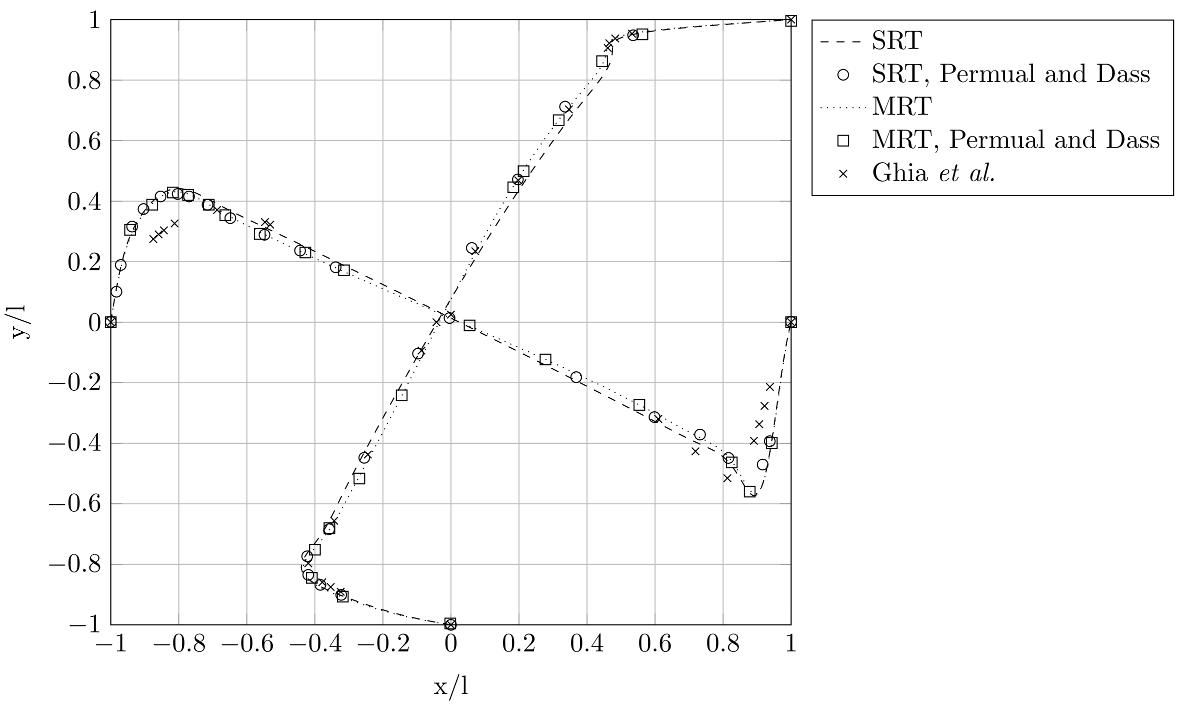


Figure 17 – Velocity profiles at the middle sections of the lid driven cavity. SRT and MRT shown compared against another lattice Boltzmann solver after Perumal and Dass [189] and a stream function approach after Ghia *et al.* [190].

means without errors. The order of the discretization error was not mentioned but the mesh size with 257 x 257 nodes. Experimental data by Prasad and Koseff [191] is available which is showing significant discrepancies. It is believed that the presence of turbulence is causing this difference and hence reference data without turbulence has been used for validation. This experimental data shows the same discrepancies with the current reference data although the reference data used for this validation process has been validated by the authors. Hence, comparisons have not only been made to a NSE based solver but also to a LBM based solver to ensure its functionality with both reference data showing good agreement.

A visual representation in terms of streamlines is given in Figure 18 and shows the SRT and MRT. The flow physics are correctly resolved, the primary center vortex is resolved alongside with its secondary vortices in the bottom left and right corner. For $Re \gtrsim 1000$, a forth vortex starts to form in the top left hand corner which has been captured by both collision approximation schemes. The MRT needed 1.63 times more iterations than the SRT to converge. This ratio was 1:10 for the channel flow and shows that the amount of iteration is of the same order which is explained by the outlet problem which forced the interruption of the simulation.

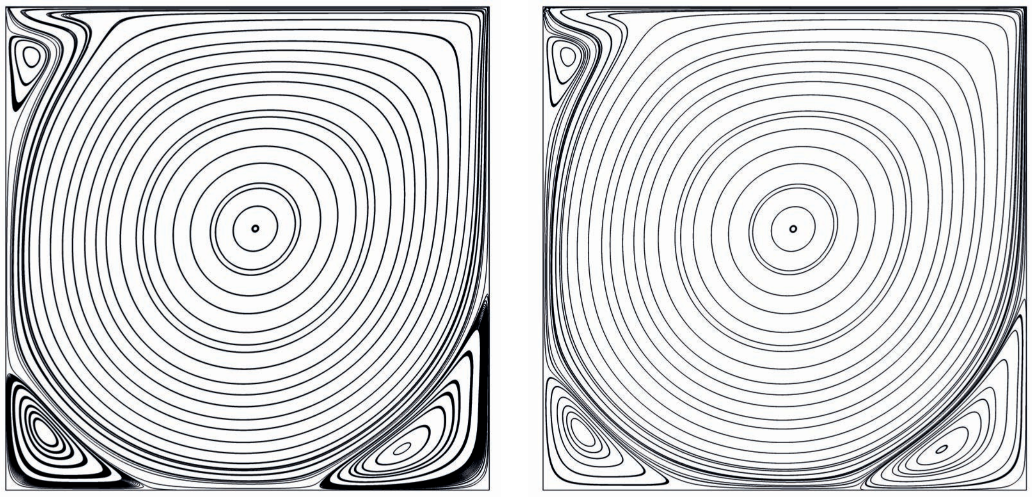
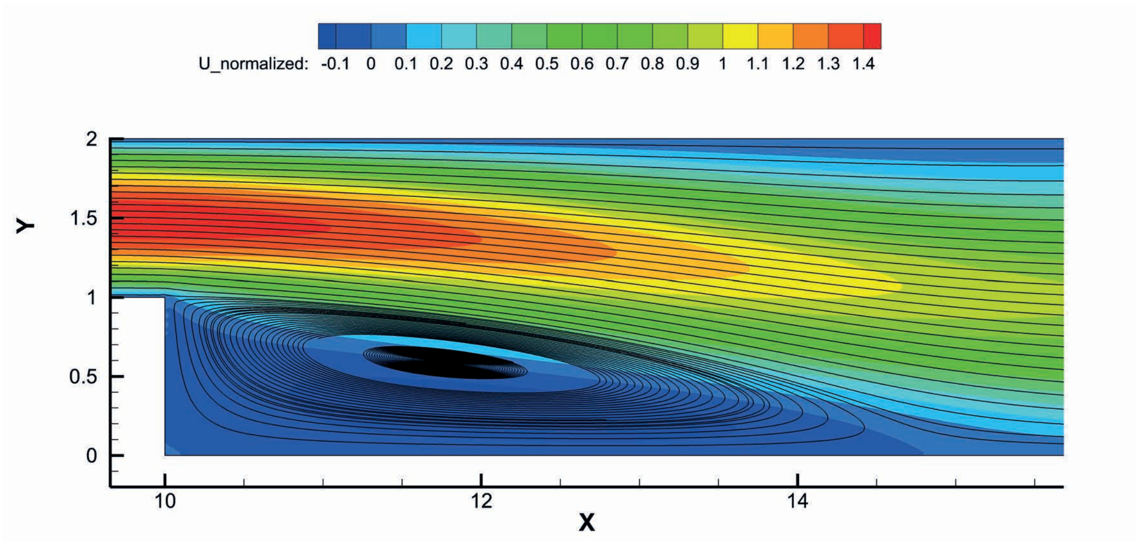


Figure 18 – Streamlines for the lid driven cavity for a Reynolds number of 3200, using the SRT on the left and MRT on the right.

5.3 Backward facing step

The next validation case is the backward facing step and simulations are compared against experimental data available from Armaly *et al.* [192]. The simulation has been done at a Reynolds number of 100. The small channel upstream of the step had the same height as the step height and it was assured that a laminar, parabolic velocity profile has been reached at the orifice. The detailed geometry can be found in [192].
Figure 19 shows the streamlines for both SRT and MRT simulations. The visual difference in both cases is close to none and the velocity magnitude, normalized by the inlet velocity, compares well for both cases. Armaly *et al.* provides measurements of the reattachment length. The SRT over predicts the reattachment length by 36.6% while the MRT performs only marginal better with an over prediction of 34.3%. This substantiates the observation seen for the lid driven cavity that only minor differences exist for the collision operators. Unlike the the lid driven cavity, where excellent agreement has been achieved, the back ward facing steps produces a substantial over prediction which is also shown in Figure 20. The velocity profiles have been normalized by the inlet velocity and the comparison have been done to the experimental data of Armaly *et al.* The parabolic velocity profile just at the orifice can be seen which then redevelops due to the recirculation zone. The peaks of maximum velocity differ for simulation and experiment and a bigger vortex size is expected due to the shift. The angular velocity of the vortex is imposed by the passing fluid on top and is therefore fixed. The moment of inertia on the other hand changes with the vortex size and therefore the simulation and experiment experience a different angular momentum. This energy is fed back to the flow, allowing it to stay longer on top and causing an over prediction. Part of this extra energy comes from the simplification of a two dimensional case. The three dimensional effect such as the side wall separation are neglected, a detailed discussion on them and their influence can be found in Malamataris and Löhner [193] and Williams and Baker [194].



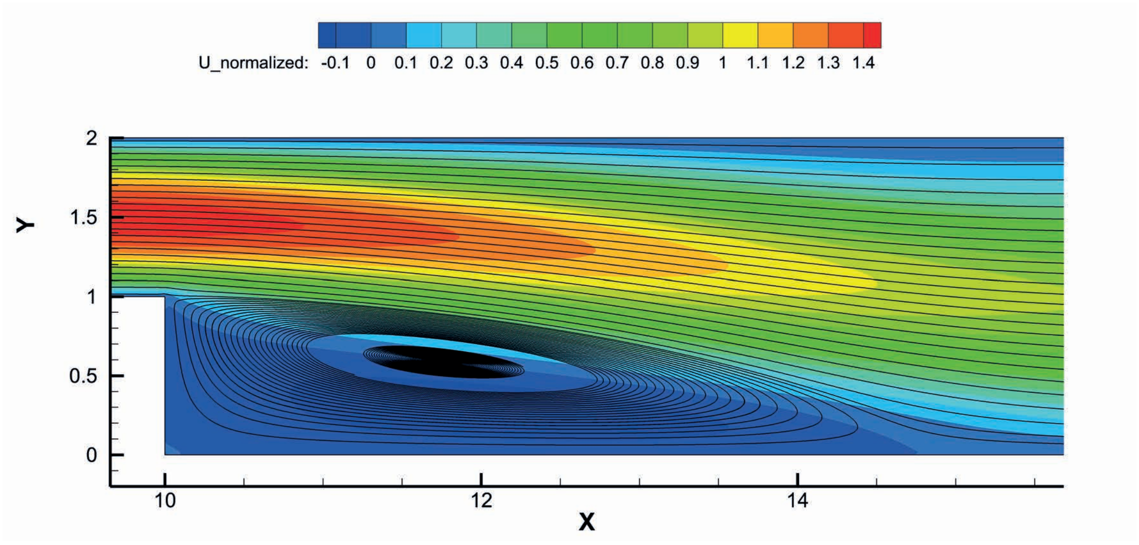


Figure 19 – Streamlines for the backward facing step for a Reynolds number of 100, using the SRT on the top and MRT on the bottom.

Since the collision approximation gave a consistent over prediction for both models, it is believed that this error stems from the use of boundary conditions and speed model. The standard D2Q9 speed model lacks Galilean invariance (see Lycett-Brown *et al.* [26, p.1227]) and accordingly could be a potential source of errors for vortical flows.

5.4 Sudden expansion

The sudden expansion is very similar to the backward facing step, having two steps symmetrically aligned at the orifice. As for the backward facing step, the step height has the same size as the small channel diameter and the length has been chosen in such a way that a fully developed velocity profile has been reached at the orifice. Experimental data is available from Fearn *et al.* [195]. The

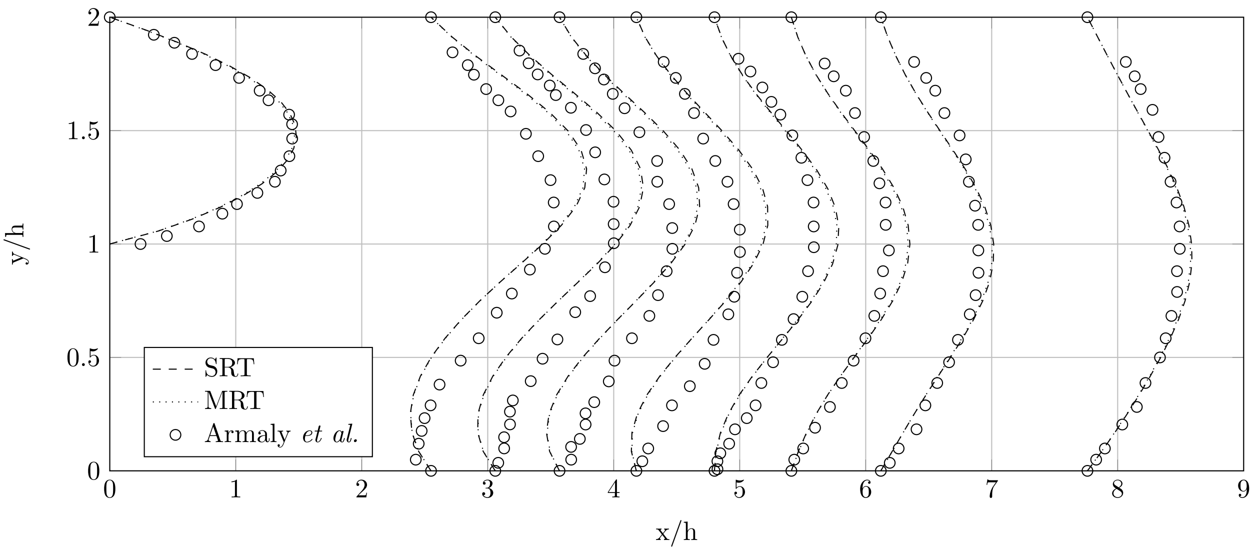


Figure 20 – Velocity profiles for the backward facing step at Re=100. X and Y direction normalized by the step hight h. Reference data taken from Armaly *et al.* [192].

backward facing step shows some minor flow changes and the appearance of secondary vortices as soon as the Reynolds number is increased. The sudden expansion behaves in a similar fashion, having an unsymmetrical flow field after passing a critical Reynolds number. Therefore it has been chosen as an other validation case, simulating the flow at Reynolds numbers Re=25 and Re=80.

Figure 21 shows the streamlines for the SRT (left) and the MRT (right) at a Reynolds number of 25. The flow field is perfectly symmetrical but an over prediction of the maximum velocity can be seen at the small channel just before the expansion. The velocity has been normalized by its inlet velocity and hence a value of 1.7 would not be expected. This behaviour has been observed for the channel flow where a continuously rising maximum velocity has been reported. Tabble 6 states the reattachment length for the SRT and MRT compared to Fearn *et al.* According to the results, the SRT under predicts the reattachment length by 40.9% but still performs better than the MRT which results in 50.1% difference. Similar to the observation made for the backward facing step, this time the increased outlet velocity at the small channel is causing an increased angular velocity which increases the angular momentum. This extra energy is yet again responsible for the bigger vortex. If the outlet velocity had been predicted correctly by the SRT than it would be expected to yield results worse than the MRT and the over prediction in outlet velocity only works in favour of the SRT.

The Streamlines for a Reynolds number of 80 are shown in Figure 22. At Reynolds numbers above Re=33 the symmetric bifurcation is broken down into an unsteady flow pattern [195]. Depending on initial perturbation, the flow will either demonstrate a lower or upper, larger recirculation area. Due to the presence of the lower, greater vortex, the flow can no longer expand freely as in the

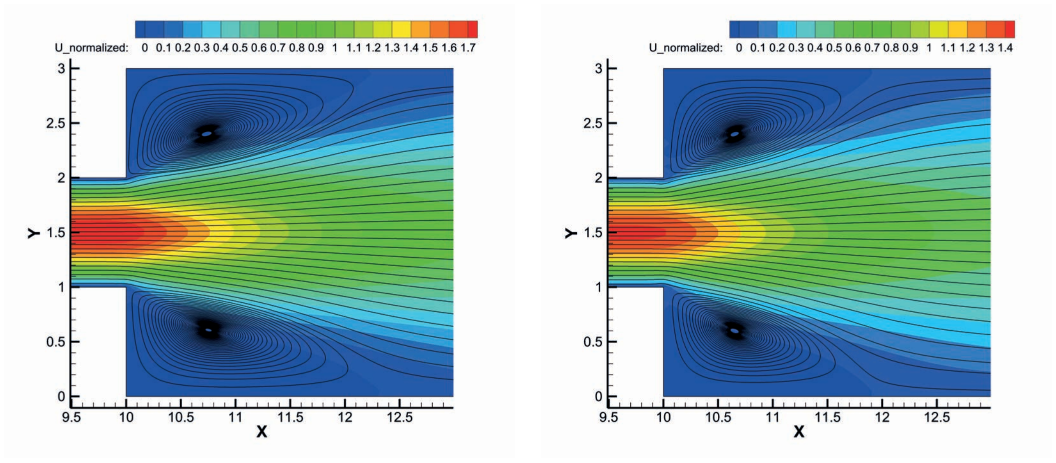
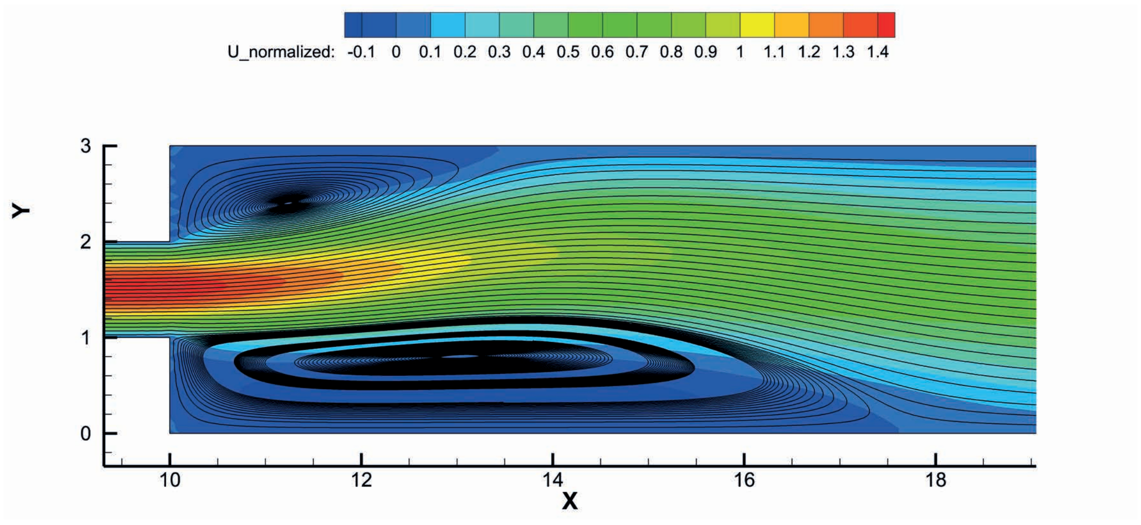


Figure 21 – Streamlines for the sudden expansion for a Reynolds number of 25, using the SRT shown on the left and MRT shown on the right.

symmetrical case. The upper, smaller vortex is pushed against the wall as the pressure is increased, forcing it to expand, resulting in a greater recirculation area than for the symmetrical case. Both collision models are able to reproduce this effect, the SRT under predicts the recirculation length by 15% while the MRT does not show any different result to the experimental data. It should be noted that the SRT correctly predicts the outlet velocity this time which raise concern in the liability of the model and its applicability. The root cause to this random behaviour could not be fully explored.

The lower vortex is predicted with 39.2% deviation for the SRT and 44.8% for the MRT. The physical representation of the vortex is physical and the residuals criteria have been the same for simulations. Furthermore the computational domain consists of the same or more cells than comparable numerical simulations in the literature and therefore the error is to be expected to have its origin somewhere else. Examining Figure 22 closely reveals that the vortex for the MRT has a flat top where the exiting fluid of the small channel passes. The SRT has a slightly more curved top which transports the fluid a little further downstream causing better agreement in the reattachment length. Thus, the error seems to be due to the incorrect prediction of the vortex size which has been argued for the backward facing step as well. A change in speed model and boundary condition could yield better results but still discrepancies could arise. Just by looking at the pressure gradient development normal to the flow, i.e $\partial p/\partial y$ it seems that this gradient is over predicted in the vicinity of the vortex center causing the velocity to slow down to fast which causes the flat plateau for the MRT. Less pressure gradient would increase the crossflow component with more curvature at the top which would increase the moment of inertia of the vortex. With the same angular velocity, the angular momentum would be increased resulting in a greater vortex size and advection of the fluid with closer agreement to the experimental reattachment length. The added curvature on top of the vortex causes the center to shift downstream.

The origin of this problem could be in the linear scaling of pressure and density. As seen by



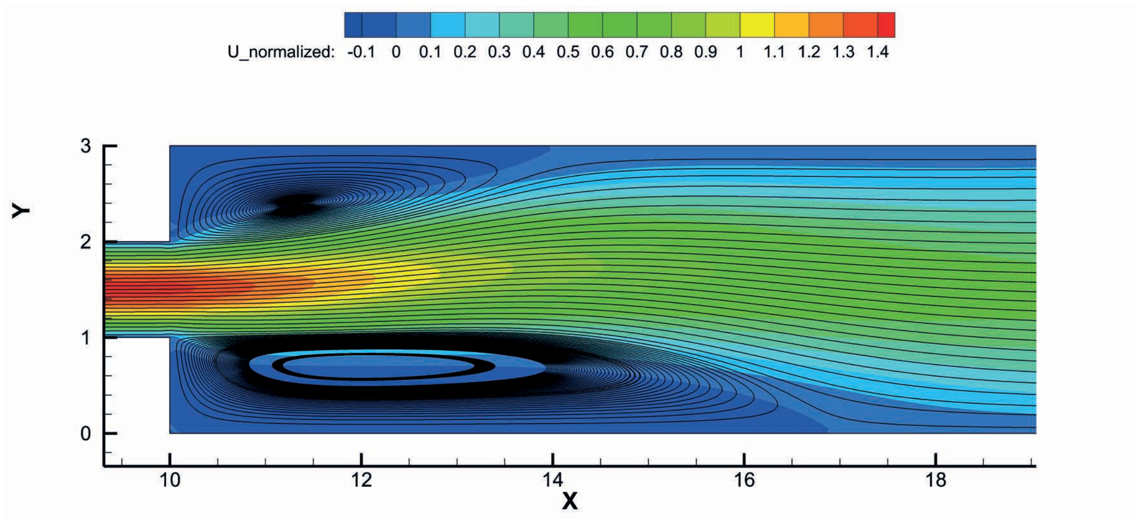


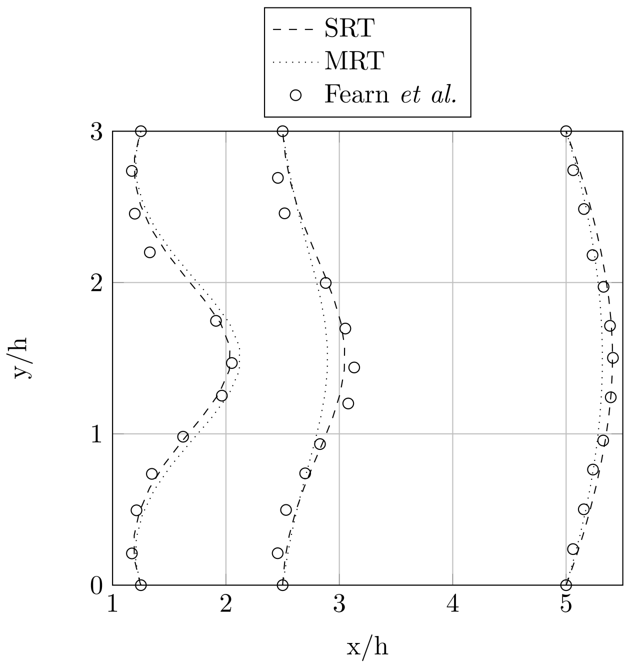
Figure 22 – Streamlines for the sudden expansion for a Reynolds number of 80, using the SRT shown on the top and MRT shown on the bottom.

the equation of state in the LBM, eq. (2.21), pressure is scaled by a constant speed of sound. This means that whenever a pressure gradient is present, the density is variable and changing, even for an incompressible flow. The problem with this deduction could be explained by an adiabatic, incompressible channel flow. It is know that a pressure loss and hence a pressure gradient must be present. For given conditions, the density must stay constant which causes a problem for the incompressible NSE. They have to use different techniques to recover the pressure field as the equation of state can no longer be used. While the pressure changes, the density and temperature stay constant rendering the equation of state useless. Doing the same simulation with the LBM however yields a change in density, even for low speed, incompressible flows and the pressure gradient can be obtained directly. This demonstrates a difference in both approaches which ultimately causes different results in density. Although the main flow features are recovered, the rather troublesome treatment of the density - which is used to derive all other macroscopic

Table 6 – Reattachment length for the sudden expansion for Reynolds number of Re=25 and Re=80 and for the top and bottom vortex. All values normalized by the step height.

	Top		Bottom	
	Re=25	Re=80	Re=25	Re=80
Fearn *et al.*	3.83	4	3.83	12.5
SRT	2.3	3.4	2.3	7.6
MRT	1.9	4	1.9	6.9

quantities - could be the cause for the incorrect pressure gradients. The pressure gradient directly influences the velocity field through the Bernoulli equation for incompressible conditions and hence vortical flow features may not be correctly reproduced due to the aforementioned discussion.
By no means should that mean that the LBM is not suitable to calculate fluid flows, as has been demonstrated so far the results showed good agreement with reference data but it should be noted at this point that there are some shortcomings as well which, under given circumstances, could jeopardise results and explain the difference of up 50% disagreement in the reattachment length.
Figure 23 show the velocity profiles at sections of $x/h = 1.25$, $x/h = 2.5$ and $x/h = 5.0$ for Re=25 on the left and Re=80 on the right. The velocities are normalized this time by 1.5 times the inlet velocity of the small channel to match the experimental data. The overall agreement is better for the SRT than the MRT as discussed before. At section $x/h = 1.25$ and $x/h = 2.5$, the SRT's MSE for Re=25 is 3 times lower than the MRT while the error is similar for $x/h = 5.0$. The Re=80 case shows similar behaviour with 1.9, 8.24 and 3.15 better MSE for the SRT then the MRT which



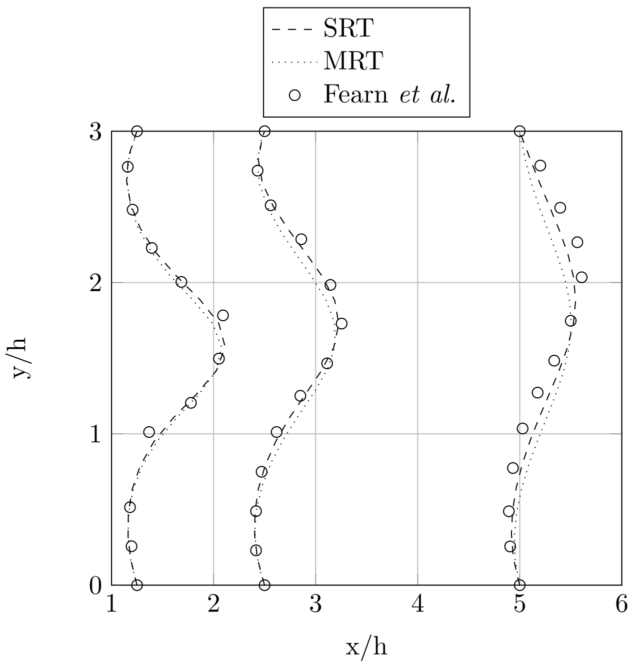


Figure 23 – Velocity profiles for the sudden expansion step at Re=25 (left) and Re=80 (right). X and Y direction normalized by the step height h. Reference data taken from Fearn *et al.* [195].

only underlines the finding of the previous discussed results.

The thorough discussion given here is applicable to many types of simulations as vorticities and pressure gradients are encountered in many practical engineering flows. The next section will elaborate on pressure gradients in conjunction with skewed mesh elements.

5.5 Flow around a cylinder

The previous section has been used to highlight some of the problems with the pressure gradient. In this section, the flow around a cylinder is investigated which is compared to an analytical solution for the pressure coefficient. This offers insight to the actual problem of the pressure treatment. The simulation shown is only for the MRT collision approximation this time as the SRT was not able to produce physical correct results.

Figure 24 shows the streamlines around the cylinder at Re=56. The theoretical onset of transition to the von Kármán vortex street is at a Reynolds number of about 50. The streamlines and contour plot however show still a perfectly symmetrical flow field with equal recirculation areas in the wake.

Figure 25 shows the normalized density distribution around the cylinder. The mesh has been plotted on top to highlight the skewed elements which have been used. All previous simulations have been done on perfectly squared elements without curved geometries. The curvature of the cylinder does not seem to affect the calculation of the velocity and the theoretical separation point of 135° and 225° has been closely matched by the simulation. The density distribution however is affected by the curvature and the points where links connects with the walls show irregularities in

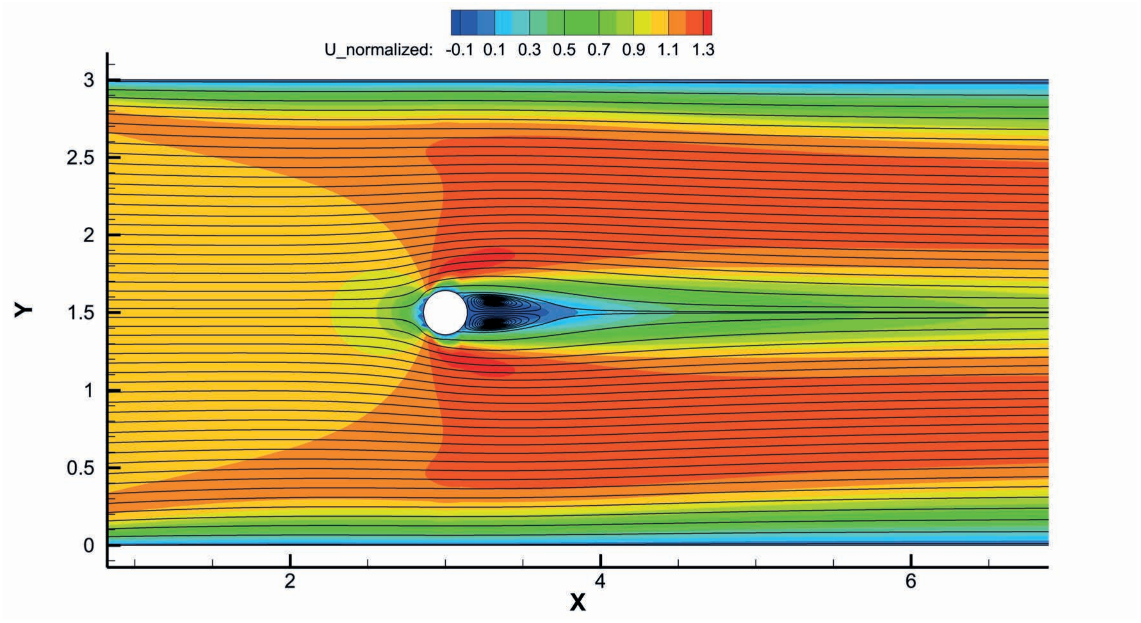


Figure 24 – Streamlines for the cylinder flow at Re=56 using the MRT.

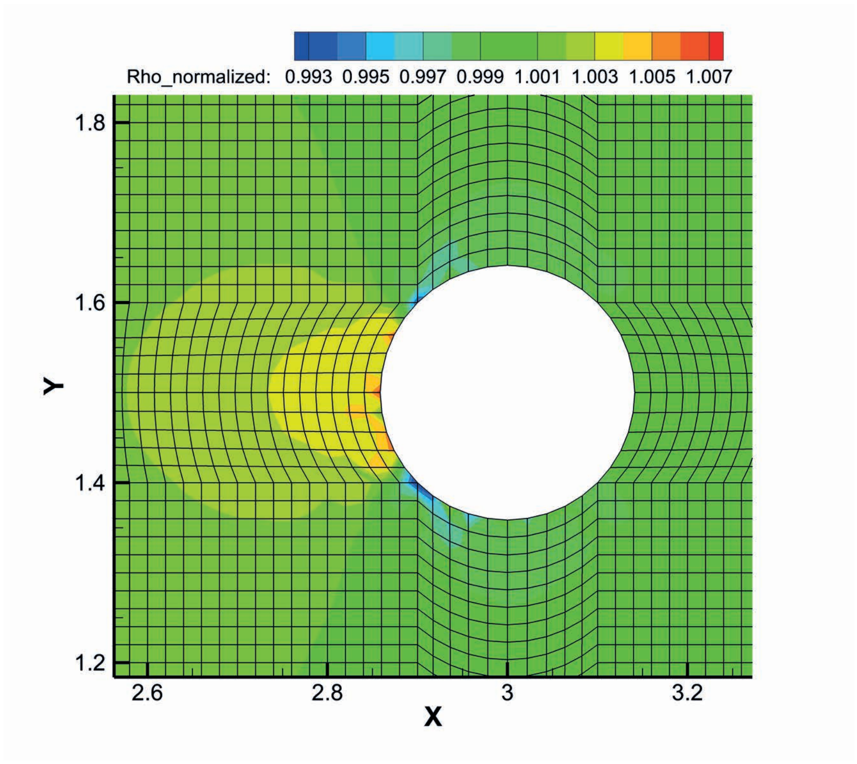


Figure 25 – Normalized density distribution at the cylinder for Re=56. The mesh is shown to highlight the skewed elements.

the density values, especially in the front of the cylinder. Although they might be all lower than 0.7%, over a period of thousands of iterations they will affect the surrounding flow. As stated in the beginning, the SRT failed to produce a physical correct flow field. Although the velocity distribution was just as good as that obtained with the MRT, the values for density were rising constantly (just as discussed for the channel flow), leaving a density of an order of magnitude of 10^{13} for the converged solution. Just as for the channel flow, the MRT is able to produce converged results while the SRT fails to do so. It is not surprising as the cylinder is placed inside a channel.

Figure 26 shows the pressure coefficient distribution around the cylinder and an analytical solution in form of:

$$C_p(\theta_c) = 1 - 4\sin^2(\theta_c) \tag{5.3}$$

exists to calculate the pressure coefficient. The cylinder's front is shown at 0 and 2π while the back of the cylinder is in the middle at π. The mesh induced fluctuations at the front can be clearly seen. The suction at $1/2\pi$ and $3/2\pi$ is not captured while the back of the cylinder experience a maximum in the pressure coefficient which is not captured either by the simulation. This is however plausible since the analytical solution is only valid for attached flows and hence the peak

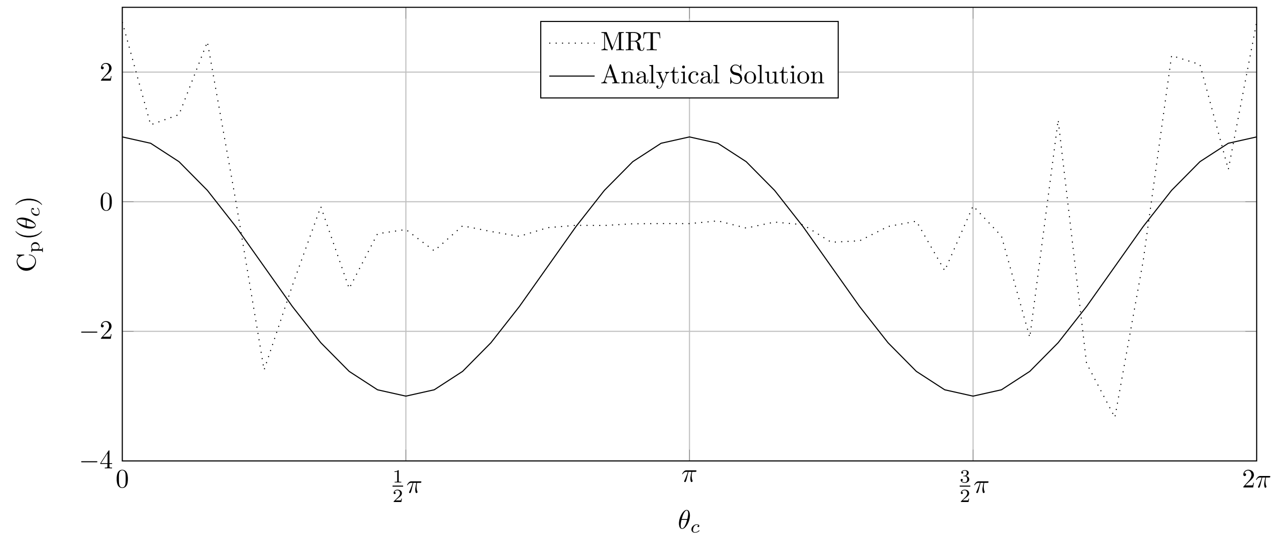


Figure 26 – Pressure coefficient around a cylinder at Re=56 compared to an analytical solution. The cylinder's front is at 0 and 2π respectively while the back is at π.

in pressure coefficient only indicates a second, theoretical stagnation point. However, the pressure coefficient exceeds a value of one at the front which violates the law of physics as one is per definition the maximum pressure coefficient and higher values could only be obtained with negative pressure.

This leaves the question why the velocities are calculated correctly but the pressure fails to give physical correct results as both properties are derived from the density. Looking back at eq.(2.23) and eq.(2.24), the velocity only needs the information of the microscopic velocity component c_i. While the fluxes are calculated using the density, the velocity is calculated by dividing by the density thus eliminating the uncertainty it introduces. Since the velocities however seem to be obtained correctly throughout the other validation cases, including the cylinder flow, leaves the conclusion that the density and therefore pressure must be obtained correctly and hence the reason for the deviation seen in Figure 5.3 must be due to the skewed mesh. This observation is consistent with the fact that the weights w_i in the equilibrium distribution function, eq.(1.26), have not been changed. As soon as the square elements are degenerated, which is the case for the current simulation and shown in Figure 25, the weights are incorrect as they are determined based on the distance to the neighbour node. One could think that adapting the weights for each cell would solve the problem but unfortunately the constraints for deriving the weights are under-constrained [182]. Further physical constraints need to be considered for constructing speed models and their weights, for example using the entropy [196, 197].

It should be noted that simulations at Re=280 have been done as well and that the von Kármán vortex street have been obtained to some extent. The current limitations of inlet velocity, viscosity and required domain size to reduce boundary condition induced instabilities yields prohibitively long simulation times and therefore no results for the unsteady cylinder flow.

6 Conclusion

The last section concludes on the findings made in the previous sections. It is split into two sections dealing separately with the LBM and with the developed code.

6.1 Verdict on the lattice Boltzmann method

Some of the advantages of the LBM have been mentioned throughout this work. The development of a fully working two dimensional fluid solver gave the possibility to explore some of the features of the LBM but also some of the drawbacks which are generally not mentioned in the literature. None the less, the LBM has proven itself to be a valid alternative to the NSE based approaches and some, if not all, of the drawbacks can be handled by using more complex models and methods. The full potential of this method could not be explored due to the time constraint given to develop the solver which restricted the implementation of more sophisticated methods. In any case, a short review is given at this point to summarized the knowledge and experience gained while working with this method.

One of the main advantages usually referred to when talking about the LBM is that its simple, advection like, linear partial differential equation requires less computational time. There is only one equation in contrast to the 3-6 equation based NSE depending on the dimensions and whether the energy equation is considered or not. It is a local approach which does not require the approximation of derivatives and hence it is easy to parallelize. This ensures, even for a sequential code, that computational speed should be very fast which can be attested. However, due to the restriction imposed by the streaming, the time step is always unity and therefore the CFL number as well. Computational speed might be faster but has a convergence rate which is rather slow. A fully implicit NSE solver with a CFL$\gg$1 is more likely to obtain faster results than the LBM. There are possibilities to reduce the time penalty which has been discussed, either by considering a steady state flow or an off-LBM which requires a different treatment of the time step. Furthermore, the convergence rate decreases with an increase of the mesh size. For example, the lid driven cavity on 100 x 100 nodes needs around 35 000 iterations to converge below 10^{-4} for all residuals (density, u_x velocity, u_y velocity) while the iterations needed on a 320 x 320 mesh are 325 000. The increase in mesh size by a factor of 3.2 caused the simulation to need ten times the iterations. The convergence rate is decreased while the time per iteration is increased. This means that the scaling between computational time and mesh size is not linear but exponential.

The locality of the LBM only holds for the classical approach. As soon as unstructured meshes are used, the approach becomes an off-LBM which requires the approximation of derivatives. Communication overheads are introduced in the parallel code and the advantage of being are local lost in favour of handling more complex geometries.

It is possible though to handle complex geometries without needing to switch to unstructured meshes. In that case one needs to switch to a Cartesian mesh with an IBM approach to track the interface between solid and fluid. This makes it very easy to mesh a complex geometry but the user still needs to specify where local mesh refinement is necessary in order to simulate a wall-bounded, turbulent flow for example. However, the resolution of the boundary for such an approach is questionable. Local mesh refinements have the same problem as the off-LBM where

cells of different size have different time steps. Furthermore, curved boundaries need to be approximated and are usually represented by stair steps and special treatment at the boundaries is necessary. The off-LBM on the other hand has elements of arbitrary shape and size and therefore the bounce back direction needs to be determined for each node. Even for a structured mesh for a body fitted mesh, the weights w_i in the equilibrium distribution function for the skewed elements need to be adapted in order to account for the irregularities. There is no right or wrong but an art of modelling in between.
There is however one great advantage that does not have a counterpart and it is the reason why it has attracted so many researcher in the past decades. Its most prominent advantage is the weakest spot of the NSE, namely the simplified equations for complex physical models.

6.2 Verdict on the developed code

Most of the code's advantages and disadvantages have been already addressed in section 4.2 and are stated here only for completeness.
The two dimensional fluid solver developed is able to read unstructured CGNS meshes and to perform calculations based on the LBM. The meshes can be created in ICEM CFD and high priority was given to create a user friendly interface to communicate user information with the solver. It is a compact code in its version 1.0 and therefore easy to understand with the documentation provided. The code's structure and algorithms have been kept short and tidy and designed to make further implementations straight forward. Although it cannot be guaranteed that no debugging is required, a great effort has been spent in designing the algorithms to minimize that time. As stated in section 4.2.2, the code has only one speed model (D2Q9), simple boundary conditions (Zou/He, Bounce back) and three collision approximations (SRT, TRT, MRT). Furthermore, as explored in the previous section, the curved boundaries are causing problems in handling the pressure and density although the velocity is still obtained correctly.

This should not compromise the fact that the developed solver produces physically correct results in the domains it has been designed for while a strong commitment towards user friendliness was done. This thesis has been partially written to give a basic introduction into the LBM and may replace introductory literature but most importantly to document the code's structure and its capability to solve actual fluid flows.
The author hopes to have not only provided a starting point for further research on the LBM but also to have communicated some problematic issues with the LBM which are generally not discussed in the literature.

References

[1] J. Hardy, Y. Pomeau, and O. D. Pazzis, “Time evolution of a two-dimensional model system. i. invariant states and time correlation functions,” *Journal of Mathematical Physics*, vol. 14, no. 12, pp. 1746–1759, 1973.

[2] J. Hardy, O. D. Pazzis, and Y. Pomeau, “Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions,” *Physical Review A*, vol. 13, no. 5, pp. 1949–1961, 1976.

[3] U. Frisch, B. Hasslacher, and Y. Pomeau, “Lattice gas automata for the navier-stokes equation,” *Physical Review Letters*, vol. 56, no. 14, pp. 1505–1508, 1986.

[4] U. Frisch, D. d’Humières, B. Hasslacher, P. Lalleman, and Y. Pomeau, “Lattice gas hydrodynamics in two and three dimensions,” *Complex systems*, vol. 1, no. 4, pp. 649–707, 1987.

[5] D. A. Wolf-Gladrow, *Lattice gas cellular automata and Lattice Boltzmann - An introduction.* Berlin: Springer, 1 ed., 2005.

[6] K. N. Premnath, J. C. Nave, and S. Banerjee, “Computation of multiphase flows with lattice boltzmann methods,” in *American Society of Mechanical Engineers, Fluids Engineering Division (Publication) FED*, vol. 261 FED, pp. 403–420, 2005.

[7] S. Chapman, “On the law of distribution of molecular velocities, and on the theory of viscosity and thermal conduction, in a non-uniform simple monatomic gas,” *Philosophical Transactions of the Royal Society A*, vol. 216, pp. 279–348, 1916.

[8] D. Enskog, *Kinetische Theorie der Vorgänge in Mässig Verdünnten Gasen.* PhD thesis, University of Upsala, 1917.

[9] E. M. Viggen, *The Lattice Boltzmann Method with Applications in Acoustics.* PhD thesis, University of Trondheim, 2009.

[10] J. Lätt, *Hydrodynamic Limit of Lattice Boltzmann Equations.* PhD thesis, University of Geneva, 2007.

[11] B. Chopard and M. Droz, *Cellular Automata Modelling of Physical Systems.* Cambridge: Cambridge University Press, 1 ed., 1998.

[12] B. Chopard, P. O. Luthi, and A. Masselot, “Cellular automata and lattice boltzmann techniques: An approach to model and simulate complex systems,” *Advances in Complex Systems*, vol. 5, no. 2, pp. 103–246, 2012.

[13] P. L. Bhatnagar, E. P. Gross, and M. Krook, “A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems,” *Physical Review*, vol. 94, no. 3, p. 511–525, 1954.

[14] P. Welander, “On the temperature jump in a rarefied gas,” *Arkiv för Fysik*, vol. 7, no. 6, pp. 507–533, 1954.

[15] I. Ginzburg, “Variably saturated flow described with the anisotropic lattice boltzmann methods,” *Computers and Fluids*, vol. 35, no. 8-9, pp. 831–848, 2006.

[16] R. Du and W. Liu, “A new multiple relaxation time lattice boltzmann method for natural convection,” *Journal of Scientific Computing*, Article in Press.

[17] R. Du and B. C. Shi, “Incompressible multi-relaxation-time lattice boltzmann model in 3d space,” *Journal of Hydrodynamics*, vol. 22, no. 6, pp. 782–787, 2010.

[18] A. A. Mohamad, *Lattice Boltzmann method*. London: Springer, 1 ed., 2011.

[19] L. S. Lin, Y. C. Chen, and C. A. Lin, “Multi relaxation time lattice boltzmann simulations of deep lid driven cavity flows at different aspect ratios,” *Computers and Fluids*, vol. 45, no. 1, pp. 233–240, 2011.

[20] L. Wang, Z. Guo, and C. Zheng, “Multi relaxation time lattice boltzmann model for axisymmetric flows,” *Computers and Fluids*, vol. 39, no. 9, pp. 1542–1548, 2010. Cited By (since 1996):4.

[21] H. Q. Xie, Z. Zeng, L. Q. Zhang, G. Y. Liang, H. Mizuseki, and Y. Kawazoe, “Multi-relaxation-time lattice boltzmann front tracking method for two-phase flow with surface tension,” *Chinese Physics B*, vol. 21, no. 12, pp. 1–7, 2012.

[22] K. Fallah, A. Fardad, N. Sedaghatizadeh, E. Fattahi, and A. Ghaderi, “Numerical simulation of flow around two rotating circular cylinders in staggered arrangement by multi-relaxation-time lattice boltzmann method at low reynolds number,” *World Applied Sciences Journal*, vol. 15, no. 4, pp. 544–554, 2011.

[23] Z. H. Chai, B. C. Shi, and L. Zheng, “Simulating high reynolds number flow in two-dimensional lid-driven cavity by multi-relaxation-time lattice boltzmann method,” *Chinese Physics*, vol. 15, no. 8, pp. 1855–1863, 2006.

[24] Q. L. Ding, L. L. Wang, D. G. Wang, and Z. Fang, “Three-dimensional numerical simulation of bend flow based on the multi-relaxation times lattice boltzmann method,” *Advances in Water Science*, vol. 23, no. 4, pp. 523–528, 2012.

[25] J. Meng and Y. Zhang, “Accuracy analysis of high-order lattice boltzmann models for rarefied gas flows,” *Journal of Computational Physics*, vol. 230, no. 3, pp. 835–849, 2011.

[26] D. Lycett-Brown, I. Karlin, and K. H. Luo, “Droplet collision simulation by a multi-speed lattice boltzmann method,” *Communications in Computational Physics*, vol. 9, no. 5, pp. 1219–1234, 2011.

[27] X. Shan, X. F. Yuan, and H. Chen, “Kinetic theory representation of hydrodynamics: a way beyond the navier–stokes equation,” *Journal of Fluid Mechanics*, vol. 550, pp. 413–441, 2006.

[28] Z. H. Chai and T. S. Zhao, “A pseudopotential-based multiple-relaxation-time lattice boltzmann model for multicomponent / multiphase flows,” *Acta Mechanica Sinica*, vol. 28, no. 4, pp. 983–992, 2012.

[29] Q. Zou and X. He, “On pressure and velocity boundary conditions for the lattice boltzmann bgk model,” *Physics of Fluids*, vol. 9, no. 6, pp. 1591–1598, 1997.

[30] S. Wolfram, “Cellular automaton fluids 1: Basic theory,” *Journal of Statistical Physics*, vol. 45, no. 3-4, pp. 471–526, 1986.

[31] P. Lavallée, J. P. Boon, and A. Noullez, “Boundaries in lattice gas flows,” *Physica D: Nonlinear Phenomena*, vol. 47, no. 1-2, pp. 233–240, 1991.

[32] T. Inamuro, M. Yoshino, and F. Ogino, “A non-slip boundary condition for lattice boltzmann simulations,” *Physics of Fluids*, vol. 7, no. 12, pp. 2928–2930, 1995.

[33] S. Maier, R. S. Bernard, and D. W. Grunau, “Boundary conditions for the lattice boltzmann method,” *Physics of Fluids*, vol. 8, no. 7, pp. 1788–1801, 1996.

[34] D. P. Ziegler, “Boundary conditions for lattice boltzmann simulations,” *Journal of statistical Physics*, vol. 71, no. 5-6, pp. 1171–1177, 1993.

[35] S. Chen, D. Martínez, and R. Mei, "On boundary conditions in lattice boltzmann methods," *Physics of Fluids*, vol. 8, no. 9, pp. 2527–2536, 1996.

[36] M. Bouzidi, M. Firdaouss, and P. Lallemand, "Momentum transfer of a boltzmann-lattice fluid with boundaries," *Physics of Fluids*, vol. 13, no. 11, pp. 3452–3459, 2001.

[37] X. He, Q. Zou, L. S. Luo, and M. Dembo, "Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice boltzmann bgk model," *Journal of Statistical Physics*, vol. 87, no. 1-2, pp. 115–136, 1997.

[38] D. R. Noble, S. Chen, J. G. Georgiadis, and R. O. Buckius, "A consistent hydrodynamic boundary condition for the lattice boltzmann method," *Physics of Fluids*, vol. 7, no. 1, pp. 203–209, 1995.

[39] I. Ginzburg and D. D'Humières, "Multireflection boundary conditions for lattice boltzmann models," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 68, no. 6 2, pp. 666141–6661430, 2003. Cited By (since 1996):167.

[40] Z. Guo, C. Zheng, and B. Shi, "An extrapolation method for boundary conditions in lattice boltzmann method," *Physics of Fluids*, vol. 14, no. 6, pp. 2007–2010, 2002.

[41] P. Dickenson, "The feasibility of smoothed particle hydrodynamics for multiphase oilfield systems," in *Seventh International Conference on CFD in the Minerals and Process Industries*, (Melbourne, Australia), pp. 1–6, 2009.

[42] A. K. Gunstensen, D. H. Rothman, S. Zaleski, and G. Zanetti, "Lattice boltzmann model of immiscible fluids," *Physical Review A*, vol. 43, pp. 4320–4327, 1991.

[43] X. Shan and H. Chen, "Lattice boltzmann model for simulating flows with multiple phases and components," *Physical Review E*, vol. 47, pp. 1815–1819, 1993.

[44] X. Shan and G. Doolen, "Multicomponent lattice boltzmann model with interparticle interaction," *Physical Review J*, vol. 81, pp. 379–393, 1995.

[45] Y. H. Qian, S. Succi, and S. A. Orszag, "Recent advances in lattice boltzmann computing," *Annual Reviews of Computational Physics*, vol. 3, pp. 195–242, 1995.

[46] M. R. Swift, W. R. Osborn, and J. M. Yeomans, "Lattice boltzmann simulation of nonideal fluids," *Physical Review Letters*, vol. 75, pp. 830–833, 1995.

[47] M. R. Swift, S. E. Orlandini, W. R. Osborn, and J. M. Yeomans, "Lattice boltzmann simulations of liquid-gas and binary fluid systems," *Physical Review E*, vol. 54, pp. 5041–5052, 1996.

[48] S. Chen and G. Doolen, "Lattice boltzmann method for fluid flows," *Annual Review of Fluid Mechanics*, vol. 30, pp. 329–364, 1998.

[49] H. Farhat and J. S. Lee, "Fundamentals of migrating multi-block lattice boltzmann model for immiscible mixtures in 2d geometries," *International Journal of Multiphase Flow*, vol. 36, no. 10, pp. 769–779, 2010.

[50] A. Kawasaki, J. Onishi, Y. Chen, and H. Ohashi, "A lattice boltzmann model for contact line motions," *Computers and Mathematics with Applications*, vol. 55, no. 7, pp. 1492–1502, 2008.

[51] X. Xing, D. L. Butler, and C. Yang, "A lattice boltzmann based single-phase method for modeling surface tension and wetting," *Computational Materials Science*, vol. 39, no. 2, pp. 282–290, 2007.

[52] J. Tölke, S. Freudiger, and M. Krafczyk, “An adaptive scheme using hierarchical grids for lattice boltzmann multiphase flow simulations,” *Computers and Fluids*, vol. 35, no. 8-9, pp. 820–830, 2006.

[53] J. Chin, E. S. Boek, and P. V. Coveney, “Lattice boltzmann simulation of the flow of binary immiscible fluids with different viscosities using the shan-chen microscopic interaction model,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 547–558, 2002.

[54] H. Huang, Z. Li, S. Liu, and X. Y. Lu, “Shan and chen-type multiphase lattice boltzmann study of viscous coupling effects for two-phase flow in porous media,” *International Journal for Numerical Methods in Fluids*, vol. 61, no. 3, pp. 341–354, 2009.

[55] H. Huang, D. T. Thorne, M. G. Schaap, and M. C. Sukop, “Proposed approximation for contact angles in shan and chen-type multicomponent multiphase lattice boltzmann models,” *Physical Review E*, vol. 76, no. 6, p. 066701, 2007.

[56] H. P. Fang, R. Z. Wan, and L. W. Fan, “Lattice boltzmann method simulation on the flow of two immiscible fluids in complex geometry,” *Chinese Physics*, vol. 9, no. 7, pp. 515–518, 2000.

[57] H. Huang, H. Zheng, X. Y. Lu, and C. Shu, “An evaluation of a 3d free energy based lattice boltzmann model for multiphase flows with large density ratio,” *International Journal for Numerical Methods in Fluids*, vol. 63, no. 10, pp. 1193–1207, 2010.

[58] J. Zhang and D. Y. Kwok, “On the validity of the cassie equation via a mean field free energy lattice boltzmann approach,” *Journal of colloid and interface science*, vol. 282, no. 2, pp. 434–438, 2005.

[59] Q. Li and A. J. Wagner, “Symmetric free energy based multicomponent lattice boltzmann method,” *Physical Review E*, vol. 76, no. 3, p. 036701, 2007.

[60] T. Inamuro, N. Konishi, and F. Ogino, “Galilean invariant model of the lattice boltzmann method for multiphase fluid flows using free energy approach,” *Computer Physics Communications*, vol. 129, no. 1, pp. 32–45, 2000.

[61] H. Huang, M. Krafczyk, and X. Lu, “Forcing term in single-phase and shan-chen-type multiphase lattice boltzmann models,” *Physical Review E*, vol. 84, no. 4, p. 046710, 2011.

[62] C. Rosales and D. S. Whyte, “Dual grid lattice boltzmann method for multiphase flows,” *International Journal for Numerical Methods in Engineering*, vol. 84, no. 9, pp. 1068–1084, 2010.

[63] L. Martinelli, A. Jameson, and F. Grasso, “Multigrid method for the navier-stokes equations,” in *24th AIAA Aerospace Sciences meeting*, (Reno, Nevada), pp. 1–10, 1986.

[64] R. J. Leveque, “A study of numerical methods for hyperbolic conservation laws with stiff source terms,” *Journal of Computational Physics*, vol. 86, no. 1, pp. 187–210, 1990.

[65] T. Reis and P. J. Dellar, “A volume-preserving sharpening approach for the propagation of sharp phase boundaries in multiphase lattice boltzmann simulations,” *Computers and Fluids*, vol. 46, no. 1, pp. 417–421, 2011.

[66] W. Bao and S. Jin, “The random projection method for hyperbolic conservation laws with stiff reaction terms,” *Journal of Computational Physics*, vol. 163, no. 1, pp. 216–248, 2000.

[67] X. He, S. Chen, and R. Zhang, “A lattice boltzmann scheme for incompressible multiphase flow and its application in simulation of rayleigh-taylor instability,” *Journal of Computational Physics*, vol. 152, no. 2, pp. 642–663, 1999.

[68] R. Zhang, X. He, and S. Chen, "Interface and surface tension in incompressible lattice boltzmann multiphase model," *Computer Physics Communications*, vol. 129, no. 1, pp. 121–130, 2000.

[69] J. Chao, R. Mei, R. Singh, and W. Shyy, "A filter-based, mass-conserving lattice boltzmann method for immiscible multiphase flows," *International Journal for Numerical Methods in Fluids*, vol. 66, no. 5, pp. 622–647, 2011.

[70] J. Kim, "A continuous surface tension force formulation for diffuse-interface models," *Journal of Computational Physics*, vol. 204, no. 2, pp. 784–804, 2005.

[71] L. M. Schwartz, N. Martys, D. P. Bentz, E. J. Garboczi, and S. Torquato, "Cross-property relations and permeability estimation in model porous media," *Physical Review E*, vol. 48, no. 6, pp. 4584–4591, 1993.

[72] H. Darcy, "Les fontaines publiques de la ville de dijon," tech. rep., Bibliothèque nationale de France, département Littérature et art, V-13914, 1856.

[73] D. H. Rothman, "Cellular-automaton fluids; a model for flow in porous media," *Geophysics April*, vol. 53, no. 4, pp. 509–518, 1988.

[74] S. Succi, E. Foti, and F. Higuera, "Three-dimensional flows in complex geometries with the lattice boltzmann method," *Europhysics Letters*, vol. 10, no. 5, pp. 438–438, 1989.

[75] A. Cancelliere, C. Chang, E. Foti, D. H. Rothman, and S. Succi, "The permeability of a random medium: Comparison of simulation with theory," *Physics of Fluids A*, vol. 2, no. 12, pp. 2085–2088, 1990.

[76] H. C. Brinkman, "A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles," *Applied Scientific Research*, vol. 1, no. 1, pp. 27–34, 1947.

[77] J. Kozeny, "Uüber kapillare leitung des wassers im boden (on capillary water transport in soil," tech. rep., Sitzungsberichte der Akademie der Wissenschaften in Wien, 1927.

[78] P. C. Carman, "Fluid flow through granular beds," in *Transactions of the Institution of Chemical Engineers*, vol. 15, (London, UK), pp. 150–166, 1937.

[79] P. C. Carman, *Flow of gases through porous media.* London: Butterworths, 1 ed., 1957.

[80] A. W. J. Heijs and C. P. Lowe, "Numerical evaluation of the permeability and the kozeny constant for two types of porous media," *Physical Review E*, vol. 51, no. 5, pp. 4346–4352, 1995.

[81] P. M. Adler and J. F. Thovert, "Real porous media: Local geometry and macroscopic properties," *Applied Mechanics Reviews*, vol. 51, no. 9, pp. 537–585, 1998.

[82] C. Pan, L. S. Luo, and C. T. Miller, "An evaluation of lattice boltzmann schemes for porous medium flow simulation," *Computers and Fluids*, vol. 35, no. 8-9, pp. 898–909, 2006.

[83] B. Ferréol and D. H. Rothman, "Lattice-boltzmann simulations of flow through fontainebleau sandstone," *Transport in Porous Media*, vol. 20, no. 1-2, pp. 3–20, 1995.

[84] Z. Guo and T. S. Zhao, "Lattice boltzmann model for incompressible flows through porous media," *Physical Review E*, vol. 66, no. 3, pp. 036304/1–036304/9, 2002.

[85] A. Koponen, D. Kandhai, E. Hellén, M. Alava, A. Hoekstra, M. Kataja, K. Niskanen, P. Sloot, and J. Timonen, "Permeability of three-dimensional random fiber webs," *Physical Review Letters*, vol. 80, no. 4, pp. 716–719, 1998.

[86] C. Pan, M. Hilpert, and C. T. Miller, "Lattice-boltzmann simulation of two-phase flow in porous media," *Water Resources Research*, vol. 40, no. 1, pp. W015011–W0150114, 2004.

[87] O. Reynolds, "On the dynamical theory of incompressible viscous fluids and the determination of the criterion," *Philosophical Transactions of the Royal Society*, vol. 186, pp. 123–164, 1895.

[88] C. E. Lynch and M. J. Smithy, "Hybrid rans-les turbulence models on unstructured grids," in *38th AIAA Fluid Dynamics Conference*, (Seattle, Washington), pp. 1–14, 2008.

[89] S. B. Pope, *Turbulent flows.* Cambridge: Cambridge University Press, 1 ed., 2000.

[90] S. Pirker, C. Goniva, C. Kloss, S. Puttinger, J. Houben, and S. Schneiderbauer, "Application of a hybrid lattice boltzmann-finite volume turbulence model to cyclone short-cut flow," *Powder Technology*, vol. 235, pp. 572–580, 2013.

[91] J. W. Kim, B. Y. Min, L. Sankar, N. Yeshala, and T. A. Egolf, "Fuselage drag reduction studies using a coupled lattice boltzmann and navier-stokes methodology," in *37th European Rotorcraft Forum 2011, ERF 2011*, (Vergiate and Gallarate, Italy), pp. 325–335, 2011.

[92] J. W. Kim, L. Sankar, B. Y. Min, N. Yeshala, and T. A. Egolf, "Multiscale modeling of active flow control for fuselage drag reduction," in *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, (Nashville, United States), pp. 1–11, 2012.

[93] S. K. Choi, S. O. Kim, T. H. Lee, Y. I. Kim, and D. Hahn, "Computation of turbulent natural convection in a rectangular cavity with the lattice boltzmann method," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 61, no. 6, pp. 492–504, 2012.

[94] P. J. Mason and A. R. Brown, "On subgrid models and filter operations in large eddy simulations," *Journal of the Atmospheric Sciences*, vol. 56, no. 13, pp. 2101–2114, 1999.

[95] J. Berland, P. Lafon, F. Daude, F. Crouzet, C. Bogey, and C. Bailly, "Filter shape dependence and effective scale separation in large-eddy simulations based on relaxation filtering," *Computers and Fluids*, vol. 47, no. 1, pp. 65–74, 2011.

[96] P. Sagaut and R. Grohens, "Discrete filters for large eddy simulation," *International Journal for Numerical Methods in Fluids*, vol. 31, no. 8, pp. 1195–1220, 1999.

[97] T. S. Lund, "The use of explicit filters in large eddy simulation," *Computers and Mathematics with Applications*, vol. 46, no. 4, pp. 603–616, 2003.

[98] P. Wu and J. Meyers, "Globally conservative high-order filters for large-eddy simulation and computational aero-acoustics," *Computers and Fluids*, vol. 48, no. 1, pp. 150–162, 2011.

[99] A. E. Tejada-Martinez and K. E. Jansen, "Spatial test filters for dynamic model large-eddy simulation with finite elements," *Communications in Numerical Methods in Engineering*, vol. 19, no. 3, pp. 205–213, 2003.

[100] J. Smagorinsky, "General circulation experiments with the primitive equations," *Monthly Weather Review*, vol. 91, no. 3, pp. 99–164, 1963.

[101] D. K. Lilly, "On the application of the eddy viscosity concept in the inertial sub range of turbulence," Tech. Rep. 123, National Center for Atmospheric Research, 1966.

[102] H. Feiz, H. J. Soo, and S. Menon, "Les of turbulent jets using the lattice boltzmann approach," in *41th AIAA Aerospace Sciences Meeting*, (Reno, Nevada), pp. 1–15, 2003.

[103] M. Stiebler, M. Krafczyk, S. Freudiger, and M. Geier, "Lattice boltzmann large eddy simulation of subcritical flows around a sphere on non-uniform grids," *Computers and Mathematics with Applications*, vol. 61, no. 12, pp. 3475–3484, 2011.

[104] H. Wu, J. Wang, and Z. Tao, "Passive heat transfer in a turbulent channel flow simulation using large eddy simulation based on the lattice boltzmann method framework," *International Journal of Heat and Fluid Flow*, vol. 32, no. 6, pp. 1111–1119, 2011.

[105] H. Wu and J. Wang, "Large eddy simulation of square duct turbulence based on the lattice boltzmann method," *Hangkong Dongli Xuebao/Journal of Aerospace Power*, vol. 27, no. 1, pp. 1–8, 2012.

[106] P. Sagaut, "Toward advanced subgrid models for lattice-boltzmann-based large-eddy simulation: Theoretical formulations," *Computers and Mathematics with Applications*, vol. 59, no. 7, pp. 2194–2199, 2010.

[107] H. Liu, M. Li, and A. Shu, "Large eddy simulation of turbulent shallow water flows using multi-relaxation-time lattice boltzmann model," *International Journal for Numerical Methods in Fluids*, vol. 70, no. 12, pp. 1573–1589, 2012.

[108] A. N. Kolmogorov, "The local structure of turbulence in incompressible viscous fluids at very large reynolds numbers," Tech. Rep. 30, Doklady Akademii Nauk, 1941.

[109] A. N. Kolmogorov, "On the degeneration of isotropic turbulence in an incompressible viscous fluids," Tech. Rep. 31, Doklady Akademii Nauk, 1941.

[110] A. N. Kolmogorov, "Dissipation of energy in isotropic turbulence," Tech. Rep. 32, Doklady Akademii Nauk, 1941.

[111] J. Jimenez, "The contributions of a.n. kolmogorov to the theory of turbulence," *Arbor*, vol. 178, no. 704, pp. 589–606, 2004.

[112] H. Yu, *Lattice Boltzmann equation simulations of turbulence, mixing, and combustion.* PhD thesis, Texas A&M University, 2004.

[113] H. Yu, S. S. Girimaji, and L. S. Luo, "Dns and les of decaying isotropic turbulence with and without frame rotation using lattice boltzmann method," *Journal of Computational Physics*, vol. 209, no. 2, pp. 599–616, 2005.

[114] C. M. Tekeira, "Incorporating turbulence models into the lattice-boltzmann method," *International Journal of Modern Physics C*, vol. 9, no. 8, pp. 1159–1175, 1998.

[115] M. Böhle and R. Becker, "Verification of a lattice-boltzmann method by using analytical flow solutions of standard flow problems," in *Proceedings of the ASME Fluids Engineering Division Summer Conference 2009*, vol. 1, (Vail, United States), pp. 2041–2047, 2009.

[116] A. Fakhari and T. Lee, "Multiple-relaxation-time lattice boltzmann method for immiscible fluids at high reynolds numbers," *Physical Review E*, vol. 87, no. 2, p. 023304, 2013.

[117] D. Chiappini, G. Bella, S. Succi, F. Toschi, and S. Ubertini, "Improved lattice boltzmann without parasitic currents for rayleigh-taylor instability," *Communications in Computational Physics*, vol. 7, no. 3, pp. 423–444, 2010.

[118] T. Lee and P. F. Fischer, "Eliminating parasitic currents in the lattice boltzmann equation method for nonideal gases," *Physical Review E*, vol. 74, no. 4, p. 046709, 2006.

[119] T. Lee, "Effects of incompressibility on the elimination of parasitic currents in the lattice boltzmann equation method for binary fluids," *Computers and Mathematics with Applications*, vol. 58, no. 5, pp. 987–994, 2009.

[120] Q. Kun, *Development of Lattice Boltzmann method for compressible flows.* PhD thesis, National University of Singapore, 2008.

[121] G. Yan, Y. Chen, and S. Hu, “Simple lattice boltzmann model for simulating flows with shock wave,” *Physical review E*, vol. 59, no. 1, pp. 454–459, 1999.

[122] W. Shi, W. Shyy, and R. Mei, “Finite-difference-based lattice boltzmann method for inviscid compressible flows,” *Numerical Heat Transfer*, vol. 40, no. 1, pp. 1–21, 2001.

[123] T. Kataoka and M. Tsutahara, “Lattice boltzmann method for the compressible euler equations,” *Physical review E*, vol. 69, no. 5, p. 056702, 2004.

[124] T. Kataoka and M. Tsutahara, “Lattice boltzmann method for the compressible navier-stokes equations with flexible specific-heat ratio,” *Physical review E*, vol. 69, no. 32, p. 035701, 2004.

[125] C. Sun, “Lattice-boltzmann method for high speed flows,” *Physical review E*, vol. 58, no. 6, pp. 7283–7287, 1998.

[126] C. Sun, “Adaptive lattice boltzmann model for compressible flows: viscous and conductive properties,” *Physical review E*, vol. 61, no. 3, pp. 2645–2653, 2000.

[127] C. Sun, “Simulations of compressible flows with strong shocks by an adaptive lattice boltzmann model,” *Journal of Computational Physics*, vol. 161, no. 1, pp. 70–84, 2000.

[128] C. Sun and A. T. Hsu, “Three-dimensional lattice boltzmann model for compressible flows,” *Physical review E*, vol. 68, no. 12, p. 016303, 2003.

[129] C. Sun and A. T. Hsu, “Multi-level lattice boltzmann model on square lattice for compressible flows,” *Computers and Fluids*, vol. 33, no. 10, pp. 1363–1385, 2004.

[130] B. He, Y. Chen, W. Feng, Q. Li, A. Song, Y. Want, M. Zhang, and W. Zhang, “Compressible lattice boltzmann method and applications,” *International Journal of Numerical Analysis and Modeling*, vol. 9, no. 2, pp. 410–418, 2012.

[131] X. Shan, X. F. Yuan, and H. Chen, “Kinetic theory representation of hydrodynamics: A way beyond the navier-stokes equation,” *Journal of Fluid Mechanics*, vol. 550, pp. 413–441, 2006.

[132] C. Zhong, K. Li, J. Sun, C. Zhuo, and J. Xie, “Compressible flow simulation around airfoil based on lattice boltzmann method,” *Transactions of Nanjing University of Aeronautics and Astronautics*, vol. 26, no. 3, pp. 206–211, 2009.

[133] C. Zhuo, C. Zhong, K. Li, J. Xie, and Y. Zhang, “Simulation of high reynolds number flow around airfoil by lattice boltzmann method,” *Hangkong Xuebao/Acta Aeronautica et Astronautica Sinica*, vol. 31, no. 2, pp. 238–243, 2010.

[134] R. A. Shock, S. Mallick, H. Chen, V. Yakhot, and R. Zhang, “Recent results on two-dimensional airfoils using a lattice boltzmann-based algorithm,” *Journal of Aircraft*, vol. 39, no. 3, pp. 434–439, 2002.

[135] G. Thömmes, M. Seaid, and M. K. Banda, “Lattice boltzmann methods for shallow water flow applications,” *International Journal for Numerical Methods in Fluids*, vol. 55, no. 7, pp. 673–692, 2007.

[136] M. Watari and M. Tsutahara, “Supersonic flow simulations by a three-dimensional multi-speed thermal model of the finite difference lattice boltzmann method,” *Physica A: Statistical Mechanics and its Applications*, vol. 364, pp. 129–144, 2006.

[137] R. Kamali and A. H. T. Frad, “Simulation of hypersonic viscous flow past a 2d circular cylinder using lattice boltzmann method,” in *American Society of Mechanical Engineers, Fluids Engineering Division (Publication) FEDSM*, vol. 1, pp. 1907–1915, 2010.

[138] F. J. Alexander, S. Chen, and J. D. Sterling, “Lattice boltzmann thermohydrodynamics,” *Physical Review E*, vol. 47, no. 4, pp. R2249–R2252, 1993.

[139] G. Vahala, P. Pavlo, L. Vahala, and H. Chen, “Effect of velocity shear on a strong temperature gradient - a lattice boltzmann approach,” *Physics Letters A*, vol. 202, no. 5-6, pp. 376–382, 1995.

[140] Y. H. Qian, “Simulating thermohydrodynamics with lattice bgk models,” *Journal of Scientific Computing*, vol. 8, no. 3, pp. 231–242, 1993.

[141] Y. Chen, H. Ohashi, and M. Akiyama, “Thermal lattice bhatnagar-gross-krook model without nonlinear deviations in macrodynamic equations,” *Physical Review E*, vol. 50, no. 4, pp. 2776–2783, 1994.

[142] Y. Chen, H. Ohashi, and M. Akiyama, “Prandtl number of lattice bhatnagar-gross-krook fluid,” *Physics of Fluids*, vol. 7, no. 9, pp. 2280–2282, 1995.

[143] G. R. McNamara, A. L. Garcia, and B. J. Alder, “Stabilization of thermal lattice boltzmann models,” *Journal of Statistical Physics*, vol. 81, no. 1-2, pp. 395–408, 1995.

[144] A. Bartoloni, C. Battista, S. Cabasino, P. S. Paolucci, and Pech, “Lbe simulations of rayleigh-benard convection on the ape100 parallel processor,”

[145] F. Massaioli, R. Benzi, and S. Succi, “Exponential tails in two-dimensional rayleigh-bernard convection,” *Europhysics Letters*, vol. 21, no. 3, pp. 305–310, 1993.

[146] X. Shan, “Simulation of rayleigh-benard convection using a lattice boltzmann method,” *Physical Review E*, vol. 55, no. 3, pp. 2780–2788, 1997.

[147] X. Shan and H. Chen, “Lattice boltzmann model for simulating flows with multiple phases and components,” *Physical Review E*, vol. 47, no. 3, pp. 1815–1819, 1993.

[148] R. H. Khiabani, Y. Joshi, and C. K. Aidun, “Heat transfer in microchannels with suspended solid particles: Lattice-boltzmann based computations,” *Journal of Heat Transfer*, vol. 132, no. 4, pp. 1–9, 2010.

[149] X. Wu and R. Kumar, “Lattice boltzmann model for flow and heat transfer of nanofluids in a microchannel,” in *Proceedings of the 3rd International Conference on Microchannels and Minichannels*, vol. PART A, (Toronto, Canada), pp. 537–543, 2005.

[150] Y. T. Yang and F. H. Lai, “Lattice boltzmann simulation of heat transfer and fluid flow in a microchannel with nanofluids,” *Heat and Mass Transfer/Wärme- und Stoffübertragung*, vol. 47, no. 10, pp. 1229–1240, 2011.

[151] K. Xing and Y. Tao, “Lattice boltzmann simulation of flow and heat transfer characteristics in a symmetrically bifurcated microchannel,” in *ASME International Mechanical Engineering Congress and Exposition*, vol. 375, (Anaheim, United States), pp. 237–240, 2004.

[152] S. Succi and R. Benzi, “Lattice boltzmann equation for quantum mechanics,” *Physica D: Nonlinear Phenomena*, vol. 69, no. 3-4, pp. 327–332, 1993.

[153] S. Palpacelli and S. Succi, “Quantum lattice boltzmann simulation of expanding bose-einstein condensates in random potentials,” *Physical Review E*, vol. 77, no. 6, 2008.

[154] S. Palpacelli, S. Succi, and R. Spigler, "Ground-state computation of bose-einstein condensates by an imaginary-time quantum lattice boltzmann scheme," *Physical Review E*, vol. 76, no. 3, 2007.

[155] S. Succi, "Lattice quantum mechanics: An application to bose-einstein condensation," *International Journal of Modern Physics C*, vol. 9, no. 8, pp. 1577–1585, 1998.

[156] G. S. O'Brien, C. J. Bean, and F. McDermott, "A comparison of published experimental data with a coupled lattice boltzmann-analytic advection-diffusion method for reactive transport in porous media," *Journal of Hydrology*, vol. 268, no. 1-4, pp. 143–157, 2002.

[157] A. Gabrielli, S. Succi, and E. Kaxiras, "A lattice boltzmann study of reactive microflows," *Computer Physics Communications*, vol. 147, no. 1-2, pp. 516–521, 2002.

[158] S. Chen, Z. Liu, C. Zhang, Z. He, Z. Tian, B. Shi, and C. Zheng, "A simple lattice boltzmann scheme for low mach number reactive flows," *Science in China, Series E: Technological Sciences*, vol. 49, no. 6, pp. 714–726, 2006.

[159] S. Chen, Z. Liu, Z. Tian, B. Shi, and C. Zheng, "A simple lattice boltzmann scheme for combustion simulation," *Computers and Mathematics with Applications*, vol. 55, no. 7, pp. 1424–1432, 2008.

[160] V. Novozhilov and C. Byrne, "Lattice boltzmann modeling of thermal explosion in natural convection conditions," *Numerical Heat Transfer; Part A: Applications*, vol. 63, no. 11, pp. 824–839, 2013.

[161] C. Sun and L. L. Munn, "Lattice-boltzmann simulation of blood flow in digitized vessel networks," *Computers and Mathematics with Applications*, vol. 55, no. 7, pp. 1594–1600, 2008.

[162] M. Navidbakhsh and M. Rezazadeh, "An immersed boundary-lattice boltzmann model for simulation of malaria-infected red blood cell in micro-channel," *Scientia Iranica*, vol. 19, no. 5, pp. 1329–1336, 2012.

[163] M. Hilpert, "Lattice-boltzmann model for bacterial chemotaxis," *Journal of mathematical biology*, vol. 51, no. 3, pp. 302–332, 2005.

[164] D. O. Martínez, S. Chen, and W. H. Matthaeus, "Lattice boltzmann magnetohydrodynamics," *Physics of Plasmas*, vol. 1, no. 6, pp. 1850–1867, 1994.

[165] H. Chen and W. H. Matthaeus, "New cellular automaton model for magnetohydrodynamics," *Physical Review Letters*, vol. 58, no. 18, pp. 1845–1848, 1987.

[166] S. Chen, H. Chen, D. Martnez, and W. H. Matthaeus, "Lattice boltzmann model for simulation of magnetohydrodynamics," *Physical Review Letters*, vol. 67, no. 27, pp. 3776–3779, 1991.

[167] W. Schaffenberger and A. Hanslmeier, "Two-dimensional lattice boltzmann model for magnetohydrodynamics," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 66, no. 4, pp. 046702/1–046702/7, 2002.

[168] D. Chatterjee and S. Amiroudine, "Lattice kinetic simulation of nonisothermal magnetohydrodynamics," *Physical Review E*, vol. 81, no. 6, 2010.

[169] G. Fogaccia, R. Benzi, and F. Romanelli, "Lattice boltzmann algorithm for three-dimensional simulations of plasma turbulence," *Physical Review E*, vol. 54, no. 4, pp. 4384–4393, 1996.

[170] A. C. Iannetti, "Cfd general notation system mid-level library," 07 2013. http://www.grc.nasa.gov/WWW/cgns/CGNS_docs_rel25/midlevel/midlevel.pdf.

[171] C. L. Rumsey, D. M. A. Poirier, R. H. Bush, and C. E. Towne, "A user's guide to cgns," 07 2013. http://www.grc.nasa.gov/WWW/cgns/CGNS_docs_rel25/user/usersguide.pdf.

[172] C. steering committee, "Cfd general notation system standard interface data structure," 07 2013. http://www.grc.nasa.gov/WWW/cgns/CGNS_docs_rel25/sids/sids.pdf.

[173] J. F. Thompson, B. K. Soni, and N. P. Weatherill, *Handbook of grid generation.* Boca Raton, Florida: CRC Press, 1 ed., 1998.

[174] K. A. Hoffmann and S. T. Chiang, *Computational fluid dynamics for engineers: 1.* Engineering Education System, 2 ed., 1993.

[175] K. A. Hoffmann and S. T. Chiang, *Computational fluid dynamics for engineers: 2.* Engineering Education System, 1 ed., 1993.

[176] A. Bardow, I. V. Karlin, and A. Gusev, "General characteristic-based algorithm for off-lattice boltzmann simulations," *Europhysics Letters*, vol. 75, no. 3, pp. 434–440, 2006.

[177] T. Lee and C. L. Lin, "A characteristic galerkin method for the discrete boltzmann equation," *Journal of Computational Physics*, vol. 171, no. 1, pp. 336–356, 1999.

[178] I. V. Karlin, S. Succi, and S. Orszag, "Lattice boltzmann method for irregular grids," *Europhysics Letters*, vol. 75, no. 3, pp. 434–440, 1999.

[179] A. Bardow, I. V. Karlin, and A. Gusev, "Multi-speed models in off-lattice boltzmann simulations," *Physics Review E*, vol. 77, no. 2, p. 025701, 2008.

[180] B. Yan and G. Yan, "A steady-state lattice boltzmann model for incompressible flows," *Computers and Mathematics with Applications*, vol. 61, no. 5, pp. 1348–1354, 2011.

[181] M. Bernaschi, S. Succi, and H. Chen, "Accelerated lattice boltzmann schemes for steady-state flow simulations," *Journal of Scientific Computing*, vol. 16, no. 2, pp. 135–144, 2001.

[182] S. Succi, *Lattice Boltzmann equation.* Oxford university press, 1 ed., 2013.

[183] J. Latt, B. Chopard, O. Malaspinas, M. Deville, and A. Michler, "Straight velocity boundaries in lattice boltzmann method," tech. rep., University of Geneva, 2008.

[184] J. H. Ferziger, "Estimation and reduction of numerical error," in *ASME Winter Annual Meeting*, (Washington, DC), 1993.

[185] J. H. Ferziger and M. Perić, "Further discussion of numerical errors in cfd," *International Journal for Numerical Methods in Fluids*, vol. 23, no. 12, pp. 1263–1274, 1996.

[186] P. J. Roache, "Perspective: A method for uniform reporting of grid refinements studies," *Journal of Fluids Engineering*, vol. 116, no. 3, pp. 405–413, 1994.

[187] I. B. Celik, U. Ghia, P. J. Roache, C. J. Freitas, R. Hugh, and E. Peter, "Procedure for estimation and reporting of uncertainty due to discretization in cfd applications," *Journal of Fluids Engineering*, vol. 130, no. 7, p. 078001, 2008.

[188] F. Durst, S. Ray, B. Ünsal, and O. A. Bayoumi, "The development lengths of laminar pipe and channel flows," *Journal of Fluids Engineering*, vol. 127, no. 6, pp. 1154–1160, 2005.

[189] D. A. Perumal and A. K. Dass, "Application of lattice boltzmann method for incompressible viscous flows," *Applied Mathematical Modelling*, vol. 37, no. 6, pp. 4075–4092, 2013.

[190] U. Ghia, K. N. Ghia, and C. T. Shin, "High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method," *Journal of Computational Physics*, vol. 48, no. 3, pp. 387–411, 1982.

[191] A. K. Prasad and J. R. Koseff, "Reynolds number and end wall effects on a lid driven cavity flow," *Physics of Fluids*, vol. 1, no. 2, pp. 208–219, 1989.

[192] B. F. Armaly, F. Durst, J. C. F. Pereira, and B. Schönung, "Experimental and theoretical investigation of backward-facing step flow," *Journal of Fluid Mechanics*, vol. 127, pp. 473–496, 1983.

[193] N. A. Malamataris and R. Löhner, "The computation of the eddy along the upper wall in the three-dimensional flow over a backward-facing step," *International Journal for Numerical Methods in Fluids*, vol. 68, no. 9, pp. 1102–1125, 2012.

[194] P. T. Williams and A. J. Baker, "Numerical simulations of laminar flow over a 3d backward-facing step," *International Journal for Numerical Methods in Fluids*, vol. 24, no. 11, pp. 1159–1183, 1997.

[195] R. M. Fearn, T. Mullin, and K. A. Cliffe, "Nonlinear flow phenomena in a symmetric sudden expansion," *Journal of Fluid Mechanics*, vol. 211, pp. 595–608, 1989.

[196] S. S. Chikatamarla and I. V. Karlin, "Entropy and galilean invariance of the lattice boltzmann theories," *Physical Review Letters*, vol. 97, no. 19, p. 190601, 2006.

[197] S. S. Chikatamarla and I. V. Karlin, "Lattices for the lattice boltzmann method," *Physical Review E*, vol. 79, no. 4, p. 046701, 2009.

A Installing the CGNS library

The following section shows how to install the CGNS library under Ubuntu v12.04 LTS. The general approach on how to install libraries on windows platforms is generally different and not elaborated on. Users wishing to install the CGNS library on windows may refer to www.learncpp.com which has an online C++ course. In Appendix A.1 to A.3 of the course, static and dynamic libraries are discussed and their installation under windows shown.

1. Download the CGNS package from http://cgns.sourceforge.net/download.html (The released used for version 1.0 of the LBM solver has been the 3.1.4 release 2).
2. Open the terminal and navigate to the folder where the downloaded archive is located.
3. To unzip the archive use the command "tar -xzvf filename.tar".
4. Now go to the folder that has been created using "cd foldername/src/".
5. To initiate the installation process type "./configure".
6. After the configuration has been done type "make".
7. After make is done type "sudo make install" and enter the systems password when prompted. The library has been installed if no error messages occur during the installation process. A new folder called lib/ should be created with all the needed libraries
8. Copy the folder /lib to the directory usr/src/ which is located in the root directory. It should have the global path of usr/src/lib.

Now the CGNS library is installed and in place. The integrated development environment (IDE) must be informed where to look for the library. The following example shows how to set up the IDE Code::Blocks to use the CGNS library. It is strongly advised to use Code::Blocks as a project file has been already created that has linked all the necessary source files together.
The following step only needs to be performed once, after that it will be available for all subsequent projects.

1. Go to the menu Settings→Compiler and Debugger
2. Under the tab "search directories", specify /usr/src/lib for the compiler and linker.
3. Go to linker settings and specify the cgns library under link libraries. Add the following path with file to it: /usr/src/lib/libcgns.a

Now the system should be set up to work with the CGNS format.

B Lid driven cavity flow in C++

C++ code for the Lid driven cavity, translated and improved from the original Fortran code from Mohamad [18].

```cpp
/* ------------------------------------------------

   Lid-Driven Cavity Flow

   Example Code for the lattice Boltzmann method
   using a D2Q9 speed model in C++

  ------------------------------------------------- */

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
/* ----------------------------------------------------------

   Step 1: Input data

----------------------------------------------------------- */

   // initialize number of lattices in x and y direction
   const int iNx = 100;
   const int iNy = 100;

   // initialize number of directions for speed model
   const int s_max = 8;

   // initialize dummy variable for loops
   int i, j, k, iTime, tot_t;
   float cu1, cu2, fTemp, fUtemp, fVtemp;

   // initialize lid-velocity, domain size, time step,
   // total timestep, viscosity, Reynolds number and omega

   float u_lid, dx, dy, nu, Re, omega;

   // variables for maximum residual
  // structs should be avoided but are used here for simplicity
   struct res_max
   {
      float f = 0;
      float u = 0;
      float v = 0;
      float rho = 0;
      float f_n1 = 0;
      float u_n1 = 0;
      float v_n1 = 0;
      float rho_n1 = 0;
      float resF;
      float resU;
```

```
        float resV;
        float resRHO;
    };

    struct res_max residuals;


    // now specify value and allocate arrays
    dx = 1.;
    dy = dx;        // assume square cavitiy
    tot_t = 40000;

    nu = 0.1;
    u_lid = 0.32;
    Re = u_lid*iNy/nu;
    omega =1./(3.*nu+0.5);

    // initialize distribution function, eq. distribution function
    // density vector, weighting vector, speed model vectors
    // (macroscopic) velocity vector

    float f_n1[s_max+1][iNx+1][iNy+1]; //= {{{1}}};
    float feq[s_max+1][iNx+1][iNy+1]; // = {{{1}}};
    float f[s_max+1][iNx+1][iNy+1]; // = {{{1}}};
    float rho[iNx+1][iNy+1]; // = {{5}};
    float u[iNx+1][iNy+1]; // = {{0}};
    float v[iNx+1][iNy+1]; // = {{0}};
    float cx[s_max+1] = {0., 1., 0., -1., 0., 1., -1., -1., 1.};
    float cy[s_max+1] = {0., 0., 1., 0., -1., 1., 1., -1., -1.};
    float w[s_max+1] = {4./9., 1./9., 1./9., 1./9., 1./9., 1./36., 1./36., 1./36.,
        1./36.};

    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            for(k=0;k<=s_max;k++)
            {
                f_n1[k][i][j] = 0.01;
                feq[k][i][j] = 0.01;
                f[k][i][j] = 0.01;
            }
            rho[i][j] = 5.0;
            u[i][j] = 0.0;
            v[i][j] = 0.0;
        }
    }

    // assign lid velocity to moving lid

    for (i=1;i<=iNx-1;i++)
    {
        u[i][iNy] = u_lid;
    }

    // now print everything relevant to screen
   cout << endl;
   cout << "Domain size:" << endl;
   cout << "In x-direction:        " << dx << endl;
   cout << "In y-direction:        " << dy << endl;
   cout << endl;
   cout << "Number of lattices (in x): " << iNx << endl;
   cout << "Number of lattices (in y): " << iNy << endl;
   cout << endl;
   cout << "Reynolds number        " << Re << endl;
   cout << "Collision frequency:    " << omega << endl;
   cout << endl;
   cout << endl;

/* -----------------------------------------------------------

    Step 2: Main loop
   ----------------------------------------------------------- */

    for (iTime=0;iTime<tot_t;iTime++)
    {

    // overwrite previous timestep to new array
    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            for(k=0;k<=s_max;k++)
            {
                f[k][i][j] = f_n1[k][i][j];
            }
        }
    }

    // get max values from last iteration for residuals
    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            for(k=0;k<=s_max;k++)
            {
                if (f[k][i][j]>residuals.f)
                {
                    residuals.f = f[k][i][j];
                }
            }
        if (u[i][j]>residuals.u)
        {
            residuals.u = u[i][j];
        }
        if (v[i][j]>residuals.v)
        {
            residuals.v = v[i][j];
        }
        if (rho[i][j]>residuals.rho)
        {
            residuals.rho = rho[i][j];
        }
        }
    }


/* ------------------------------------------------------------

    Step 3: Calculate feq
    Step 4: Collision

   ------------------------------------------------------------ */

    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            cu1 = u[i][j]*u[i][j]+v[i][j]*v[i][j];
            for(k=0;k<=s_max;k++)
            {
                cu2 = u[i][j]*cx[k]+v[i][j]*cy[k];
                feq[k][i][j]=rho[i][j]*w[k]*(1.0+3.0*cu2+4.5*cu2*cu2-1.5*cu1);
                f_n1[k][i][j]=omega*feq[k][i][j]+(1.0-omega)*f[k][i][j];
            }
        }
    }

/* ------------------------------------------------------------

    Step 5: Streaming

   ------------------------------------------------------------ */

    for(j=0;j<=iNy;j++)
    {
        for(i=iNx;i>=1;i--)
        {
            // stream from right to left
            f_n1[1][i][j] = f_n1[1][i-1][j];          // 1
        }
        for(i=0;i<=iNy-1;i++)
        {
            // stream from left to right
            f_n1[3][i][j] = f_n1[3][i+1][j];          // 3
        }
    }

    // Top to bottom
    for(j=iNy;j>=1;j--)
    {
        for(i=0;i<=iNx;i++)
        {
            f_n1[2][i][j] = f_n1[2][i][j-1];          // 2
        }
        for(i=iNx;i>=1;i--)
        {
            f_n1[5][i][j] = f_n1[5][i-1][j-1];        // 5
        }
        for(i=0;i<=iNx-1;i++)
        {
            f_n1[6][i][j] = f_n1[6][i+1][j-1];       // 6
        }
    }

    // Bottom to top
    for(j=0;j<=iNy-1;j++)
    {
        for(i=0;i<=iNx;i++)
        {
            f_n1[4][i][j] = f_n1[4][i][j+1];         // 4
        }
        for(i=0;i<=iNx-1;i++)
        {
            f_n1[7][i][j] = f_n1[7][i+1][j+1];       // 7
        }
        for(i=iNx;i>=1;i--)
        {
            f_n1[8][i][j] = f_n1[8][i-1][j+1];       // 8
        }
    }

/* -----------------------------------------------------------

    Step 6: Calculating f at boundaries

   ----------------------------------------------------------- */

    for(j=0;j<=iNy;j++)
    {
        // bounce back on west face
        f_n1[1][0][j] = f_n1[3][0][j];
        f_n1[5][0][j] = f_n1[7][0][j]-0.5*(f_n1[2][0][j]-f_n1[4][0][j]);
        f_n1[8][0][j] = f_n1[6][0][j]+0.5*(f_n1[2][0][j]-f_n1[4][0][j]);

        // bounce back on east face
        f_n1[3][iNx][j] = f_n1[1][iNx][j];
        f_n1[7][iNx][j] = f_n1[5][iNx][j]+0.5*(f_n1[2][0][j]-f_n1[4][0][j]);
        f_n1[6][iNx][j] = f_n1[8][iNx][j]-0.5*(f_n1[2][0][j]-f_n1[4][0][j]);
    }

    for(i=0;i<=iNx;i++)
    {
        // bounce back on south boundary
        f_n1[2][i][0] = f_n1[4][i][0];
        f_n1[5][i][0] = f_n1[7][i][0]-0.5*(f_n1[1][0][j]-f_n1[3][0][j]);
        f_n1[6][i][0] = f_n1[8][i][0]+0.5*(f_n1[1][0][j]-f_n1[3][0][j]);
    }

    // lid at north boundary
    for(i=1;i<=iNx-1;i++)
    {
        fTemp = (f_n1[0][i][iNy])+(f_n1[1][i][iNy])+(f_n1[3][i][iNy])+
        2*((f_n1[2][i][iNy])+(f_n1[6][i][iNy])+(f_n1[5][i][iNy]));

        f_n1[4][i][iNy] = f_n1[2][i][iNy];
        f_n1[8][i][iNy] = f_n1[6][i][iNy]+0.5*(f_n1[3][0][j]-f_n1[1][0][j])+fTemp*u_lid/6.;
        f_n1[7][i][iNy] = f_n1[5][i][iNy]+0.5*(f_n1[1][0][j]-f_n1[3][0][j])-fTemp*u_lid/6.;
    }

/* ------------------------------------------------------------

    Step 7: Calculating macroscopic values for rho, u and v

   ------------------------------------------------------------ */

    // Density
    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            fTemp = 0;
            for(k=0;k<=s_max;k++)
            {
                fTemp=fTemp+f_n1[k][i][j];
            }
            rho[i][j] = fTemp;
        }
    }
    // Density at the moving lid, special treatment required
    for(i=1;i<=iNx;i++)
    {
        rho[i][iNy]=(f_n1[0][i][iNy])+(f_n1[1][i][iNy])+(f_n1[3][i][iNy])+
        2*((f_n1[2][i][iNy])+(f_n1[6][i][iNy])+(f_n1[5][i][iNy]));
    }

    // U and V velocity
    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            fUtemp = 0;
            fVtemp = 0;

            for(k=0;k<=s_max;k++)
            {
                fUtemp=fUtemp+f_n1[k][i][j]*cx[k];
                fVtemp=fVtemp+f_n1[k][i][j]*cy[k];
            }
            u[i][j] = fUtemp/rho[i][j];
            v[i][j] = fVtemp/rho[i][j];
        }
    }

    // calculate residual
    // get max values from last iteration for residuals
    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            for(k=0;k<=s_max;k++)
            {
                if (f_n1[k][i][j]>residuals.f_n1)
                {
                    residuals.f_n1 = f_n1[k][i][j];
                }
            }
        if (u[i][j]>residuals.u_n1)
        {
            residuals.u_n1 = u[i][j];
        }
        if (v[i][j]>residuals.v_n1)
        {
            residuals.v_n1 = v[i][j];
        }
        if (rho[i][j]>residuals.rho_n1)
        {
            residuals.rho_n1 = rho[i][j];
        }
        }
    }

    residuals.resF = residuals.f_n1-residuals.f;
    residuals.resU = residuals.u_n1-residuals.u;
    residuals.resV = residuals.v_n1-residuals.v;
    residuals.resRHO = residuals.rho_n1-residuals.rho;

    if(residuals.resF < 0)
    {
        residuals.resF *= -1;
    }
    if(residuals.resU < 0)
    {
        residuals.resU *= -1;
    }
    if(residuals.resV < 0)
    {
        residuals.resV *= -1;
    }
    if(residuals.resRHO < 0)
    {
        residuals.resRHO *= -1;
    }


    if (iTime%10==0)
    {
        cout << "_______________________________________________________________________" << endl;
        cout << "It   " << "f              " << "u              " << "v              " << "rho     " << endl;
        cout << "_______________________________________________________________________" << endl;
    }
        printf("%d   %E    %E    %E    %E\n", iTime, residuals.resF, residuals.resU, residuals.resV, residuals.resRHO);

        // reset residuals variables
        residuals.f = 0;
        residuals.u = 0;
        residuals.v = 0;
        residuals.rho = 0;
        residuals.f_n1 = 0;
        residuals.u_n1 = 0;
        residuals.v_n1 = 0;
        residuals.rho_n1 = 0;
    }

/* ------------------------------------------------------------

    Step 8: End main loop
    Step 9: Output data

   ------------------------------------------------------------ */

    ofstream myfile;
    myfile.open ("output_new.dat");
    myfile << "TITLE= \"Mesh\" " << endl;
    myfile << "VARIABLES = X, Y, U, V, RHO, p" << endl;
    myfile << "ZONE " << "I=" << iNx+1 << " J=" << iNy+1 << " F=POINT" << endl;
    for(i=0;i<=iNx;i++)
    {
        for(j=0;j<=iNy;j++)
        {
            myfile << (dx/iNx)*i << " " << dy/(iNy)*j << " " << u[i][j] << " " << v[i][j]
                << " " << rho[i][j] << " " << rho[i][j]/3. << endl;
        }

    }
    myfile.close();

    return 0;
}
```

C Creating a mesh with ICEM CFD

The following section assumes that the reader posses a basic knowledge of ICEM CFD and is capable of creating structured 2D meshes. This section will explain how to create a mesh for the lid driven cavity.

1. Start a new session in ICEM CFD and save the project before proceeding.
2. Go to “geometry” and click on “create point” on the top panel just beneath the menu.
3. First, rename the part to “POINTS” and then enter the following points: (0,0,0), (0,1,0), (1,0,0), (1,1,0). Resize the window to make sure all points are created correctly.
4. Go to “geometry→create/modify curve”. It is mandatory to give each line that is being created a single name. Use the option “from points” and start by creating the south boundary. Type “SOUTH” for the part name and create the line. Next, create 3 more lines for “EAST”, “NORTH” and “WEST”.
5. Go to “geometry→create body” and rename the part to “FLUID”. This will help to distinguish the fluid domain from the boundary conditions. Select the points (0,0,0) and (1,1,0) to create the material point.
6. Go to “blocking→create block”. Select “FLUID” as the part and “2D planar” under “type” for “initialize blocks”. Click apply.
7. Go to “blocking→associate”, then to “associate edge to curve” and associate the south edge to “SOUTH”, the east edge to “EAST” and so on.
8. Go to “blocking→pre-mesh params” and click apply. To see the subsequent changes to the mesh, the pre-mesh needs to be activated from the model view. Expand the “blocking” entry in the model tree and activate the pre-mesh.
9. While still in “pre-mesh params”, go to “edge params” and select the south boundary. Enter 101 in “nodes” and scroll down to activate “copy parameters→to all parallel edges”. Click apply and do the same for the west boundary.
10. To make the change visible, right-click on “pre-mesh” in the “blocking” entry of the model tree and select “recompute”. A fine mesh will be displayed, ready to be exported.
11. Right click on “pre-mesh” again and click on “convert to unstruct mesh”. A new entry “mesh” will be created in the model tree.
12. Go directly to “output→select solver”. Under output solver select “CGNS” and for the “common structural solver” select “ANSYS”. Click apply.
13. Go to “output→boundary conditions” and a new window will appear. All the boundaries will be stored under “edges” while the fluid is stored under “mixed/unknown”. Go to “edges” and expand the entries. Under each boundary condition, click on “create new” and select the boundary condition of type “BCType”. The boundary conditions will become visible

and “BCWall” under “BCType” should be selected. The current version 1.0 of the solver supports “BCWall”, “BCInflow”, “BCOutflow” and “BCGeneral”.

14. The next step is crucial and must be done, otherwise the solver will crash unexpected as it can’t find a fluid domain. Go to “mixed/unknown” and expand fluid. Select “create new” and again use “BCType”. Select “BCGeneral” as the boundary condition. ICEM saves the fluid domain as a boundary condition, hence this work around. Click accept.

15. Go to “output→write output” and follow the instruction on the screen. In the last panel, make sure to select “unstructured” as “input grid type” and to give a suitable name under “output file”. A good practice is to give a short descriptive name which has the amount of nodes along the characteristic length in the filename to be able to reconstruct the Reynolds number later. Type “./lid_101” and leave the other parameters to their default option. Click done.

The CGNS file has been created. It can be viewed using the ADFviewer if it has been compiled together with the CGNS library while installing or a precompiled CGNS browser can be obtained from http://cgns.sourceforge.net/Utilities.html (CGNS viewer by Christian Lundh, windows only). To use it with the lattice Boltzmann solver, copy the mesh to the mesh folder and run the program in the terminal (go to the folder with the compiled version of the code and type “./LBM” to start the program). If the code is being used for the first time, tips from the help menu can be activated which will help to set up the simulation step by step. The parameters $u_{lid} = 0.5$ and $\nu = 0.1$ can be used to simulate the lid driven cavity at Re=500 which should take around 40 000 iterations to converge.

D In-depth review of key methods

This section will clarify some of the code fragments as they would be difficult to understand without any explanation.

D.1 setLookUpTable

The following method is taken from the class cReadMeshICEM:

```
void cReadMeshICEM::setLookUpTable()
{
    cout << "Checking Boundary Conditions............";
    // get number of bases and loop over them
    cg_nbases(index_file,&nbases);

    // allocate memory for arrays
    iBocoList  = new int***[nbases];
    lookUpTable = new int****[nbases];

    for (int i=0;i<nbases;i++)
    {
        index_base = i+1;

        // get number of zones and loop over them
        cg_nzones(index_file,index_base,&nzones);

        iBocoList[index_base] = new int**[nzones];
        lookUpTable[index_base] = new int***[nzones];

        for (int j=0;j<nzones;j++)
        {
            index_zone = j+1;

            // get number of sections
            cg_nsections(index_file,index_base,index_zone,&nsections);

            /*
            Since the fluid phase is written into the boundary conditions as well
            this means that number of sections and boundary conditions must
            coincide. Unfortunatelly ICEM does not provide any information
            on how to distinguish the boundaries from the fluid phase so the
            "risky" assumption of "number of sections" == "number of BC"
            has been done which means that the sections are mapped to the BC
            and the BC information can be called from each section which
            the index_section. If the above is not true, than the programm
            will abort here
            */

            cg_nbocos(index_file,index_base,index_zone,&nbocos);

            if (nsections != nbocos)
            {
                cout << "\n/////////////////////////////////////////////////////" << endl;
                cout << "// Warning, error occured during mesh generation. //" << endl;
                cout << "// Number of sections and boundaries do not fit. //" << endl;
                cout << "// Please assure that all boundaries and fluid //" << endl;
48                  cout << "// phases have been created properly.       //" << endl;
49                  cout << "////////////////////////////////////////////////////////" << endl;
50                  return;
51              }
52              else
53              {
54                  cout << "DONE!" << endl;
55              }
56  
57              lookUpTable[index_base][index_zone] = new int**[nsections];
58  
59              for(int l=0;l<nsections;l++)
60              {
61                  index_section = l+1;
62  
63                  // find out what kind of elements domain consist of
64                  cg_section_read(index_file,index_base,index_zone,index_section,sectionname,
65                  &itype,&istart,&iend,&nbndry,&iparent_flag);
66  
67                  // get elementnumber (i.e. 2 for BAR_2, 4 for QUAD_4 etc.)
68                  cg_npe(itype,&npe);
69  
70                  Element = new int[(iend-istart+1)*npe];
71  
72                  // get Element data
73                  cg_elements_read(index_file,index_base,index_zone,index_section,
74                  &Element[0],&ParentData);
75  
76                  lookUpTable[index_base][index_zone][index_section] = new
                         int*[iend-istart+1];
77  
78                  // write Elements to global lookUpTable to be accessed by other classes
79                  IncreaseCounter = 0;
80                  for (int m=0;m<(iend-istart+1);m++)
81                  {
82                      lookUpTable[index_base][index_zone][index_section][m] = new int[npe];
83                      for (int n=0;n<npe;n++)
84                      {
85                          lookUpTable[index_base][index_zone][index_section][m][n] =
                                 Element[IncreaseCounter];
86                          IncreaseCounter += 1;
87                      }
88                  }
89              }
90          }
91      }
92  }
```

This method gets the look up table from the unstructured mesh and reads it into the variable "Element" in line 73-74. This is a one dimensional array for different kinds of meshes, i.e. 1D/2D and 3D. To give it a structure, the "Element" variable is written into a two dimensional form, where the first index is the cell number and the second element the cell element. For a quad4 element, the second index would be always four. The "lookUpTable" array is five dimensional in nature but the first three dimensions are only due to the specifics of the CGNS format, i.e. the first three indices are counting the bases, zones and sections.

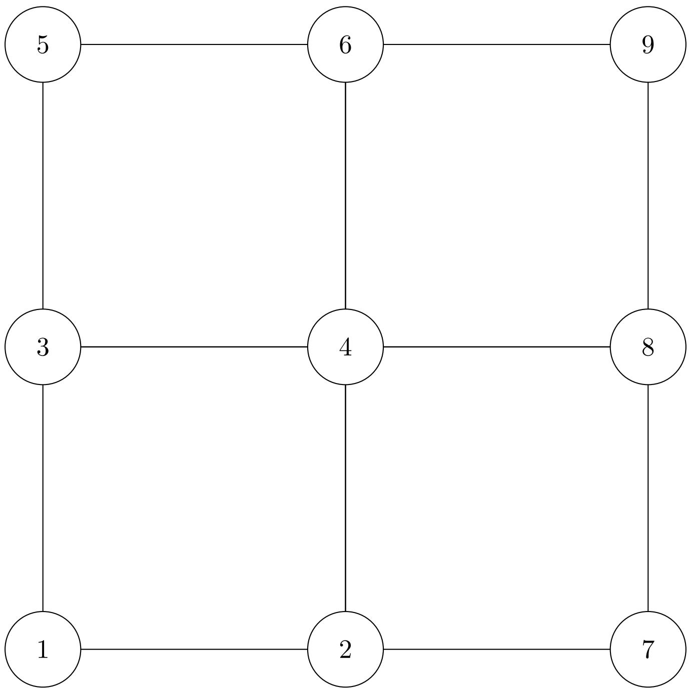


Figure 27 – Coarse mesh that shows the numbering of the vertices for the look up table in ICEM CFD 14.0

Figure 27 shows a coarse mesh of 4 cells of type quad4 and its corresponding vertex numbering. The numbering of a cell is always done in a counter clock-wise direction so that the normal vectors are pointing into the domain. Taking this figure as an example, the "Element" vector and "lookUpTable" matrix would take the shape of:

$$\text{Element} = (1 \quad 2 \quad 4 \quad 3 \quad 3 \quad 4 \quad 6 \quad 5 \quad 2 \quad 7 \quad 8 \quad 4 \quad 4 \quad 8 \quad 9 \quad 6)^T \tag{D.1}$$

$$\text{lookUpTable} = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 3 & 4 & 6 & 5 \\ 2 & 7 & 8 & 4 \\ 4 & 8 & 9 & 6 \end{pmatrix} \tag{D.2}$$

The only difference, as can be seen, is that the "lookUpTable" contains a more structured format which is necessary for the next method, setConnection.

D.2 setConnection

The key fragment of the code for the method setConnection is given below. Npe stands for "number per element" and is the number of points that each element has. For quad4, npe is 4 and for bar2, npe is 2. maxSize is an array containing information about the number of vertices and cells, in this case $maxSizes[index\_base][index\_zone][1][0]$ equals the number of cells in the domain.

```
for(int l=0;l<maxSizes[index_base][index_zone][1][0];l++)
{
   for(int m=0;m<npe;m++)
   {
      if (m == 0)
      {
         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][0] =
             lookUpTable[index_base][index_zone][index_section][l][0];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][1] =
             lookUpTable[index_base][index_zone][index_section][l][3];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][2] =
             lookUpTable[index_base][index_zone][index_section][l][2];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][3] =
             lookUpTable[index_base][index_zone][index_section][l][1];
         }
      else if(m == 1)
      {
         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][0] =
             lookUpTable[index_base][index_zone][index_section][l][1];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][1] =
             lookUpTable[index_base][index_zone][index_section][l][2];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][7] =
             lookUpTable[index_base][index_zone][index_section][l][0];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][8] =
             lookUpTable[index_base][index_zone][index_section][l][3];
      }
      else if(m == 2)
      {
         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][0] =
             lookUpTable[index_base][index_zone][index_section][l][2];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][5] =
             lookUpTable[index_base][index_zone][index_section][l][1];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][6] =
             lookUpTable[index_base][index_zone][index_section][l][0];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][7] =
             lookUpTable[index_base][index_zone][index_section][l][3];
      }
      else if(m == 3)
      {
         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][0] =
            lookUpTable[index_base][index_zone][index_section][l][3];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][3] =
            lookUpTable[index_base][index_zone][index_section][l][2];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][4] =
            lookUpTable[index_base][index_zone][index_section][l][1];

         connection[index_base][index_zone][index_section][lookUpTable[index_base][index_zone]
         [index_section][l][m]-1][5] =
            lookUpTable[index_base][index_zone][index_section][l][0];
      }
   }
}
```

It is found in the cReadMeshICEM class, just like the setLookUpTable method. This section of the code can be regarded as an semi-mesh generator as the LBM requires information of all the neighbours instead of only the neighbours to the north, east, south and west. So what it does is to set up an array, called connection, which is again five dimensional which has again the first three indices reserved for the CGNS format. The fourth index has the size of the number of vertices, so it spans over all points in the domain and the fifth index is depending on the speed model. For the simple D2Q9 case, the fifth index would have 9 elements. The first being always the vertex number itself and then from index 1-8 the neighbours are stored, beginning at the north element and proceeding in a clock-wise direction. If one of the links is outside of the domain, it will not get updated by this method and the initialized value of zero will be displayed instead. To put this into an example, the connection array at vertex number 4 would look as follows:

$$\begin{aligned}
\text{connection}[1][1][1][3][0] &= 4\\
\text{connection}[1][1][1][3][1] &= 6\\
\text{connection}[1][1][1][3][2] &= 9\\
\text{connection}[1][1][1][3][3] &= 8\\
\text{connection}[1][1][1][3][4] &= 7\\
\text{connection}[1][1][1][3][5] &= 2\\
\text{connection}[1][1][1][3][6] &= 1\\
\text{connection}[1][1][1][3][7] &= 3\\
\text{connection}[1][1][1][3][8] &= 5
\end{aligned} \tag{D.3}$$

Here it was assumed that the information was written to base = zone =section = 1. It should be noted that the CGNS format starts counting at the number one, against the C++ convention which is to start with zero. This can easily cause confusion. This is also the reason for the "[m]-1" entries in the source code above as the numbering has to be brought from CGNS to C++ standard. To complete this section, the connection array at vertex 1 should be given as well to see how this array behaves at boundaries. The same assumption about the bases, zones and sections hold and the array takes the form of:

$$\begin{aligned}
\text{connection}[1][1][1][0][0] &= 1\\
\text{connection}[1][1][1][0][1] &= 3\\
\text{connection}[1][1][1][0][2] &= 4\\
\text{connection}[1][1][1][0][3] &= 2\\
\text{connection}[1][1][1][0][4] &= 0\\
\text{connection}[1][1][1][0][5] &= 0\\
\text{connection}[1][1][1][0][6] &= 0\\
\text{connection}[1][1][1][0][7] &= 0\\
\text{connection}[1][1][1][0][8] &= 0
\end{aligned} \tag{D.4}$$

As stated before, the first index always gives the current vertex number, in this case one and four in the case before. The elements outside of the domain do not exist so they have a value of zero. This is important for the streaming so that it can be checked, whether the neighbour is inside the domain connection$[1][1][1][0][i] \neq 0$ or outside connection$[1][1][1][0][i] = 0$. The details will be given in the next section.

D.3 D2Q9

Once the concept of the connection array is understood, it is easy to understand how the streaming works. It checks if current node has a neighbour by checking if the corresponding link is 0 or not. If not, than it streams its information to the neighbour or, if it is zero, it just overwrites its value to the array. The value at the center is just overwritten and does not stream. The source code for the streaming of the D2Q9 speed model is as follows:

```
float**** cStreaming::streamD2Q9(int nbases, int nzones, int nsections, int*** checkBCArr,
int**** iSizes, int***** connection, float**** fn1, float**** fn05)
{
    for(int i=0;i<nbases;i++)
    {
        index_base = i+1;
        for(int j=0;j<nzones;j++)
        {
            index_zone = j+1;
            for(int k=0;k<nsections;k++)
        {
            index_section = k+1;
            if(checkBCArr[index_base][index_zone][index_section] == 0)
            {
                for(int m=0;m<iSizes[index_base][index_zone][0][0];m++)
                {
                    /*
                    stream to neighbor node if neighbor node is not ZERO (zero means
                        does not exist, i.e. is outside of the domain)
                    if it is ZERO (i.e. outside of the domain) then assign the value
                        obtained in the collision process
                    this is necessary because there are always two links which do not
                        stream in corners however they have to be assigned
                    a value to satisfy conservation mass. The consequences would be two
                        missing links which do not contribute to the
                    calculation of density hence the overall density will always be
                        lower in corner nodes compared to neighbor nodes.
                    Since density is proportinal to pressure as p=rho/3 this means that
                        the pressure is always lower in the corners
                    and hence fluid will exit the domain at corners (fluid is leaking
                        out of the domain in the corners).
                    The second consequence is that this lower density information will
                        propagate into the domain through streaming at
                    future timesteps giving non-physical results in the vecinity of
                        corners. Practical example is the lid driven cavity,
                    here the secondary vortecis in the lower right and left hand
                        corners are not calculated properly.

                    In the end the non moving distribution function in the center is
                        updated.
                    */
                    if(connection[index_base][index_zone][index_section][m][5] != 0)
                    {
                        fn1[index_base][index_zone][m][1] =
                            fn05[index_base][index_zone][connection[index_base]
                        [index_zone][index_section][m][5]-1][1];
                    }
                    else if(connection[index_base][index_zone][index_section][m][5] ==
                        0)
                    {
                        fn1[index_base][index_zone][m][1] =
                            fn05[index_base][index_zone][m][1];
                    }

                    if(connection[index_base][index_zone][index_section][m][6] != 0)
                    {
                        fn1[index_base][index_zone][m][2] =
                            fn05[index_base][index_zone][connection[index_base]
                        [index_zone][index_section][m][6]-1][2];
                    }
                    else if(connection[index_base][index_zone][index_section][m][6] ==
                        0)
                    {
                        fn1[index_base][index_zone][m][2] =
                            fn05[index_base][index_zone][m][2];
                    }

                if(connection[index_base][index_zone][index_section][m][7] != 0)
                {
                    fn1[index_base][index_zone][m][3] =
                        fn05[index_base][index_zone][connection[index_base]
                    [index_zone][index_section][m][7]-1][3];
                }
                else if(connection[index_base][index_zone][index_section][m][7] ==
                     0)
                {
                    fn1[index_base][index_zone][m][3] =
                        fn05[index_base][index_zone][m][3];
                }

                if(connection[index_base][index_zone][index_section][m][8] != 0)
                {
                    fn1[index_base][index_zone][m][4] =
                        fn05[index_base][index_zone][connection[index_base]
                    [index_zone][index_section][m][8]-1][4];
                }
                else if(connection[index_base][index_zone][index_section][m][8] ==
                     0)
                {
                    fn1[index_base][index_zone][m][4] =
                        fn05[index_base][index_zone][m][4];
                }

                if(connection[index_base][index_zone][index_section][m][1] != 0)
                {
                    fn1[index_base][index_zone][m][5] =
                        fn05[index_base][index_zone][connection[index_base]
                    [index_zone][index_section][m][1]-1][5];
                }
                else if(connection[index_base][index_zone][index_section][m][1] ==
                     0)
                {
                    fn1[index_base][index_zone][m][5] =
                        fn05[index_base][index_zone][m][5];
                }

                if(connection[index_base][index_zone][index_section][m][2] != 0)
                {
                    fn1[index_base][index_zone][m][6] =
                        fn05[index_base][index_zone][connection[index_base]
                    [index_zone][index_section][m][2]-1][6];
                }
                else if(connection[index_base][index_zone][index_section][m][2] ==
                     0)
                {
                    fn1[index_base][index_zone][m][6] =
                        fn05[index_base][index_zone][m][6];
                }

                if(connection[index_base][index_zone][index_section][m][3] != 0)
                {
                    fn1[index_base][index_zone][m][7] =
                        fn05[index_base][index_zone][connection[index_base]
                    [index_zone][index_section][m][3]-1][7];
```

```
                        }
                        else if(connection[index_base][index_zone][index_section][m][3] == 0)
                        {
                            fn1[index_base][index_zone][m][7] = fn05[index_base][index_zone][m][7];
                        }

                        if(connection[index_base][index_zone][index_section][m][4] != 0)
                        {
                            fn1[index_base][index_zone][m][8] = fn05[index_base][index_zone][connection[index_base]
                            [index_zone][index_section][m][4]-1][8];
                        }
                        else if(connection[index_base][index_zone][index_section][m][4] == 0)
                        {
                            fn1[index_base][index_zone][m][8] = fn05[index_base][index_zone][m][8];
                        }

                        // update center value here
                        fn1[index_base][index_zone][m][0] = fn05[index_base][index_zone][m][0];
                    }
                }
            }
        }
    }
    return fn1;
}
```

Structured meshes offer the flexibility of overwriting the distribution function vector in a way that it does not need a second array as in the code above. For unstructured meshes, the extra amount of storage has to be accepted in order to not overwrite information accidentally.