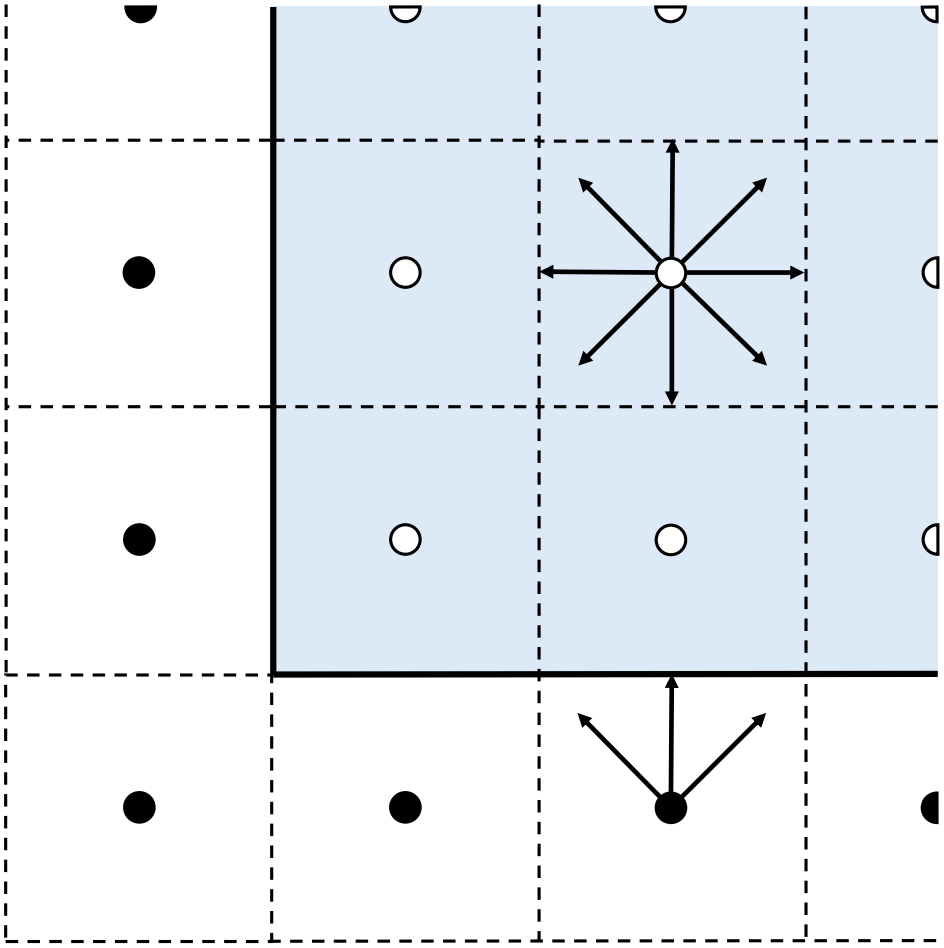


Lattice-Boltzmann for Aeronautical Flows

An introduction to and evaluation of the Lattice-Boltzmann Method

Master's thesis in Applied Mechanics

EMIL ELLÉNIUS

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's Thesis in Applied Mechanics

Lattice-Boltzmann for Aeronautical Flows

An introduction to and evaluation of the Lattice-Boltzmann Method

EMIL ELLÉNIUS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Fluid Dynamics
Chalmers University of Technology
Gothenburg, Sweden 2024

Lattice-Boltzmann for Aeronautical Flows
An introduction to and evaluation of the Lattice-Boltzmann Method
EMIL ELLÉNIUS

Supervisor: Magnus Carlsson, Saab Aeronautics
Examiner: Huadong Yao, Department of Mechanics and Maritime Sciences

Master's Thesis 2024
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Computational cells in a lattice at a corner boundary. Mesoscopic particle velocity populations are shown at an interior and a boundary lattice node.

Typeset in LaTeX
Gothenburg, Sweden 2024

Lattice Boltzmann for Aeronautical Flows
An introduction to and evaluation of the standard LBM
EMIL ELLÉNIUS
Department of Mechanics and Maritime Sciences
Division of Fluid Dynamics
Chalmers University of Technology

Abstract

In aircraft design, there is a need for accurate, efficient and robust computational fluid dynamics (CFD) simulations. Industry dominated methods are based on the non-linear Navier-Stokes equations which are rather computationally expensive to solve. The Lattice-Boltzmann method (LBM) is an alternative CFD method that has risen in popularity lately due to the promised performance gain resulting from its linear equations. The method describes the evolution of a particle distribution function (PDF) at the meso-scale through the Boltzmann equation. The PDF is a statistical function describing the probability of finding a particle with a certain velocity at a certain location in time and space and is connected to the macro-scale through integrals over velocity space. In the standard LBM, the discretisation of the Boltzmann equation involves expressing the PDF at equilibrium through a truncated polynomial expansion. This allows for exact computation of the macroscopic density and fluid velocity through finite sums and a limited set of particle velocities. However, the truncation introduces an error scaling with the Mach number, limiting the method to Ma $\lesssim 0.3$. There is also a correlation between the viscosity, grid spacing and time step. To simulate high Reynolds number (Re) flows the grid must therefore be very fine, which adds computational cost.

In this master's thesis, the standard LBM has been evaluated for aeronautical applications. It was implemented in Python, where part of the work focused on increasing the performance resulting in 30 times faster code. The Euler equations were used as a baseline, but since the standard LBM is always viscous there were difficulties reaching good correspondence. Partly, this was due to using simple boundary conditions (BCs), but a great improvement could be shown through a proposed modification. The limitation in Re was still an issue, however, and the conclusion is that more advanced BCs should be used for arbitrary geometries. Through a minor modification to the equilibrium PDF, an Euler equation test case for isentropic vortex convection was successfully simulated, although with some viscous dissipation present. The stability of the method was also explored, finding that the Ma limit was stricter at low viscosities since the method operates closer to its stability limit there. Lastly, the initialisation proved another challenge due to the interplay between the macro- and meso-scales, often leading to polluting the solution with numerical acoustic noise. It is possible to create non-reflecting BCs, but stability problems where the solution diverges were encountered when using established methods, leading to the development of new boundary treatments.

Keywords: Lattice-Boltzmann, CFD, aeronautical, Euler equations, non-reflecting.

Preface

This report presents the outcome of the master's thesis project carried out at the Department of Mechanics and Maritime Sciences at Chalmers University of Technology during the spring of 2024.

Acknowledgements

First and foremost, I would like to thank my supervisor Magnus Carlsson and Examiner Huadong Yao for the continuous support and regular meetings with good discussions throughout the project. Their experience in CFD and suggestions for suitable test cases where invaluable feedback to the project. Especially thanks to Huadong as it is far beyond of what the role of examiner formally requires. I would also like to thank Per Weinerfelt at Saab for really pushing me to understand every aspect of the mathematical derivation of the method, proposing ideas for extentions such as the isentropic thermal LBM with excellent discussions as a result. Then, I want to thank Sebastian Arvidsson, along with Magnus once again, for selecting me as the candidate for this thesis work, putting their trust in my ability to do a good job. Furthermore, I would like to thank Mattias at Saab for the supplied code and finally, my thesis worker colleague Rasmus Olausson for good company and useful tips for parts of the code development, as he is a far more seasoned programmer than me.

Emil Ellénius, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BB	Bounce-back
BC	Boundary condition
BGK	Bhatagnar-Gross-Krook
CBC	Characteristic boundary condition
CFD	Computational fluid dynamics
CPU	Central processing unit
DDF	Dual distribution function
DdQq	d-dimensional set of q velocities
DOF	Degrees of freedom
GPU	Graphics processing unit
HP	Hermite polynomial
IC	Initial condition
LBE	Lattice-Boltzmann equation
LBM	Lattice-Boltzmann method
LODI	Local one-dimensional inviscid
MRT	Multi-relaxation time
NS	Navier-Stokes
NSE	Navier-Stokes equations
ODE	Ordinary differential equation
PDE	Partial differential equation
PDF	Particle distribution function
TRT	Two-relaxation time

Nomenclature

Below is the nomenclature of indices, parameters, constants, variables, functions and non-dimensional numbers that have been used throughout this thesis. Although many parameters, variables and functions are frequently used in non-dimensional forms, the units of their dimensional versions are specified here.

Indices

0	Index for reference value
∞	Index for free-stream or far-field value
$i,\ j$	Index for velocity in DdQq
$\bar{i}$	Index for velocity opposite of i
$\alpha,\ \beta,\ \gamma,$ $\alpha_1, \ldots, \alpha_n$	Indices for spatial dimensions using Einstein summation convention
$x,\ y,\ z$	Spatial dimensions
t	Time dimension
b	First node inside boundary
w	Value at the boundary/wall
n	Normal direction
t	Tangential direction
f	Density distribution function
g	Potential energy distribution function

Parameters

d	Number of dimensions [-]
q	Number of velocities [-]
R	Specific gas constant [J/(kg $\cdot$ K)]

c_V	Specific heat capacity at constant volume [J/K]
c_p	Specific heat capacity at constant pressure [J/K]
γ	Specific heat ratio [-]
ν	Kinematic viscosity [m^2/s]
n_{DOF}	Number of degrees of freedom [-]
K	Number of internal degrees of freedom [-]
τ	Relaxation time [s]
t_c	Collision time scale [s]
t_{mfp}	Average time between collisions [s]
ℓ	Length scale [m]
ℓ_{mfp}	Average length between collisions [m]
L	Length [m]
h	Height [m]
r	Radius [m]
c_s	Speed of sound [m/s]
$\mathcal{U}$	Velocity scale [m/s]
C_ϕ	Unit conversion factor for flow variable ϕ [units of ϕ]
ϵ	Smallness parameter in Kn [-]
$N,\ n$	Integers [-]
k	Relaxation parameter [-]
σ	Tuning parameter [-]
χ	Interpolation parameter [-]
$\mathcal{K}$	Vortex strength [-]
α	Vortex decay [-]
a	Line slope parameter [-]
b	Line y-intercept parameter [-]

Constants

C	Constant [varying units]
w_i	Stencil weights [-]
k_B	Boltzmann's constant [J/K]
e	Euler's constant [-]
S_{d-1}	Surface area of unit $d-1$-sphere [m^{d-1}]

Variables

$\boldsymbol{\sigma} = [\sigma_{\alpha\beta}]$	Stress tensor [Pa]
$\boldsymbol{F} = [F_\alpha]$	Body force [N/m^3]
$\boldsymbol{x} = [x_\alpha]$	Position [m]
$\boldsymbol{\xi} = [\xi_\alpha]$	Continuous particle velocity [m/s]
$\boldsymbol{\xi}_i = [\xi_{i\alpha}]$	Discrete particle velocity [m/s]
$\boldsymbol{u} = [u_\alpha]$	Fluid velocity [m/s]
$\boldsymbol{v} = [v_\alpha]$	Relative velocity [m/s]
$\boldsymbol{c}_i = [c_{i\alpha}]$	Discrete lattice velocity [m/s]
$\boldsymbol{q} = [q_\alpha]$	Coordinate or momentum [varying units]
$\boldsymbol{\eta}$	Vector containing internal degrees of freedom [m/s]
$\boldsymbol{m}$	Vector of primitive variables [units of the contained variables]
$\boldsymbol{\Gamma}$	Matrix of primitive variables [units of the contained variables]
$\boldsymbol{S}$	Matrix of eigenvectors [varying units]
$\boldsymbol{\Lambda}$	Matrix of eigenvalues [m/s]
$\boldsymbol{\mathcal{L}}$	Vector of amplitude variations [varying units]
λ	Eigenvalue [m/s]
t	Time [s]
p	Pressure [Pa]
ρ	Density [kg/m^3]
ϕ	General flow variable [varying units]
F_i	Force term [kgs^{2}m^{-6}]
S_i	Force source term [kgs^{2}m^{-6}]
s	Entropy [J/K]
T	Temperature [K]
θ	Temperature factor [J/kg]
Ω	Multiplicity [-]
S	Thermodynamical state [-]
U	Internal energy [J]
V	Volume [m^3]
$\mathcal{N}$	Number of particles [-]
m	Mass [kg]
E	Energy [J]

Z	Partition function [-]
e	Specific internal energy [J/kg]

Functions

$\mathcal{P}(S)$	Probability of state S [-]
$\mathcal{D}(\lvert\boldsymbol{v}\rvert)$	Maxwell speed distribution [$\mathrm{s}^3\mathrm{m}^{-3}$]
$f(\boldsymbol{x}, \boldsymbol{\xi}, t)$	Particle distribution function [$\mathrm{kgs}^3\mathrm{m}^{-6}$]
$f^{\mathrm{eq}}(\boldsymbol{x}, \lvert\boldsymbol{v}\rvert, t)$	Equilibrium distribution function [$\mathrm{kgs}^3\mathrm{m}^{-6}$]
$g(\boldsymbol{x}, \boldsymbol{\xi}, t)$	Potential energy distribution function [$\mathrm{Js}^3\mathrm{m}^{-6}$]
$g^{\mathrm{eq}}(\boldsymbol{x}, \lvert\boldsymbol{v}\rvert, t)$	Potential energy equilibrium distribution [$\mathrm{Js}^3\mathrm{m}^{-6}$]
$\Omega(f)$	Collision operator [$\mathrm{kgs}^2\mathrm{m}^{-6}$]
$\Gamma(z)$	Gamma function (z complex number) [-]
$\boldsymbol{H}^{(n)}(\boldsymbol{x}) = [H^{(n)}_{\alpha_1 \dots \alpha_n}(\boldsymbol{x})]$	d-dimensional Hermite polynomial of n-th order [-]
$\boldsymbol{a}^{(n)}(\rho, \boldsymbol{u}, \theta) = [a^{(n)}_{\alpha_1 \dots \alpha_n}(\rho, \boldsymbol{u}, \theta)]$	Hermite series expansion coefficient of n-th order [-]
$P^{(N)}(\boldsymbol{x})$	Multi-dimensional polynomial of N-th order [-]
$Q(\boldsymbol{x})$	Scalar-valued multi-variate polynomial [-]
$\boldsymbol{R}(\boldsymbol{x})$	Composed polynomial [-]

Non-dimensional numbers

$\mathrm{Kn} = \frac{\ell_{\mathrm{mfp}}}{\ell}$	Knudsen number
$\mathrm{Ma} = \frac{u}{c_s}$	Mach number
$\mathrm{Re} = \frac{u_0 \ell}{\nu}$	Reynolds number
$\mathrm{Pr} = \frac{\nu}{\alpha}$	Prandtl number (α=thermal diffusivity)

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

In aircraft design, there is a considerable need for both simulation and testing. Complex flows are regularly encountered and therefore, the computational fluid dynamics (CFD) tools need to be accurate, efficient, and robust. In cases of heavy unsteady flow, extensive time and computer resources are required and expensive experimental testing is often used instead. To cut cost and development times, new efficient methods to simulate these parts of the flight envelope must be developed.

The industry dominated CFD methods are based on the Navier Stokes equations (NSE), which are a set of nonlinear partial differential equations (PDE) and are thus rather computationally expensive to solve. An alternative to NSE that has risen in popularity lately is the Lattice Boltzmann method (LBM). By reformulating the equations of fluid mechanics at the smaller mesoscale, using statistical mechanics rather than treating the fluid as a continuum, the resulting evolution equation is only nonlinear locally and completely linear in its non-local interactions, making the method especially suited for significant performance gains through means of parallel computing. However, in the standard formulation of LBM, the Mach number is assumed to be low (below 30 % of the speed of sound). In aeronautical applications, this assumption rarely holds, and the LBM therefore needs special treatment in these regimes. [1], [2]

1.2 Purpose

The objective of the master's thesis is to systematically explore and evaluate the Lattice Boltzmann method and possible extensions for aeronautical flows. The work also aims to gain experience in practical usage of the LBM, noting which challenges might emerge and finding solutions for these. The final product should serve as an introduction to LBM for anyone wanting to apply this method to aeronautical applications, such as Saab Aeronautics, the employer of this thesis work.

1.3 Goals

Based on the purpose, the following questions, which the work aims to answer, are formulated:

- Does the Lattice Boltzmann method have advantageous properties for typical aeronautical flows?
- Can the Lattice Boltzmann method be used to simulate inviscid flow, commonly found in benchmark cases for compressible high Mach number flows?
- When does the standard formulation of the Lattice Boltzmann method break down? (What Mach numbers? Does it depend on the type of problem?)
- How can the Lattice Boltzmann method be extended to higher Mach numbers?
- Are there other challenges to using the method and how can these be alleviated?

1.4 Limitations / Demarcations

Due to the time constraint of one semester, only the standard LBM as well as possibly some minor extensions will be implemented. The evaluation will mostly focus on comparing the LBM with the Euler equations (no viscosity), although no actual simulations in traditional CFD software will be performed as part of the project. Unexpectedly large challenges might limit the scope further.

2

Theory

In the following sections the theory and derivation of the basic **Lattice-Boltzmann method (LBM)**, as well as necessary theory for its extensions, are presented. The LBM derivation closely follows the one in the excellent book written by *T. Krüger et al.* [1], albeit in a more condensed format. For the basics in statistical mechanics, including the derivation of the Maxwellian speed distribution, the more detailed explanations in *D. V. Schroeder's* textbook in thermal physics [3] were used as a basis.

2.1 Presumptions

The reader is assumed to be familiar with basic continuum based fluid dynamics and the **Navier-Stokes equations (NSE)** as well as the fundamentals of thermodynamics. All of the required theory in statistical mechanics will be explained. Some parts of this chapter are also heavy in tensor notation, using the Einstein summation convention in the spatial indices α, β.

2.2 The Underlying Theory of LBM

In this section, the theory on which the Lattice-Boltzmann method is based on is explained. This includes a definition of the mesoscopic scale, the equation of state for the standard LBM and an introduction to statistical mechanics and kinetic theory.

2.2.1 Relevant Scales

The LBM is formulated at the **mesoscopic** scale, which lies between the *microscopic* and *macroscopic* scales. The microscopic scale is defined here as the scale at which molecular interactions describe the equations of motion, while macroscopic is the usual scale at which continuum based descriptions like the usual NSE are formulated. At the mesoscopic scale, every individual molecules is not kept track of, but instead *distributions* of molecules using the theory of statistical mechanics. This scale is still small enough for the continuum hypothesis not to hold and consequently rendering macroscopic variables such as density, fluid velocity and temperature void of any physical relevance. However, since the macroscopic quantities are the ones that are possible to measure in real life experiments, an essential part of LBM is to be able to link the mesoscopic and macroscopic descriptions. [1]

2.2.2 Equations of State

The *four* conservation equations consisting of the continuity equation together with the three spatial components of the full compressible NSE does not form a closed system for the *five* unknown macroscopic variables density ρ, pressure p and fluid velocity components u_x, u_y, u_z. (Although four equations are enough to solve the incompressible NSE, since momentum is then decoupled from energy.) Therefore an **equation of state** such as the famous ideal gas law

$$p = \rho R T, \tag{2.1}$$

where R is the **specific gas constant** and T the temperature, is needed as a necessary step to close the system. It is also possible to express the pressure, using the ideal gas law, as a function of density and entropy s:

$$\frac{p}{p_0} = \left(\frac{\rho}{\rho_0}\right)^{\gamma} \mathrm{e}^{(s-s_0)/c_V}, \tag{2.2}$$

where the constants with index 0 refer to the values at some reference state. The specific heat ratio γ is defined as the ratio of the specific heat capacity at constant pressure c_p and at constant volume c_V. For an ideal gas they are related through

$$c_p = c_V + R. \tag{2.3}$$

Neither of the Equations (2.1) and (2.2) solve the closure problem by themselves, however, as they both introduce another new variable. This problem could be eliminated using the energy equation as a fifth conservation equation, but a simpler way is to instead linearise the equation as deviations from a reference state. Applying this strategy to Eq. (2.2) yields

$$p = p_0 + p' \approx p_0 + \left(\frac{\partial p}{\partial \rho}\right)_s \rho' + \left(\frac{\partial p}{\partial s}\right)_\rho s' \tag{2.4}$$

with primed variables as the deviations. Assuming constant entropy further simplifies Eq. (2.4) to

$$p = p_0 + c_s^2 \rho' \tag{2.5}$$

as the speed of sound c_s is given by the relation

$$c_s^2 = \left(\frac{\partial p}{\partial \rho}\right)_s. \tag{2.6}$$

Finally, noting that the reference pressure p_0 does not enter the NSE, allows for the reformulation of Eq. (2.5) as

$$p = c_s^2 \rho, \tag{2.7}$$

in which p_0 has been redefined as $c_s^2 \rho_0$. This is the equation of state used in the standard LBM. [1]

2.2.3 Statistical Mechanics

The state of a thermodynamical system could be divided into a microstate and a macrostate. Knowing the microstate is equivalent to knowing the state of every part of the system, while the macrostate represent a more general knowledge, such as the total energy of the system, but not how exactly it is distributed. Therefore, there could be many microstates corresponding to a single macrostate. This number is called the **multiplicity** Ω of the system and is in general very large for a system consisting of many molecules. Assuming all microstates to be equally probable leads to the explanation of why irreversible processes happen; by arguments of simple statistics, the process will tend to the most probable macrostate, i.e. the one with the highest multiplicity. The entropy can now be defined as

$$s := k_B \ln \Omega, \tag{2.8}$$

where k_B is the Boltzmann constant. By virtue of the aforementioned statistical argument it is easy to see from Eq. (2.8) why the second law of thermodynamics $ds \geq 0$ must hold. [3]

2.2.3.1 The Boltzmann Factor

For a system in thermal equilibrium with a reservoir (a system large enough to resist temperature changes) at a specified temperature, the ratio of probabilities for two states S_1 and S_2 is equal to the ratio of the multiplicities of the reservoir Ω_R when the system is at these states:

$$\frac{\mathcal{P}(S_1)}{\mathcal{P}(S_2)} = \frac{\Omega_R(S_1)}{\Omega_R(S_2)}. \tag{2.9a}$$

The reasoning is that, for the specified states, these multiplicities represent the total number of possible states for the combined system, which is an isolated system and therefore has an equal probability for all states. Using Eq. (2.8) the ratio of probabilities can be rewritten in terms of the entropy of the reservoir s_R:

$$\frac{\mathcal{P}(S_1)}{\mathcal{P}(S_2)} = \frac{\mathrm{e}^{s_R(S_1)/k_B}}{\mathrm{e}^{s_R(S_2)/k_B}} = \mathrm{e}^{\left(s_R(S_1) - s_R(S_2)\right)/k_B}. \tag{2.9b}$$

Now, the ratio of probabilities is expressed in terms of a difference in entropy and for a reservoir, which is much larger than the system, the difference will be really small. Therefore, the thermodynamical identity

$$ds = \frac{1}{T}(\mathrm{d}U + p\,\mathrm{d}V - \mu\,\mathrm{d}\mathcal{N}) \tag{2.10}$$

can be invoked. The equation relates the change of entropy with the change in internal energy U, volume V and number of particles $\mathcal{N}$. When there is no exchange of particles, the $\mu\,\mathrm{d}\mathcal{N}$ term is zero and the volume change is usually negligible compared to the change in energy, allowing for the removal of that term as well. Noting that the change in one quantity in the reservoir is equal to minus the change

in the same quantity in the system, the difference of entropy in Eq. (2.9b) can be rewritten in terms of the energy E of the system:

$$s_R(S_1) - s_R(S_2) = \frac{1}{T}\Big(U_R(S_1) - U_R(S_2)\Big) = -\frac{1}{T}\Big(E(S_1) - E(S_2)\Big). \tag{2.11}$$

Putting this back into Eq. (2.9b) nets

$$\frac{\mathcal{P}(S_1)}{\mathcal{P}(S_2)} = \mathrm{e}^{-\big(E(S_1)-E(S_2)\big)/(k_B T)} = \frac{\mathrm{e}^{-E(S_1)/(k_B T)}}{\mathrm{e}^{-E(S_2)/(k_B T)}}. \tag{2.12}$$

This equation involves the **Boltzmann factor** $\mathrm{e}^{-E(S)/(k_B T)}$ and it is also completely separable in the two states leading to the constant value

$$\frac{\mathcal{P}(S_1)}{\mathrm{e}^{-E(S_1)/(k_B T)}} = \frac{\mathcal{P}(S_2)}{\mathrm{e}^{-E(S_2)/(k_B T)}} := \frac{1}{Z}, \tag{2.13}$$

where Z is called the **partition function**. The probability of finding the system in any particular state S is therefore

$$\mathcal{P}(S) = \frac{1}{Z}\mathrm{e}^{-E(S)/(k_B T)}. \tag{2.14}$$

The partition function is found using the fact that the sum of probabilities for all possible states is equal to 1:

$$Z = \sum_S \mathrm{e}^{-E(S)/(k_B T)}. \tag{2.15}$$

[3]

2.2.3.2 The Equipartition Theorem

The energy of a molecule can be divided into different sub-types of energy, such as kinetic energy, rotational energy and elastic potential energy through spring-like motion in the molecular bonds. These energy types are called the molecule's **degrees of freedom** (DOF), and all the given examples are specifically *quadratic* degrees of freedom since the formulas can be written on the form

$$E(q) = C q_\alpha^2, \tag{2.16}$$

where C is some constant coefficient and q_α is a coordinate or momentum component (linear or angular). The equipartition theorem states that the average energy of such a degree of freedom is $\frac{1}{2}k_B T$. The total thermal energy of a system is then simply

$$U_{\text{thermal}} = \mathcal{N} \cdot n_{\text{DOF}} \cdot \frac{1}{2} k_B T, \tag{2.17a}$$

where $\mathcal{N}$ is the number of molecules and n_{DOF} is the number of DOF. Using notation with the specific gas constant ($mR = \mathcal{N}k_B$) instead, the same equation can be written in terms of the specific internal energy

$$e := \frac{U_{\text{thermal}}}{m} = n_{\text{DOF}} \cdot \frac{1}{2} R T. \tag{2.17b}$$

Due to quantum mechanical reasons, rotational DOF along axes of rotational symmetry do not count. Hence, a gas consisting of diatomic molecules, for instance, has only two rotational DOF. A monoatomic gas is even simpler with only the three translational DOF. The theorem can easily be proven using Boltzmann factors and the partition function Z from the previous section. [3]

2.2.3.3 Kinetic Theory

The kinetic theory of gases is a fluid description at the mesoscopic scale, as defined in Section 2.2.1. It describes the distribution of particles in a gas, which evolves at the timescales around the mean collision time t_{mfp} (the average time between collisions). For simplicity, the discussion here will be limited to *dilute monoatomic* gases. The dilute part is equivalent to the assumption that only two molecules collide at a time, since the actual collision time is much smaller than the time between collisions: $t_{\text{c}} \ll t_{\text{mfp}}$. The monoatomic part is to ensure that all collisions are elastic, since the only DOF are the translational ones, as described previously. This choice is further motivated since the macroscopic behaviour of polyatomic and monoatomic gases are largely similar. [1]

2.2.3.4 The Distribution Function

At the heart of the kinetic theory lies the **particle distribution function** $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$. It is to some extent a more general form of density ρ, which also incorporates the particle velocity $\boldsymbol{\xi}$. Thus, the particle distribution function, or **pdf**, lies in the 7-dimensional *space-velocity-time* space and has the units $[\text{kgs}^3\text{m}^{-6}]$. Simply put, it represents the density of particles with velocity $\boldsymbol{\xi}$ at position $\boldsymbol{x}$ and time t. However, in similarity with a probability density function, which coincidentally shares the same abbreviation, the continuous PDF hold little meaning by itself and its true potential is realised only when integrated. It then represent the mass of the particles in the integrated *ranges* of space and velocity at time t. [1]

2.2.3.5 The Moments of the PDF

Integration of the PDF multiplied with certain weighted functions of $\boldsymbol{\xi}$ over the entire velocity space are called its **moments** and serves as the connection to the macroscopic variables such as density ρ and fluid velocity $\boldsymbol{u}$. The simplest moment is the **mass density**

$$\rho(\boldsymbol{x}, t) = \int f(\boldsymbol{x}, \boldsymbol{\xi}, t)\, \mathrm{d}^3\xi. \tag{2.18a}$$

Next, the **momentum density** can be found as the moment

$$\rho(\boldsymbol{x}, t)\boldsymbol{u}(\boldsymbol{x}, t) = \int \boldsymbol{\xi} f(\boldsymbol{x}, \boldsymbol{\xi}, t)\, \mathrm{d}^3\xi, \tag{2.18b}$$

where the mean (fluid) velocity can be obtained simply by dividing with the first moment. The **total energy density** moment has the form

$$\rho(\boldsymbol{x}, t)E(\boldsymbol{x}, t) = \frac{1}{2}\int |\boldsymbol{\xi}|^2 f(\boldsymbol{x}, \boldsymbol{\xi}, t)\, \mathrm{d}^3\xi. \tag{2.18c}$$

This includes both the kinetic energy $\frac{1}{2}\rho|\boldsymbol{u}|^2$ due to the bulk motion and the internal energy due to random thermal fluctuations. The **internal energy density** can be separated out as a moment

$$\rho(\boldsymbol{x},t)e(\boldsymbol{x},t) = \frac{1}{2}\int |\boldsymbol{v}|^2 f(\boldsymbol{x},\boldsymbol{\xi},t)\,\mathrm{d}^3\xi, \tag{2.18d}$$

with the **relative velocity**

$$\boldsymbol{v} = \boldsymbol{\xi} - \boldsymbol{u}. \tag{2.19}$$

Finally, to find a moment for the pressure or temperature, the internal energy density moment can be inserted into Eq. (2.17b) using the equipartition theorem for a monoatomic gas ($n_{\mathrm{DOF}} = 3$) and combined with the ideal gas law in Eq. (2.1):

$$\rho e = \frac{3}{2}\rho RT = \frac{3}{2}p. \tag{2.20}$$

Now, Eq. (2.20) is exactly equal to Eq. (2.18d) and as a result, both the pressure and temperature is related to the moment of internal energy as

$$p = \rho RT = \frac{1}{3}\int |\boldsymbol{v}|^2 f(\boldsymbol{x},\boldsymbol{\xi},t)\,\mathrm{d}^3\xi. \tag{2.21}$$

[1]

2.2.3.6 The Equilibrium Distribution

Given enough time, the PDF will eventually reach a state of thermal equilibrium where it is isotropic in velocity space around the fluid velocity: $\boldsymbol{\xi} = \boldsymbol{u}$. The physical reasoning is that due to the sensitivity in how the alignment during the particle collisions affect the resulting post-collision velocity directions, the distribution will even out in all directions after sufficiently many collisions. This resulting **equilibrium distribution function** $f^{\mathrm{eq}}(\boldsymbol{x},\boldsymbol{\xi},t)$ may therefore, after changing to the relative velocity frame as defined in Eq. (2.19), be expressed in its isotropic form $f^{\mathrm{eq}}(\boldsymbol{x},|\boldsymbol{v}|,t)$. [1]

2.2.3.7 The Maxwell Speed Distribution

To find a formula for $f^{\mathrm{eq}}(\boldsymbol{x},|\boldsymbol{v}|,t)$, the first step is to apply the reservoir concept from Section 2.2.3.1. At equilibrium, some arbitrary molecule can be chosen as the system of interest while all other neighbouring molecules serve as the reservoir. Then, a Boltzmann factor can be used to evaluate the probability of finding the molecule at a specific *speed* $|\boldsymbol{v}|$. The energy is the kinetic energy $\frac{1}{2}m|\boldsymbol{v}|^2$ and the available states are all speeds from zero to infinity. Since the state space is continuous, Eq. (2.14) cannot be used directly, and the Boltzmann factor will instead be part of a probability density function with some constant of proportionality C:

$$\mathcal{D}(|\boldsymbol{v}|) = C\mathrm{e}^{-m|\boldsymbol{v}|^2/(2k_BT)} = C\mathrm{e}^{-|\boldsymbol{v}|^2/(2RT)}. \tag{2.22}$$

The last step rewrites the Boltzmann factor in terms of R, noting that the number of particles $\mathcal{N}$ is just one. To find C, the same principle as finding the constant in

Eq. (2.14) could be used, namely noting that the probability of finding the molecule at *any* speed is equal to one:

$$1 = \int \mathcal{D}(|\boldsymbol{v}|)\,\mathrm{d}^d\xi. \tag{2.23}$$

To allow for LBM schemes at different number of spatial dimensions d, the integration is kept general. The next step is to note that the integrand is spherically symmetrical and that the differential element can be substituted as

$$\mathrm{d}^d\xi = S_{d-1}|\boldsymbol{v}|^{d-1}\,\mathrm{d}|\boldsymbol{v}|, \tag{2.24}$$

where S_{d-1} is the $(d-1)$-dimensional surface area of the d-dimensional unit sphere, allowing the integration to be performed in one dimension with the new limits $0 \leq |\boldsymbol{v}| < \infty$. S_{d-1} can be found by the general formula [4]

$$S_{d-1} = \frac{2\pi^{d/2}}{\Gamma(\frac{d}{2})}, \tag{2.25}$$

where $\Gamma(z)$ is the gamma function. Then, Eq. (2.23) is transformed into

$$1 = S_{d-1}C\int_0^\infty \mathrm{e}^{-|\boldsymbol{v}|^2/(2RT)}|\boldsymbol{v}|^{d-1}\,\mathrm{d}|\boldsymbol{v}|. \tag{2.26}$$

One last variable change of $x = |\boldsymbol{v}|/\sqrt{2RT}$ yields

$$1 = S_{d-1}C(2RT)^{d/2}\int_0^\infty x^{d-1}\mathrm{e}^{-x^2}\,\mathrm{d}x. \tag{2.27}$$

The definite integral in Eq. (2.27) can be found in a formula collection [5]:

$$\int_0^\infty x^{d-1}\mathrm{e}^{-x^2}\,\mathrm{d}x = \frac{1}{2}\Gamma\left(\tfrac{d}{2}\right). \tag{2.28}$$

Inserting this back into Eq. (2.27) along with Eq. (2.25), noting that the gamma functions cancel, finally results in the expression

$$C = \frac{1}{(2\pi RT)^{d/2}}. \tag{2.29}$$

Consequently, the **Maxwell speed distribution** has the form

$$\mathcal{D}(|\boldsymbol{v}|) = \frac{1}{(2\pi RT)^{d/2}}\mathrm{e}^{-|\boldsymbol{v}|^2/(2RT)}. \tag{2.30}$$

To put this into terms of a particle distribution function, $\mathcal{D}(|\boldsymbol{v}|)$ is simply multiplied by the macroscopic density ρ to ensure that the moments in Eqs. (2.18) are correctly obtained. Thus, the **equilibrium distribution function** in d dimensions is expressed in terms of macroscopic variables as

$$f^{\mathrm{eq}}(\boldsymbol{x}, |\boldsymbol{v}|, t) = \frac{\rho}{(2\pi RT)^{d/2}}\mathrm{e}^{-|\boldsymbol{v}|^2/(2RT)}. \tag{2.31}$$

[1], [3]

2.2.3.8 The Boltzmann Equation

The last few sections introduced the particle distribution function and its form at equilibrium, but the question still remains how it evolves in time. The total time derivative is expressed in terms of the partial derivatives using the chain rule:

$$\frac{\mathrm{d}f(\boldsymbol{x},\boldsymbol{\xi},t)}{\mathrm{d}t} = \frac{\partial f}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}t} + \frac{\partial f}{\partial x_\alpha}\frac{\mathrm{d}x_\alpha}{\mathrm{d}t} + \frac{\partial f}{\partial \xi_\alpha}\frac{\mathrm{d}\xi_\alpha}{\mathrm{d}t}. \tag{2.32}$$

In the first term $\frac{\mathrm{d}t}{\mathrm{d}t} = 1$ and, by definition, $\frac{\mathrm{d}x_\alpha}{\mathrm{d}t} = \xi_\alpha$. For the last term, Newton's second law of motion gives the specific body force $\frac{\mathrm{d}\xi_\alpha}{\mathrm{d}t} = \frac{F_\alpha}{\rho}$, where the body force $\boldsymbol{F}$ has the units [N/m^3]. The resulting equation is the **Boltzmann equation**

$$\frac{\partial f}{\partial t} + \xi_\alpha\frac{\partial f}{\partial x_\alpha} + \frac{F_\alpha}{\rho}\frac{\partial f}{\partial \xi_\alpha} = \Omega(f), \tag{2.33}$$

where the total differential has been rewritten using the common notation of the **collision operator** $\Omega(f) = \frac{\mathrm{d}f}{\mathrm{d}t}$. Eq. (2.33) can be interpreted as an advection equation, where the first two terms represent advection of f with the particle velocity $\boldsymbol{\xi}$, the third term representing forces affecting this velocity and $\Omega(f)$ acting as a source term which locally redistributes f due to collisions, hence its name. [1]

2.2.3.9 The BGK Collision Operator

The nature of Eq. (2.33) might seem deceptively simple at first sight, but primarily due to the fact that Boltzmann's original expression for $\Omega(f)$ is a complicated integral over velocity space, it is in general classified as a first order *non-linear integro*-partial differential equation (PDE). Consequently, most LBM schemes are based on simplified versions of the collision operator. Any useful version of the collision operator must fulfil these two conditions:

1. The PDF locally evolves towards the equilibrium distribution f^{eq}.
2. The collisions conserve mass, momentum and energy.

In the case of the considered monoatomic gas, the stricter condition of conserved translational energy can be applied due to elastic collisions. The conservation constraints can be expressed in terms of moments of the collision operator in a similar manner to the moments of f in Eqs. (2.18):

$$\text{Mass conservation:} \qquad \int \Omega(f)\,\mathrm{d}^3\xi = 0, \tag{2.34a}$$

$$\text{Momentum conservation:} \qquad \int \boldsymbol{\xi}\Omega(f)\,\mathrm{d}^3\xi = \mathbf{0}, \tag{2.34b}$$

$$\text{Energy conservation:} \qquad \int |\boldsymbol{\xi}|^2\Omega(f)\,\mathrm{d}^3\xi = 0, \tag{2.34c}$$

$$\text{Internal energy conservation:} \qquad \int |\boldsymbol{v}|^2\Omega(f)\,\mathrm{d}^3\xi = 0. \tag{2.34d}$$

The simplest collision operator is the commonly used **Bhatnagar-Gross-Krook (BGK) collision operator**

$$\Omega(f) = -\frac{1}{\tau}(f - f^{\mathrm{eq}}), \tag{2.35}$$

which directly captures the evolution of f towards equilibrium through the relaxation time parameter τ. It is also easy to see that it satisfies Eqs. (2.34), since both f and f^{eq} has the same moments. [1]

2.2.3.10 Connection to Macroscopic Fluid Equations

Taking moments of the Boltzmann equation (Eq. (2.33)) by multiplying it with $\boldsymbol{\xi}$ to a particular order and integrating it over velocity space yields macroscopic conservation equations. This is linked to how the moments of f results in expressions in macroscopic quantities. The zeroth order moment directly gives the usual continuity equation. The first and second order moments can be used to derive the momentum and energy equations, respectively. These equations depend on the stress tensor $\sigma_{\alpha\beta}$, which in turn depends on f. Letting $f = f^{\text{eq}}$ turns out to give the Euler momentum equation and a simplified 'Euler' energy equation, showing that viscous dissipation and heat conduction is connected to *non-equilibrium* (i.e. the deviation $f^{\text{neq}} = f - f^{\text{eq}}$), since these phenomena are not present in the Euler equations. To further analyse the macroscopic equations, the *Chapman-Enskog analysis* can be used, which expresses the PDF as a perturbation expansion around equilibrium:

$$f = f^{\text{eq}} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + \dots. \tag{2.36}$$

The *smallness parameter* ϵ relates each term to its order in the *Knudsen number*

$$\text{Kn} = \frac{\ell_{\text{mfp}}}{\ell}, \tag{2.37}$$

that compares the mean free path to a representative length scale. As already mentioned, expanding to the zeroth order leads to the Euler equations, while expanding to first order gives the equations of the Navier-Stokes-Fourier model. Going further at second order leads to the so-called *Burnett equations*, which actually predicts ultrasonic sound propagation with even better agreement at high Knudsen numbers. At even higher expansions, the agreement instead turns out to be worse, and the expansion actually diverges. [1]

2.3 The Lattice-Boltzmann Method

In this section, the theory of the previous section is put together into the derivation of the standard LBM. At first the Boltzmann equation is discretised. Then the LBM algorithm is described followed by a discussion on boundary conditions.

2.3.1 Discretisation in Velocity Space

The main principle of the Lattice-Boltzmann method is to solve the **Boltzmann equation** (2.33) for the evolution of the **particle distribution function** f. Since f lives in the 7-dimensional space-velocity-time space, it would be rather computationally inefficient to discretise all of the dimensions equally. Instead, it turns out to be enough to discretise the velocity space with a small number of velocities while still maintaining the correct macroscopic behaviour. The idea is to perform

the discretisation in such a way that the correct macroscopic moments are obtained through *finite sums* instead of *continuous integration* over the *entire* velocity space. [1]

2.3.1.1 Non-dimensionalisation

Before beginning with the velocity discretisation, the governing equations will be *non-dimensionalised* to simplify the subsequent steps. As this is a very common procedure in the field of fluid mechanics, the process will not be described in too much detail.

In accordance with Buckingham's Π-theorem, a mechanical system can be non-dimensionalised in terms of a characteristic length ℓ, velocity $\mathcal{U}$ and density ρ_0 [1], [6]. Using the non-dimensionalised quantities

$$\begin{aligned} t^\star &= \frac{\mathcal{U}}{\ell}t, \qquad \boldsymbol{x}^\star = \frac{1}{\ell}\boldsymbol{x}, \quad \boldsymbol{\xi}^\star = \frac{1}{\mathcal{U}}\boldsymbol{\xi}, \qquad f^\star = \frac{\mathcal{U}^d}{\rho_0}f, \\ \boldsymbol{F}^\star &= \frac{\ell}{\rho_0\mathcal{U}^2}\boldsymbol{F}, \quad \rho^\star = \frac{1}{\rho_0}\rho, \quad \Omega^\star = \frac{\ell\mathcal{U}^{d-1}}{\rho_0}\Omega, \quad \theta^\star = \frac{R}{\mathcal{U}^2}T, \end{aligned} \tag{2.38}$$

the **non-dimensional continuous Boltzmann equation** reads

$$\frac{\partial f^\star}{\partial t^\star} + \xi^\star_\alpha \frac{\partial f^\star}{\partial x^\star_\alpha} + \frac{F^\star_\alpha}{\rho^\star}\frac{\partial f^\star}{\partial \xi^\star_\alpha} = \Omega^\star(f^\star), \tag{2.39}$$

with the **non-dimensional equilibrium distribution function**

$$f^{\mathrm{eq}\star}(\rho^\star, \boldsymbol{u}^\star, \theta^\star, \boldsymbol{\xi}^\star) = \frac{\rho^\star}{(2\pi\theta^\star)^{d/2}}\mathrm{e}^{-(\boldsymbol{\xi}^\star - \boldsymbol{u}^\star)^2/(2\theta^\star)}. \tag{2.40}$$

From this point onward, the $\star$ notation will be dropped for compactness and all further references to the Boltzmann equation implies the non-dimensional version in Eq. (2.39) unless explicitly stated otherwise. [1]

2.3.1.2 Hermite Polynomials

The method of discretisation is by expansion in **Hermite polynomials** (**HP**s), which is a type of polynomial with useful integral discretisation properties. They are constructed using a **generating function** which, extended to d dimensions, reads

$$\omega(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}}\mathrm{e}^{-\boldsymbol{x}^2/2}. \tag{2.41}$$

Notably, it has the same form as the equilibrium distribution function, which consequently can be written as

$$f^{\mathrm{eq}}(\rho, \boldsymbol{u}, \theta, \boldsymbol{\xi}) = \frac{\rho}{(2\pi\theta)^{d/2}}\mathrm{e}^{-(\boldsymbol{\xi}-\boldsymbol{u})^2/(2\theta)} = \frac{\rho}{\theta^{d/2}}\omega\left(\frac{\boldsymbol{\xi}-\boldsymbol{u}}{\sqrt{\theta}}\right). \tag{2.42}$$

The d-dimensional HP of n-th order is found using the formula

$$\boldsymbol{H}^{(n)}(\boldsymbol{x}) = (-1)^n \frac{1}{\omega(\boldsymbol{x})}\boldsymbol{\nabla}^{(n)}\omega(\boldsymbol{x}), \tag{2.43}$$

where $\boldsymbol{H}^{(n)}$ and $\boldsymbol{\nabla}^{(n)}$ are tensors of rank n with d^n components. The components of $\boldsymbol{\nabla}^{(n)}$ consists of the n consecutive spatial derivatives

$$\nabla^{(n)}_{\alpha_1 \dots \alpha_n} = \frac{\partial}{\partial x_{\alpha_1}} \cdots \frac{\partial}{\partial x_{\alpha_n}}. \tag{2.44}$$

One of the useful properties of the HPs is that they form a complete basis in $\mathbb{R}$. Thus, a sufficiently well-behaved function $f(\boldsymbol{x}) \in \mathbb{R}$ may be written as the series

$$f(\boldsymbol{x}) = \omega(\boldsymbol{x}) \sum_{n=0}^{\infty} \frac{1}{n!} \boldsymbol{a}^{(n)} \cdot \boldsymbol{H}^{(n)}(\boldsymbol{x}), \quad \boldsymbol{a}^{(n)} = \int f(\boldsymbol{x}) \boldsymbol{H}^{(n)}(\boldsymbol{x}) \, \mathrm{d}^d x. \tag{2.45}$$

Here, the dot product between the tensors $\boldsymbol{a}^{(n)}$ and $\boldsymbol{H}^{(n)}$ is defined as the full contraction $a^{(n)}_{\alpha_1 \dots \alpha_n} H^{(n)}_{\alpha_1 \dots \alpha_n}$, where all indices $\alpha_1 \dots \alpha_n$ are summed over the d dimensions in accordance with the Einstein summation convention. [1]

2.3.1.3 HP Series Expansion of the Equilibrium Distribution

Applying Eq. (2.45) to f^{eq} while using the expression in Eq. (2.42) for the coefficients $\boldsymbol{a}^{(n),\mathrm{eq}}$ results in

$$\begin{aligned} f^{\mathrm{eq}}(\rho, \boldsymbol{u}, \theta, \boldsymbol{\xi}) &= \omega(\boldsymbol{\xi}) \sum_{n=0}^{\infty} \frac{1}{n!} \boldsymbol{a}^{(n),\mathrm{eq}}(\rho, \boldsymbol{u}, \theta) \cdot \boldsymbol{H}^{(n)}(\boldsymbol{\xi}), \\ \boldsymbol{a}^{(n),\mathrm{eq}}(\rho, \boldsymbol{u}, \theta) &= \frac{\rho}{\theta^{d/2}} \int \omega \left(\frac{\boldsymbol{\xi} - \boldsymbol{u}}{\sqrt{\theta}} \right) \boldsymbol{H}^{(n)}(\boldsymbol{\xi}) \, \mathrm{d}^d \xi. \end{aligned} \tag{2.46}$$

Using the substitution $\boldsymbol{\eta} = (\boldsymbol{\xi} - \boldsymbol{u})/\sqrt{\theta}$ in the coefficient integral:

$$\boldsymbol{a}^{(n),\mathrm{eq}} = \rho \int \omega(\boldsymbol{\eta}) \boldsymbol{H}^{(n)} \left(\sqrt{\theta} \boldsymbol{\eta} + \boldsymbol{u} \right) \mathrm{d}^d \eta. \tag{2.47}$$

Computing this integral up to second order ($n = 0, 1, 2$), show that the coefficients $\boldsymbol{a}^{(n),\mathrm{eq}}$ are closely related to the macroscopic moments of f:

$$\begin{aligned} a^{(0),\mathrm{eq}} &= \rho = \int f \, \mathrm{d}^d \xi, \\ a^{(1),\mathrm{eq}}_{\alpha} &= \rho u_\alpha = \int \xi_\alpha f \, \mathrm{d}^d \xi, \\ a^{(2),\mathrm{eq}}_{\alpha\beta} &= \rho \Big(u_\alpha u_\beta + (\theta - 1) \delta_{\alpha\beta} \Big) \implies \\ \frac{a^{(2),\mathrm{eq}}_{\alpha\alpha} + \rho d}{2} &= \frac{1}{2} \rho |\boldsymbol{u}|^2 + \frac{d}{2} \rho \theta = \rho E = \frac{1}{2} \int |\boldsymbol{\xi}|^2 f \, \mathrm{d}^d \xi, \end{aligned} \tag{2.48}$$

where $\delta_{\alpha\beta}$ is the Kronecker delta. It also turns out that the coefficients $\boldsymbol{a}^{(n)}$ for the HP expansion of the regular PDF relate to the moments in the same way as in Eq. (2.48). The implication is that only the first *three* terms of the HP expansion are required to satisfy the conservation laws and recover the correct macroscopic behaviour. Using this approximation, the truncated HP expansion $\hat{f}^{\mathrm{eq}}$ reads

$$\begin{aligned} \hat{f}^{\mathrm{eq}}(\rho, \boldsymbol{u}, \theta, \boldsymbol{\xi}) = \omega(\boldsymbol{\xi}) \rho \Big(1 + \xi_\alpha u_\alpha + \frac{1}{2} \Big(u_\alpha u_\beta + (\theta - 1) \delta_{\alpha\beta} \Big) \Big(\xi_\alpha \xi_\beta - \delta_{\alpha\beta} \Big) \Big) \\ := \omega(\boldsymbol{\xi}) \rho Q(\boldsymbol{u}, \theta, \boldsymbol{\xi}). \end{aligned} \tag{2.49}$$

Once again, the $\hat{f}^{\mathrm{eq}}$ notation is dropped for brevity and all further references to f^{eq} implies the truncated expansion. The full derivation from Eq. (2.46) to Eq. (2.49) is shown in Appendix A.1.1. Another possible path that leads to the exact same expression is to perform a Mach number expansion to second order (although at higher order these two methods stop coinciding). The error at this stage is $\mathcal{O}(u^3)$. [1]

2.3.1.4 Discretisation of the Equilibrium Distribution

The technique that allows for the discretisation of the continuous equilibrium distribution to a finite set of velocities $\{\boldsymbol{\xi}_i\}$ is the **Gauss-Hermite quadrature rule** generalised to d dimensions:

$$\int \omega(\boldsymbol{x}) P^{(N)}(\boldsymbol{x}) \,\mathrm{d}^d x = \sum_{i=0}^{n^d-1} w_i P^{(N)}(\boldsymbol{x}_i), \tag{2.50}$$

where $P^{(N)}(\boldsymbol{x})$ is a multi-dimensional polynomial of N-th order and the so-called *abscissae* $\boldsymbol{x}_i$ are a finite number ($q = n^d$) of points where each component is a root of the 1-dimensional Hermite polynomial $H^{(n)}(x_{i\alpha}) = 0$. The associated weights w_i are the key that permits the *continuous* integral of the polynomial to be evaluated exactly using only the *finite* number of abscissae. n is subject to the condition

$$n \geq \frac{N+1}{2}. \tag{2.51}$$

The most straightforward way of using the quadrature rule in this case, is by first inserting the truncated HP expansion from Eq. (2.49) into the coefficient integral in Eq. (2.45):

$$\boldsymbol{a}^{(n),\mathrm{eq}} = \int f^{\mathrm{eq}}(\boldsymbol{\xi}) \boldsymbol{H}^{(n)}(\boldsymbol{\xi}) \,\mathrm{d}^d \xi = \rho \int \omega(\boldsymbol{\xi}) Q(\boldsymbol{\xi}) \boldsymbol{H}^{(n)}(\boldsymbol{\xi}) \,\mathrm{d}^d \xi. \tag{2.52}$$

Then, the Gauss-Hermite quadrature rule is applied to the composed polynomial $\boldsymbol{R}(\boldsymbol{\xi}) = Q(\boldsymbol{\xi}) \boldsymbol{H}^{(n)}(\boldsymbol{\xi})$:

$$\boldsymbol{a}^{(n),\mathrm{eq}} = \rho \int \omega(\boldsymbol{\xi}) \boldsymbol{R}(\boldsymbol{\xi}) \,\mathrm{d}^d \xi = \rho \sum_i w_i \boldsymbol{R}(\boldsymbol{\xi}_i) = \rho \sum_i w_i Q(\boldsymbol{\xi}_i) \boldsymbol{H}^{(n)}(\boldsymbol{\xi}_i). \tag{2.53}$$

This leads to the discrete set of q equilibrium distributions $\{f_i^{\mathrm{eq}}\}$, where

$$f_i^{\mathrm{eq}} := \frac{w_i}{\omega(\boldsymbol{\xi}_i)} f^{\mathrm{eq}}(\boldsymbol{\xi}_i) = w_i \rho \left(1 + \xi_{i\alpha} u_\alpha + \frac{1}{2}\Big(u_\alpha u_\beta + (\theta - 1)\delta_{\alpha\beta}\Big)\Big(\xi_{i\alpha}\xi_{i\beta} - \delta_{\alpha\beta}\Big)\right). \tag{2.54}$$

Crucially, this set satisfies the same conservation of the first three moments of f (see Eq. (2.48)) relating to mass, momentum and energy as the continuous distribution $f^{\mathrm{eq}}(\boldsymbol{\xi})$. To further simplify Eq. (2.54) and to keep the model consistent with the derived equation of state in Eq. (2.7), the *isothermal* assumption is used. This results in a globally constant speed of sound $c_s = \sqrt{RT}$ (c.f. Eqs. (2.1) and (2.7)). Selecting the characteristic velocity scale $\mathcal{U} = c_s$ nets a constant $\theta = 1$ cancelling

the term $(\theta - 1)\delta_{\alpha\beta}$ from Eq. (2.54). Finally, since many of the abscissae $\boldsymbol{\xi}_i$ turns out to contain factors of $\sqrt{3}$, the new particle velocity

$$\boldsymbol{c}_i = \frac{\boldsymbol{\xi}_i}{\sqrt{3}} \tag{2.55}$$

is introduced. Consequently, also the fluid velocity $\boldsymbol{u}$ is rescaled in the same manner. As the non-dimensional $|\boldsymbol{\xi}^\star|$ is now scaled by the physical speed of sound, it is essentially a particle Mach number. By multiplying a Mach number with the speed of sound, the velocity is regained. The rescaling factor may therefore be interpreted as the (non-dimensional) **lattice speed of sound**

$$c_s = \frac{1}{\sqrt{3}}. \tag{2.56}$$

Thus, the **velocity-discretised equilibrium distribution** reads

$$f_i^{\mathrm{eq}} = w_i \rho \left(1 + \frac{c_{i\alpha} u_\alpha}{c_s^2} + \frac{u_\alpha u_\beta (c_{i\alpha} c_{i\beta} - c_s^2 \delta_{\alpha\beta})}{2 c_s^4}\right). \tag{2.57}$$

Note, that the function is still continuous in space and time through the macroscopic variables $\rho(\boldsymbol{x}, t)$ and $\boldsymbol{u}(\boldsymbol{x}, t)$. [1]

2.3.1.5 Discretisation of the PDF

To ensure that the PDF also conserves the macroscopic moments, the discretisation is performed similarly to f^{eq}. By multiplying and diving by a factor $\omega(\boldsymbol{c})$, the Gauss-Hermite quadrature rule can be applied to the coefficient in Eq. (2.45) as

$$\begin{aligned} \boldsymbol{a}^{(n)}(\boldsymbol{x}, t) &= \int f(\boldsymbol{x}, \boldsymbol{c}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}) \, \mathrm{d}^d c = \int \frac{\omega(\boldsymbol{c})}{\omega(\boldsymbol{c})} f(\boldsymbol{x}, \boldsymbol{c}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}) \, \mathrm{d}^d c \\ &\approx \sum_i \frac{w_i}{\omega(\boldsymbol{c}_i)} f(\boldsymbol{x}, \boldsymbol{c}_i, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i) = \sum_i f_i(\boldsymbol{x}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i), \end{aligned} \tag{2.58}$$

with the **velocity-discretised particle distribution function**

$$f_i(\boldsymbol{x}, t) = \frac{w_i}{\omega(\boldsymbol{c}_i)} f(\boldsymbol{x}, \boldsymbol{c}_i, t). \tag{2.59}$$

A more in depth explanation of Eq. (2.58) showing that the error is $\mathcal{O}(u^4)$ is presented in Appendix A.2.

At this point, it is possible to write down the **force-free discrete-velocity Boltzmann equation**

$$\partial_t f_i + c_{i\alpha} \partial_\alpha f_i = \Omega(f_i), \quad i = 0, \ldots, q-1. \tag{2.60}$$

The discretisation has transformed the moments in Eqs. (2.18a) & (2.18b) into the sums

$$\rho = \sum_i f_i = \sum_i f_i^{\mathrm{eq}}, \tag{2.61a}$$

$$\rho \boldsymbol{u} = \sum_i \boldsymbol{c}_i f_i = \sum_i \boldsymbol{c}_i f_i^{\mathrm{eq}}. \tag{2.61b}$$

Performing the Chapman-Enskog analysis (see Section 2.2.3.10) on Eq. (2.60) with the BGK collision operator and truncated f_i^{eq}, reveals that the second order moment of the first order perturbation $f^{(1)}$ takes the form

$$\sum_i c_{i\alpha}c_{i\beta}f_i^{(1)} = -\rho c_s^2\tau\left(\partial_\beta^{(1)}u_\alpha + \partial_\alpha^{(1)}u_\beta\right) + \tau\partial_\gamma^{(1)}(\rho u_\alpha u_\beta u_\gamma), \tag{2.62}$$

where $\partial_\alpha^{(1)}$ refers to a Kn expanded derivative

$$c_{i\alpha}\partial_\alpha f_i = \left(\epsilon c_{i\alpha}\partial_\alpha^{(1)}\right) f_i. \tag{2.63}$$

In Eq. (2.62), the first term corresponds to a Navier-Stokes viscous stress tensor, while the second is an error term originating from the $\mathcal{O}(u^3)$ error of the truncated equilibrium distribution. Comparing the magnitude of these terms shows that the error term may be neglected if $u^2 \ll c_s^2$, which means that the LBM based on the second order truncated f^{eq} is only suitable for $\text{Ma}^2 \ll 1$. [1]

2.3.1.6 Discretisation of the Force Term

The force term in the Boltzmann equation (2.33) may be written as

$$\frac{\boldsymbol{F}}{\rho}\cdot\boldsymbol{\nabla}_{\boldsymbol{\xi}}f. \tag{2.64}$$

Instead of discretising $\boldsymbol{F}$ separately, the entire force term in Eq. (2.64) will be discretised at once. Inserting the HP generation formula (2.43) directly into the HP series expansion in Eq. (2.45) allow for the formulation of f as

$$f(\boldsymbol{\xi}) = \sum_{n=0}^{\infty}\frac{(-1)^n}{n!}\boldsymbol{a}^{(n)}\cdot\boldsymbol{\nabla}_{\boldsymbol{\xi}}^{(n)}\omega(\boldsymbol{\xi}). \tag{2.65}$$

Putting this back into Eq. (2.64) yields

$$\begin{aligned}\frac{\boldsymbol{F}}{\rho}\cdot\boldsymbol{\nabla}_{\boldsymbol{\xi}}f &= \frac{\boldsymbol{F}}{\rho}\cdot\sum_{n=0}^{\infty}\frac{(-1)^n}{n!}\boldsymbol{a}^{(n)}\cdot\boldsymbol{\nabla}_{\boldsymbol{\xi}}^{(n+1)}\omega \\ &= \frac{\boldsymbol{F}}{\rho}\cdot\sum_{n=1}^{\infty}\frac{-(-1)^n}{(n-1)!}\boldsymbol{a}^{(n-1)}\cdot\boldsymbol{\nabla}_{\boldsymbol{\xi}}^{(n)}\omega \\ &= -\frac{\boldsymbol{F}}{\rho}\cdot\omega\sum_{n=1}^{\infty}\frac{1}{(n-1)!}\boldsymbol{a}^{(n-1)}\cdot\boldsymbol{H}^{(n)}.\end{aligned} \tag{2.66}$$

The velocity-discretised forcing term is then defined similarly to f_i in Eq. (2.59) as

$$F_i = -\frac{w_i}{\omega}\frac{\boldsymbol{F}}{\rho}\cdot\boldsymbol{\nabla}_{\boldsymbol{\xi}}f = \frac{\boldsymbol{F}}{\rho}\cdot w_i\sum_{n=1}^{\infty}\frac{1}{(n-1)!}\boldsymbol{a}^{(n-1)}\cdot\boldsymbol{H}^{(n)}. \tag{2.67}$$

Truncating the series expansion at second order and rescaling with $\boldsymbol{\xi} = \boldsymbol{c}/c_s$ puts the **velocity-discretised force term** on the form

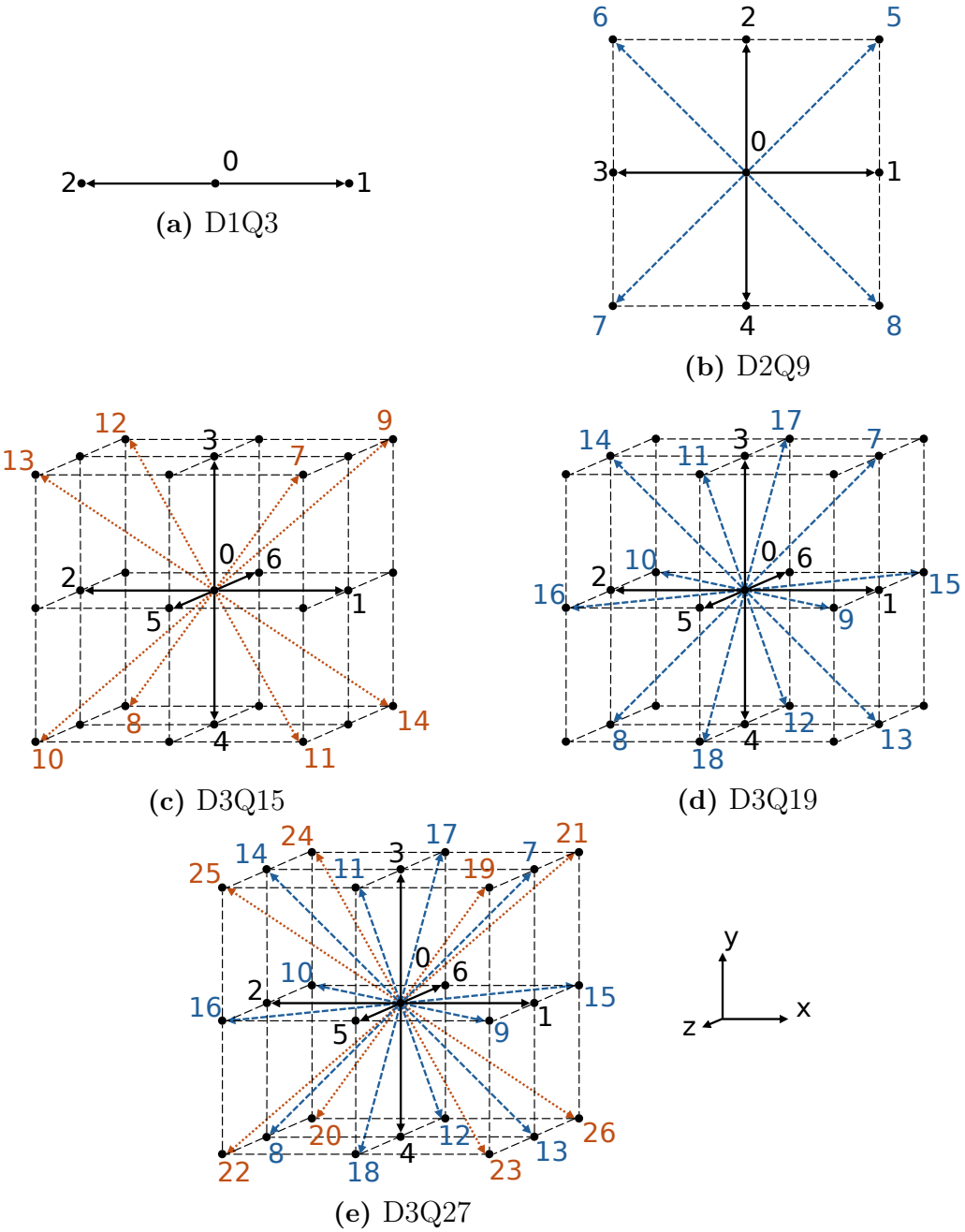
$$F_i = w_i\left(\frac{c_{i\alpha}}{c_s^2} + \frac{(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})u_\beta}{c_s^4}\right)F_\alpha. \tag{2.68}$$

With this, the force term can be put back into Eq. (2.60) to obtain the full **discrete-velocity Boltzmann equation**:

$$\partial_t f_i + c_{i\alpha}\partial_\alpha f_i = \Omega(f_i) + F_i, \quad i = 0, \ldots, q-1. \tag{2.69}$$

[1]

2.3.1.7 Velocity Sets



(a) D1Q3 (b) D2Q9

(c) D3Q15 (d) D3Q19

(e) D3Q27

Figure 2.1: Commonly used stencils in 1-3 dimensions annotated with their usual velocity numbering. The line styles represent the magnitude of the velocity vectors: $|\boldsymbol{c}_i| = 1$; ——, $|\boldsymbol{c}_i| = \sqrt{2}$; - - - -, $|\boldsymbol{c}_i| = \sqrt{3}$; ·····.

With the discretisation in velocity space completed, the only thing remaining before moving on to discretisation in time and space is to define some discrete velocity sets to be used in simulations. Due to the HP series expansion being truncated at second order, the composed polynomial $\boldsymbol{R}(\boldsymbol{\xi})$ in Eq. (2.53) is of fourth order. According to the condition in Eq. (2.51), the velocity set must therefore be comprised of at least $q = n^d = 3^d$ velocities. For 1D, 2D and 3D, this leads to the velocity sets D1Q3, D2Q9 and D3Q27, where the notation DdQq represents q velocities in d dimensions. It is also possible to construct velocity sets using 15 and 19 velocities in 3D by discarding certain velocities in D3Q27, a process called *pruning*. On the other hand, performing the HP expansion to higher order would instead necessitate larger velocity sets. [1]

Stencil	$\boldsymbol{c}_i$	$\lvert\boldsymbol{c}_i\rvert$	w_i	$\{i\}$	n
D1Q3	(0)	0	$2/3$	$\{0\}$	1
	(±1)	1	$1/6$	$\{1,2\}$	2
D2Q9	$(0,0)$	0	$4/9$	$\{0\}$	1
	$(\pm1,0),(0,\pm1)$	1	$1/9$	$\{1,2,3,4\}$	4
	$(\pm1,\pm1)$	$\sqrt{2}$	$1/36$	$\{5,6,7,8\}$	4
D3Q15	$(0,0,0)$	0	$2/9$	$\{0\}$	1
	$(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$	1	$1/9$	$\{1,\ldots,6\}$	6
	$(\pm1,\pm1,\pm1)$	$\sqrt{3}$	$1/72$	$\{7,\ldots,14\}$	8
D3Q19	$(0,0,0)$	0	$1/3$	$\{0\}$	1
	$(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$	1	$1/18$	$\{1,\ldots,6\}$	6
	$(\pm1,\pm1,0),(\pm1,0,\pm1),(0,\pm1,\pm1)$	$\sqrt{2}$	$1/36$	$\{7,\ldots,18\}$	12
D3Q27	$(0,0,0)$	0	$8/27$	$\{0\}$	1
	$(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$	1	$2/27$	$\{1,\ldots,6\}$	6
	$(\pm1,\pm1,0),(\pm1,0,\pm1),(0,\pm1,\pm1)$	$\sqrt{2}$	$1/54$	$\{7,\ldots,18\}$	12
	$(\pm1,\pm1,\pm1)$	$\sqrt{3}$	$1/216$	$\{19,\ldots,26\}$	8

Table 2.1: Commonly used stencils in 1-3 dimensions with velocities $\boldsymbol{c}_i$ of magnitude $|\boldsymbol{c}_i|$, weight w_i, numbering i and count n. An expanded version of this table can be found as Table B.1 in Appendix B.

The specific form of a *velocity set* $\{\boldsymbol{c}_i\}$ with its associated *weights* w_i will henceforth be called a **stencil** and referred to using the DdQq notation. In the following section, the Boltzmann equation will be spatially discretised into a uniform rectangular *lattice* with spacing Δx and temporally discretised using a time step Δt. It would be rather convenient if the velocities were scaled in such a way that the distributions f_i end up exactly at neighbouring lattice nodes after one time step, i.e. $c_{i\alpha} = n\Delta x/\Delta t$, where n is an integer. Furthermore, in the non-dimensionalised *lattice units*, the discretisation constants Δx and Δt are usually set to 1, leaving $c_{i\alpha} = n$. With a rest velocity $\boldsymbol{c}_0 = \boldsymbol{0}$, it is possible to construct the stencils of the aforementioned D1Q3, D2Q9, D3Q15, D3Q19 and D3Q27 using $|n| \leq 1$. Figure 2.1 visualises these stencils and Table 2.1 also includes the weights. The speed of sound $c_s = 1/\sqrt{3}$ for all of these examples. Since multiple different velocity magnitudes are used, these stencils are called **multi-speed (MS)**. It is worth mentioning that

it is possible to construct *single-speed* stencils as well; for example D2Q6, using a hexagonal lattice. Evidently, these stencils cannot have a rest velocity. [1]

2.3.2 Discretisation in Space and Time

As already mentioned in the previous section, the space and time discretisations are performed uniformly on a rectangular **lattice** (grid) with spacing Δx and time step Δt. Consequently, the particle distributions f_i at position $\boldsymbol{x}$ and time t will end up at position $\boldsymbol{x} + \boldsymbol{c}_i\Delta t$ at time $t + \Delta t$. The initial discretisation will be performed on the non-dimensional force-free discrete-velocity Boltzmann equation from Eq. (2.60), which reads

$$\partial_t f_i + c_{i\alpha}\partial_\alpha f_i = \Omega_i, \tag{2.70}$$

where $f_i(\boldsymbol{x}, t) = f(\boldsymbol{x}, \boldsymbol{c}_i, t)$ and $\Omega_i := \Omega(f_i)$ is a general collision operator, assumed to depend only on the discretised populations $\{f_i\}$ and the equilibrium distributions $\{f_i^{\mathrm{eq}}\}$. Since the equilibrium distributions are computed via the macroscopic moments of the PDF, Ω_i can be determined fully through $\{f_i\}$. This classifies Eq. (2.70) as a first-order hyperbolic PDE, which can be solved through the so-called *method of characteristics*. The method exploits the existence of trajectories, called **characteristics**, along which the PDE can be parameterised and re-expressed as an ordinary differential equation (ODE). Using the parameter ζ, which parameterises $\boldsymbol{x}$ and t according to

$$\frac{\mathrm{d}x_\alpha}{\mathrm{d}\zeta} = c_{i\alpha}, \quad \frac{\mathrm{d}t}{\mathrm{d}\zeta} = 1, \tag{2.71}$$

the left-hand side of Eq. (2.70) can be collected into a total derivative and the following ODE is obtained:

$$\frac{\partial f_i}{\partial t} + c_{i\alpha}\frac{\partial f_i}{\partial x_\alpha} = \frac{\partial f_i}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}\zeta} + \frac{\partial f_i}{\partial x_\alpha}\frac{\mathrm{d}x_\alpha}{\mathrm{d}\zeta} = \frac{\mathrm{d}f_i(\boldsymbol{x}(\zeta), t(\zeta))}{\mathrm{d}\zeta} = \Omega_i(\boldsymbol{x}(\zeta), t(\zeta)) \tag{2.72}$$

The next step is to integrate Eq. (2.72) along any characteristic over one time step Δt. The initial point is chosen as $(\boldsymbol{x}_0, t_0)$, where $\boldsymbol{x}_0 = \boldsymbol{x}(\zeta = 0)$ and $t_0 = t(\zeta = 0)$. By the condition in Eq. (2.71), $\zeta = \Delta t$ at time $t_0 + \Delta t$ and the integrated equation then reads

$$\int_0^{\Delta t} \frac{\mathrm{d}f_i}{\mathrm{d}\zeta}\,\mathrm{d}\zeta = \int_0^{\Delta t} \Omega_i\left(\boldsymbol{x}_0 + \zeta\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\zeta}, t_0 + \zeta\frac{\mathrm{d}t}{\mathrm{d}\zeta}\right)\mathrm{d}\zeta. \tag{2.73}$$

Again, using Eq. (2.71), performing the integration of the left-hand side exactly and realising that the point $(\boldsymbol{x}_0, t_0)$ is arbitrary and might as well be replaced with the more general point $(\boldsymbol{x}, t)$, the above equation is transformed into

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = \int_0^{\Delta t} \Omega_i(\boldsymbol{x} + \boldsymbol{c}_i\zeta, t + \zeta)\,\mathrm{d}\zeta. \tag{2.74}$$

Although not yet completely finished with the discretisation, it is already evident that the stated goal of having f_i end up at position $\boldsymbol{x} + \boldsymbol{c}_i\Delta t$ at time $t + \Delta t$ has been successfully achieved. The last step is to approximate the integral on the right-hand side in Eq. (2.74). This is done through the second-order accurate *trapezoidal rule*:

$$\int_0^{\Delta t} \Omega_i(\boldsymbol{x} + \boldsymbol{c}_i\zeta, t + \zeta)\,\mathrm{d}\zeta \approx \Delta t\frac{\Omega_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t)}{2} \tag{2.75}$$

Inserting this back into Eq. (2.74) would yield an implicit equation, but it can easily be transformed into an explicit equation through the simple variable transformation $f_i \rightarrow \bar{f}_i$, where

$$\bar{f}_i = f_i - \frac{\Delta t}{2}\Omega_i. \tag{2.76}$$

The resulting equation is the **Lattice-Boltzmann equation (LBE)**

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{x}, t) + \Delta t\Omega_i(\boldsymbol{x}, t), \tag{2.77}$$

where the PDF has been permanently redefined as $f_i := \bar{f}_i$, dropping the bar notation. The transformation needs no further consideration, since the collision operator Ω conserves the macroscopic moments as seen in Eqs. (2.34). Thus, the transformed PDF in Eq. (2.76) yields the same macroscopic moments as the untransformed version. [1]

2.3.2.1 The Discrete BGK Collision Operator

The **Bhatnagar-Gross-Krook (BGK) collision operator** was introduced in Section 2.2.3.9 and in its discretised form it reads

$$\Omega_i = -\frac{1}{\tau}(f_i - f_i^{\mathrm{eq}}). \tag{2.78}$$

Insertion into Eq. (2.77) yields the **Lattice-Boltzmann equation with BGK collision operator (LBGK)**

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{x}, t) - \frac{\Delta t}{\tau}\Big(f_i(\boldsymbol{x}, t) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t)\Big). \tag{2.79}$$

One major difference between the continuous force-free Boltzmann equation with the BGK operator and the discrete LBGK is that, while the PDF always evolves *towards* equilbrium in the continuous case, it might relax *directly to*, or even *past* equilibrium in the discrete case. Depending on the value of the ratio $\Delta t/\tau$ the LBGK can be divided into the cases [1]:

- **Under-relaxed** for $\Delta t/\tau > 1$, where the continuous behaviour of *exponential decay* towards equilibrium is recovered.
- **Fully relaxed** for $\Delta t/\tau = 1$, where f_i reaches f_i^{eq} *immediately* after one time step.
- **Over-relaxed** for $1/2 < \Delta t/\tau < 1$, where the PDF *oscillates* with an *exponentially decreasing* amplitude around equilibrium.
- **Unstable** for $\Delta t/\tau < 1/2$, where the oscillations *increase exponentially.*

2.3.2.2 Including the Force Term

The full discrete-velocity Boltzmann equation in Eq. (2.69) is shown here once again:

$$\partial_t f_i + c_{i\alpha}\partial_\alpha f_i = \Omega_i + F_i \tag{2.80}$$

Since the left-hand side is identical to the force-free case, its discretisation is also handled in the same way as before. The right-hand side nets an integral, now containing the sum $\Omega_i + F_i$, which to second-order accurate trapezoidal approximation reads (cf. Eq. (2.75))

$$\int_0^{\Delta t} (\Omega_i(\boldsymbol{x} + \boldsymbol{c}_i\zeta, t+\zeta) + F_i(\boldsymbol{x} + \boldsymbol{c}_i\zeta, t+\zeta))\,\mathrm{d}\zeta \approx$$
$$\Delta t \left(\frac{\Omega_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t+\Delta t)}{2} + \frac{F_i(\boldsymbol{x}, t) + F_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t+\Delta t)}{2} \right). \quad (2.81)$$

Similarly to the force-free case, this yields an implicit form of the LBE which can be transformed to an explicit version through the transformation $f_i \to \bar{f}_i$, where

$$\bar{f}_i = f_i - \frac{\Delta t}{2}(\Omega_i + F_i). \quad (2.82)$$

After some simple algebraic manipulations, the resulting LBE for $\bar{f}_i$ including forces takes the form

$$\bar{f}_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t+\Delta t) = \bar{f}_i(\boldsymbol{x}, t) + \Delta t\Big(\Omega_i(\boldsymbol{x}, t) + F_i(\boldsymbol{x}, t)\Big). \quad (2.83)$$

With the BGK collision operator this becomes the **LBGK with forcing term**:

$$\bar{f}_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t+\Delta t) = \bar{f}_i(\boldsymbol{x}, t) - \frac{\Delta t}{\bar{\tau}}\Big(\bar{f}_i(\boldsymbol{x}, t) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t)\Big) + S_i(\boldsymbol{x}, t)\Delta t, \quad (2.84)$$

where the source term $S_i = \left(1 - \frac{\Delta t}{2\bar{\tau}}\right) F_i$ and the redefined relaxation time $\bar{\tau} = \tau + \Delta t/2$. In contrast with the force-free case, the transformation of variables does effect the macroscopic moments which are now computed as (cf. Eqs. (2.85))

$$\rho = \sum_i \bar{f}_i + \frac{\Delta t}{2}\sum_i F_i, \quad (2.85a)$$

$$\rho\boldsymbol{u} = \sum_i \boldsymbol{c}_i\bar{f}_i + \frac{\Delta t}{2}\sum_i \boldsymbol{c}_i F_i. \quad (2.85b)$$

Although, the new formula for computing the density in Eq. (2.85a) is only needed if non-zero mass sources are chosen to be incorporated into F_i besides forces. The bar notation is also usually dropped even in the case with forces included. [1]

2.3.3 Streaming and Collision

When implementing LBM, the Lattice-Boltzmann equation is usually divided into the steps of streaming and collision. The first step is the **collision** step, where each of the q populations f_i are locally redistributed into f_i^* due to collisions and eventual force contributions as

$$f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) - \frac{\Delta t}{\tau}\Big(f_i(\boldsymbol{x}, t) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t)\Big) + S_i(\boldsymbol{x}, t)\Delta t. \quad (2.86a)$$

The post-collision populations f_i^* are then propagated along their characteristics $\boldsymbol{c}_i$ to the neighbouring lattice nodes in the streaming step:

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) = f_i^*(\boldsymbol{x}, t) \tag{2.86b}$$

In practice, this step is just a redistribution of data in memory. To avoid overwriting the old data before it has been propagated, it is common to create a new instance of f_i to which the data of f_i^* is copied according to the streaming instructions. Figure 2.2 shows an example of how a set of populations are streamed from a certain lattice node unto its set of neighbours. [1]

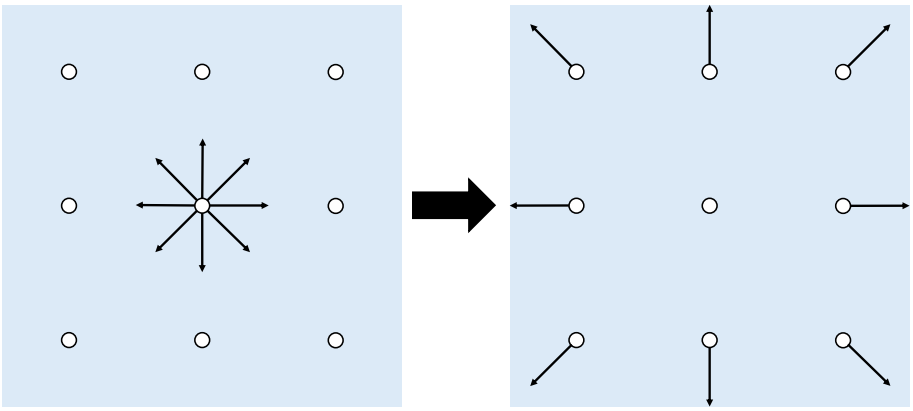


Figure 2.2: Schematic depiction of the streaming step with a D2Q9 stencil. The lattice nodes are represented by the circles "○" and the populations f_i with arrows pointing in the directions of $\boldsymbol{c}_i$.

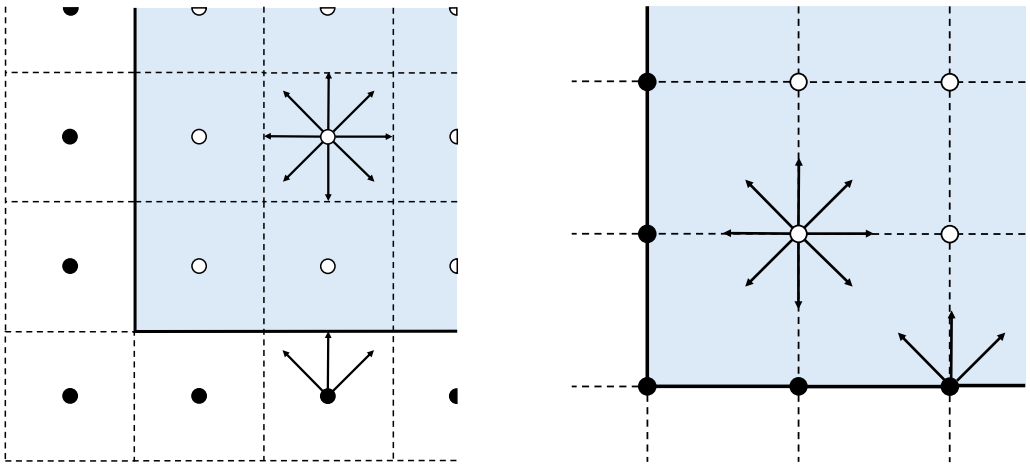
2.3.4 Boundary and Initial Conditions

Specifying boundary conditions (BCs) and initial conditions (ICs) for LBM presents a rather interesting challenge. Since the dynamics of interest are at the macroscopic scale, the desired macroscopic behaviour has to be translated to the mesoscopic scale in order to properly set the boundary and initial conditions. For steady problems, one may sometimes "cheat" by letting the simulation run for a while to dampen out undesired behaviour caused by improper ICs, or by placing inlets and outlets further away from the areas of interest to let the flow from imperfect BCs fully develop. This is, of course, basic knowledge for anyone with any experience in computational fluid dynamics (CFD), but still important to keep in mind when considering the trade-offs between more complicated yet possibly more accurate schemes and simpler ones. In this thesis, the aim is to keep it as simple as possible and evaluate when these methods break down to then determine where more complex schemes are needed.

2.3.4.1 The Perspective Choices regarding BCs

The simplest type of boundary conditions in LBM are in the family of **link-wise** BCs. For these boundary conditions, the physical boundary is located between lattice nodes which are considered to placed at the centres of the computational cells. The other big family of boundary conditions use the so-called **wet-node**

approach, with the lattice nodes located at the corners of the computational cells. Here, the boundary is instead considered at the node itself and the populations are modified directly. Figure 2.3 illustrates the difference between these two modelling approaches. Note that the physical boundary is aligned with the cell faces in both cases. [1]



(a) Link-wise **(b)** Wet node

Figure 2.3: The two main boundary approach types for LBM. The fluid region is lightly shaded and a set of internal and a set of boundary populations are included. Otherwise: fluid nodes; ○, boundary/outside nodes; ●, physical boundary; ——, computational cell face; -----.

Since simplicity is the stated goal here, the choice naturally falls on the link-wise approach. This presents yet another choice of perspective; which nodes to consider as the "boundary" nodes. Either the fluid nodes directly inside the boundary, or the nodes directly outside of the simulation domain. One advantage of choosing the latter, is that the boundary nodes are never part of the simulation itself and populations entering the fluid domain may be stored in these "ghost" nodes (depicted in Figure 2.3a) to then propagate into the domain during the streaming step. For this approach, the boundary should be treated *before* streaming, while the other approach modifies the boundary nodes *after* streaming is completed.

2.3.4.2 Bounce-Back BC

The most common BC in hydrodynamics is the no-slip condition at solid walls. In LBM, this is represented by the **Bounce-Back (BB) boundary condition**. The populations are simply bounced back in the opposite direction when a wall is encountered. This may be expressed as the modified equation for streaming

$$f_{\bar{i}}(\boldsymbol{x}_\mathrm{b}, t + \Delta t) = f_i^*(\boldsymbol{x}_\mathrm{b}, t), \tag{2.87}$$

where $\boldsymbol{x}_\mathrm{b}$ is the fluid node directly inside the boundary and $\bar{i}$ is the opposite direction of i such that $\boldsymbol{c}_{\bar{i}} = -\boldsymbol{c}_i$. The reason this BC behaves as a no-slip condition is that both the normal and tangential momentum averages to zero during the "bouncing".

Figure 2.4 depicts the simple BB schematically. The main advantages of BB are its simpleness, stability and guaranteed mass conservation, while its biggest disadvantages are its deteriorated accuracy (limited to first-order) if the boundary is not aligned with the lattice and positioned exactly halfway between the nodes and that the location of the no-slip boundary is *viscosity dependent*. [1]

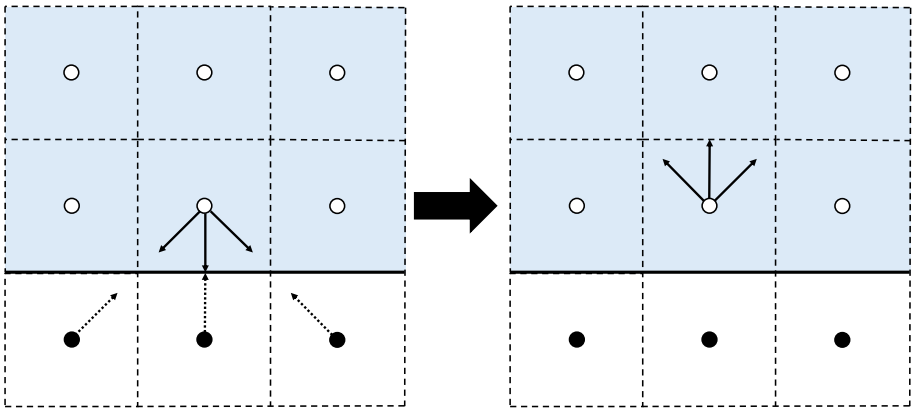


Figure 2.4: The Bounce-Back boundary condition. In the boundary nodes, opposite versions of the populations pointing back into the domain are stored and represented by the dotted line (·····) arrows. These are then streamed to the "bounced back" locations. Otherwise: Fluid nodes; ○, boundary/outside nodes; ●, physical boundary; ——, computational cell face; -----.

It is quite simple to extend the BB scheme to moving walls. By transforming to the rest frame of the wall, performing the Bounce-Back, and then transforming back yields the formula for **Dirichlet Bounce-Back boundary condition** with prescribed wall velocity $\boldsymbol{u}_\mathrm{w}$:

$$f_{\bar{i}}(\boldsymbol{x}_\mathrm{b}, t+\Delta t) = f_i^*(\boldsymbol{x}_\mathrm{b}, t) - 2w_i\rho_\mathrm{w}\frac{\boldsymbol{c}_i\cdot\boldsymbol{u}_\mathrm{w}}{c_s^2}, \tag{2.88}$$

where the subscript w denotes the values at the wall, i.e. at $\boldsymbol{x}+\frac{1}{2}\boldsymbol{c}_i\Delta t$. ρ_w may be prescribed or estimated using for example ρ_b, extrapolation or the domain average if the variation is small. The exact same formulation as in Eq. (2.88) can be used as a **velocity inlet** at an open boundary. Then, the "wall" remains stationary even though it has a non-zero normal component in $\boldsymbol{u}_\mathrm{w}$. [1]

2.3.4.3 Anti-Bounce-Back BC

The prescription of pressure (or rather density through $p=c_s^2\rho$) at an open boundary can be done through the **Anti-Bounce-Back** method:

$$f_{\bar{i}}(\boldsymbol{x}_\mathrm{b}, t+\Delta t) = -f_i^*(\boldsymbol{x}_\mathrm{b}, t) + 2w_i\rho_\mathrm{w}\left(1+\frac{(\boldsymbol{c}_i\cdot\boldsymbol{u}_\mathrm{w})^2}{2c_s^4}-\frac{\boldsymbol{u}_\mathrm{w}^2}{2c_s^2}\right) \tag{2.89}$$

The working principle is the exact same as for the regular BB, although the formula is altered and most notably, the populations are bounced back with their sign reversed. Now, ρ_w is prescribed and $\boldsymbol{u}_\mathrm{w}$ has to be estimated. One possible way is by

extrapolation through the formula

$$\boldsymbol{u}_{\mathrm{w}} \approx \boldsymbol{u}(\boldsymbol{x}_{\mathrm{b}}) + \frac{\Delta x}{2}\Big(\boldsymbol{u}(\boldsymbol{x}_{\mathrm{b}}) - \boldsymbol{u}(\boldsymbol{x}_{\mathrm{b}+1})\Big), \tag{2.90}$$

where the subscript $\mathrm{b}+1$ refers to the interior node one step in from $\boldsymbol{x}_{\mathrm{b}}$ in the direction of the boundary normal. [1]

This boundary condition can also be formulated with an extra second order correction term for increased accuracy, yielding the formula

$$\begin{aligned} f_{\bar{i}}(\boldsymbol{x}_{\mathrm{b}}, t+\Delta t) = &-f_i^*(\boldsymbol{x}_{\mathrm{b}}, t) + 2w_i\rho_0\left(\frac{\rho_{\mathrm{w}}}{\rho_0} + \frac{(\boldsymbol{c}_i\cdot\boldsymbol{u}_{\mathrm{w}})^2}{2c_s^4} - \frac{\boldsymbol{u}_{\mathrm{w}}^2}{2c_s^2}\right) \\ &+ \left(2-\frac{1}{\tau}\right)\left(f_i^+(\boldsymbol{x}_{\mathrm{b}}, t) - w_i\rho_0\left(\frac{\rho_{\mathrm{b}}}{\rho_0} + \frac{(\boldsymbol{c}_i\cdot\boldsymbol{u}_{\mathrm{b}})^2}{2c_s^4} - \frac{\boldsymbol{u}_{\mathrm{b}}^2}{2c_s^2}\right)\right), \end{aligned} \tag{2.91}$$

where

$$f_i^+ = \frac{1}{2}\left(f_i + f_{\bar{i}}\right) \tag{2.92}$$

which is calculated using the pre-collision f. ρ_0 is the mean density in the decomposition of mean and fluctuation $\rho = \rho_0 + \rho'$. [7]

2.3.4.4 Symmetry BC

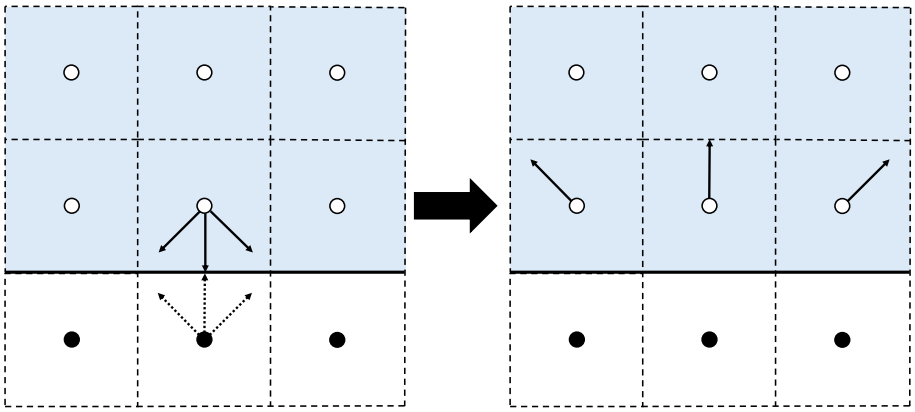


Figure 2.5: The symmetry boundary condition. In the boundary node opposite to the fluid node, mirrored versions of the populations pointing back into the domain are stored and represented by the dotted line (·····) arrows. These are then streamed to the reflected locations. Otherwise: Fluid nodes; ○, boundary/outside nodes; ●, physical boundary; ——, computational cell face; -----.

The symmetry BC may be used when there is mirror symmetry anywhere in the domain or as a free-slip condition. As opposed to the BB, the populations are reflected specularly and therefore conserves tangential momentum. The formula for the **symmetry boundary condition** may be written as

$$f_j(\boldsymbol{x}_{\mathrm{b}} + c_{j,\mathrm{t}}\Delta t, t+\Delta t) = f_i^*(\boldsymbol{x}_{\mathrm{b}}, t), \tag{2.93}$$

where j relates to i in such a way that the normal component is reversed, i.e. $c_{j,\mathrm{n}} = -c_{i,\mathrm{n}}$, while the tangential component is preserved: $c_{j,\mathrm{t}} = c_{i,\mathrm{t}}$. The symmetry BC is illustrated in Figure 2.5. Its biggest weakness is that it behaves identical to BB at corners, which turns it from a free-slip to no-slip condition at places if more complex geometries are approximated using a stair-stepping approach. [1]

2.3.4.5 Periodic BC

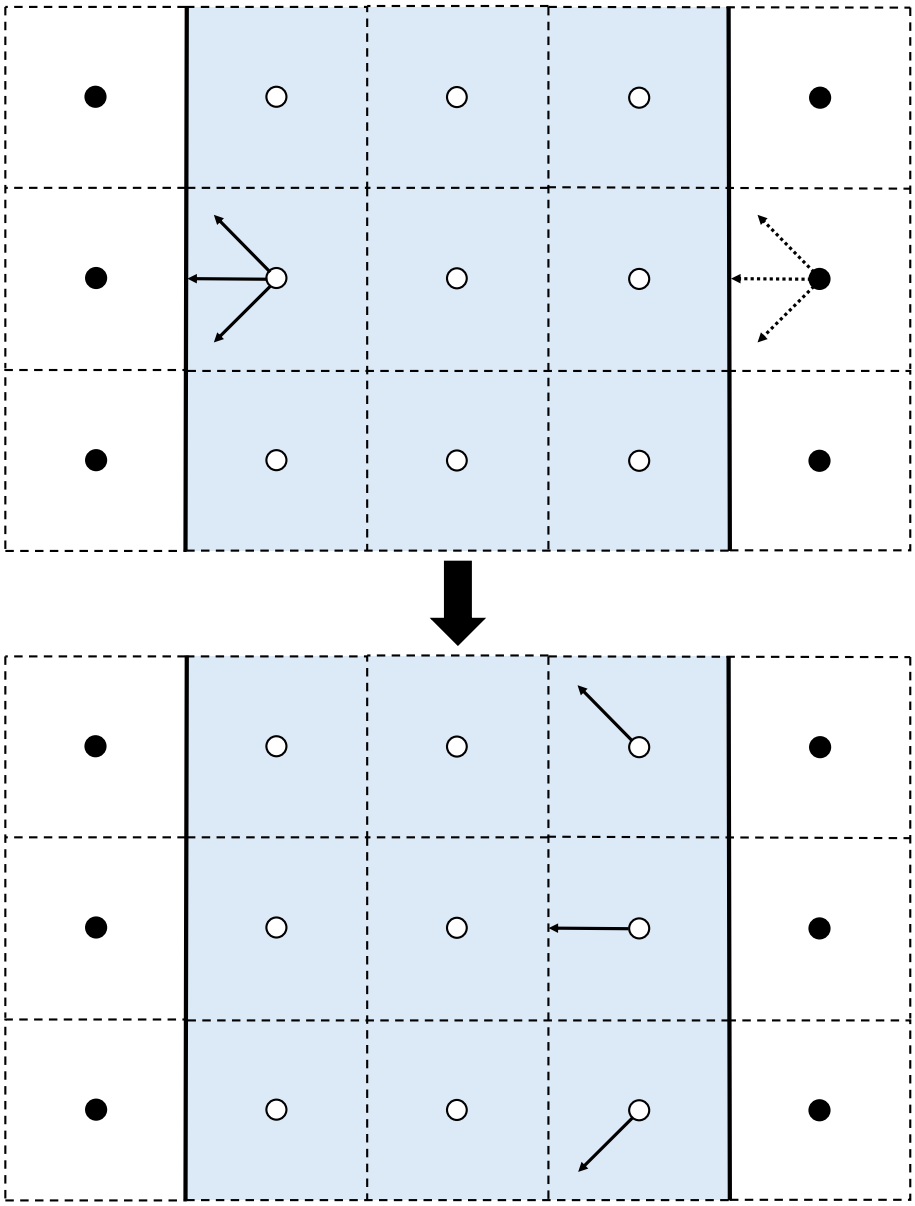


Figure 2.6: The periodic boundary condition. In the boundary node on the opposite side of the domain, copied versions of the populations pointing into the domain are stored and represented by the dotted line (·····) arrows. These are then streamed to their new locations. Otherwise: Fluid nodes; ○, boundary/outside nodes; ●, physical boundary; ——, computational cell face; -----.

Periodic BCs can be used when the flow is periodic, reconnecting the outlet of the domain back to the inlet on the other side. This concept is also easily translated into the mesoscopic scale and the **periodic boundary condition** reads

$$f_i^*(\boldsymbol{x}, t) = f_i^*(\boldsymbol{x} + \boldsymbol{L}, t), \tag{2.94}$$

where $\boldsymbol{L}$ is the vector between the periodic boundaries. Figure 2.6 shows the process of streaming the populations over to the other side of the domain. [1]

2.3.4.6 Initial Conditions

The simplest initialisation of the populations is by simply setting them to the equilibrium computed from initial values for the macroscopic variables ρ and $\boldsymbol{u}$. If these are unknown, the initial values $\rho = 1$ (in non-dimensional *lattice units*) and $\boldsymbol{u} = \boldsymbol{u}_0$ could be used, where $\boldsymbol{u}_0$ is some reference velocity, such as an inlet velocity or similar. These simple ICs might severely affect the simulation, however, and require a lot of expensive simulation time before any undesired behaviour has exited the system. One such behaviour could be the creation of undesired sound waves, since the standard LBM is a *weakly* compressible fluid solver. All of the BCs presented in the previous sections are reflective, and since the LBM does not suffer from high numerical dissipation, these sound waves might remain undamped for a long time. Ideally, all of the mesoscopic populations should be specified exactly, but some initialisation algorithm which runs the simulation until sufficient convergence while keeping the initial velocity $\boldsymbol{u}_0$ fixed could accomplish this task rather efficiently. Lastly, it is important that the simulation starts with collision rather than streaming, since the macroscopic variables which determined the initialised state should be used in a collision step. If the streaming is performed first, the populations will be redistributed and the macroscopic variables changed accordingly. [1]

2.3.5 Lattice Units

As already touched upon, the non-dimensionalisation performed in Section 2.3.1.1, results in the simulation being performed in the so-called **lattice units**. There are a certain freedom in choosing these units, and usually Δx, Δt and the average density ρ_0 are all set to equal 1 in the lattice units. Then, the conversion factors C to get the regular units are equal to the dimensional values:

$$C_\ell = \Delta x, \quad C_t = \Delta t, \quad C_\rho = \rho_0. \tag{2.95}$$

Almost all other quantities can be expressed through these conversion factors using dimensional analysis, the exception being the absolute pressure. In the LBM equation of state in Eq. (2.7), the average pressure was taken to relate to the average density through $p_0 = c_s^2 \rho_0$. This is not true in general and only taken as a fact to simplify the expression for the equation of state, which is allowed since only the pressure gradient is required to correctly simulate fluid behaviour. To actually compute the correct absolute pressure, the fluctuating pressure is found through $p' = c_s^2 \rho'$ and the total pressure as

$$p = p_0 + p', \tag{2.96}$$

where p_0 is the known reference pressure. [1]

An example where only the dimensional analysis is required, is the conversion factor for the kinematic viscosity ν, which is $C_\nu = C_\ell^2/C_t$, since ν has the units [m^2/s]. In the standard LBM, the viscosity can then be shown to depend on the simulation parameters through the relation

$$\nu = c_s^{\star 2}\left(\tau^\star - \frac{1}{2}\right)\frac{\Delta x^2}{\Delta t}, \tag{2.97}$$

where the starred variables are the usual non-dimensional parameters explicitly marked to distinguish from the dimensional non-starred quantities. This limits the choice of the three parameters $\tau^\star$, Δx and Δt to two of them, which can be chosen freely. Due to the stability condition $\tau^\star > 1/2$ found in Section 2.3.2.1, this necessitates the lattice to be refined when trying to simulate low viscosities (for example in cases of high Reynolds number $\mathrm{Re} = u_0\ell/\nu$). Since the standard LBM has an error which scales with Ma^2 (see Section 2.3.1.3), the fluid velocity in the simulation should be kept well below $c_s^\star$. Again, this puts an upper limit on Δx for high Re flows. At this point it has not yet been explicitly mentioned, but all non-dimensional numbers are the same in physical and lattice units. Therefore, the problem should be specified in terms of non-dimensional numbers and the available freedom of choice for the simulation parameters should be used to find a suitable trade-off between resolution, stability and accuracy. [1]

2.4 Extensions of the LBM

There are a multitude of extensions to the LBM, aiming to alleviate the limitations of the standard formulation [1]. In this section, a few extensions that were explored as part of the thesis work are presented as well as an outlook on extensions of interest for further investigation.

2.4.1 Thermal Flows

One important step towards simulating higher Ma flows is to use a thermal LBM model. The isothermal assumption in the standard LBM was made to make it consistent with the equation of state in Eq. (2.7). The consequence is a globally constant speed of sound with a specific heat ratio $\gamma = 1$.

2.4.1.1 Isentropic Non-Isothermal LBM

Using the ideal gas law of the form used in Eq. (2.2) with the isentropic assumption ($s = s_0$), yields the isentropic relation

$$\frac{p}{p_0} = \left(\frac{\rho}{\rho_0}\right)^\gamma. \tag{2.98}$$

Then p is eliminated from the usual ideal gas law

$$p = \rho\theta_0\left(\frac{\rho}{\rho_0}\right)^{\gamma-1} = \rho RT, \tag{2.99}$$

with

$$\frac{p_0}{\rho_0} := RT_0 = \theta_0. \tag{2.100}$$

Using the non-dimensionalisations

$$\rho^\star := \frac{\rho}{\rho_0}, \quad \theta^\star = \frac{RT}{\mathcal{U}^2} \tag{2.101}$$

results in a final expression for $\theta^\star$:

$$\theta^\star = \theta_0^\star \rho^{\star\gamma-1}. \tag{2.102}$$

This is then inserted into the truncated velocity discretised equilibrium distribution function in Eq. (2.54). With a thermal model, the speed of sound for a calorically perfect ideal gas is calculated as [2]

$$c_s = \sqrt{\gamma RT}. \tag{2.103}$$

The characteristic velocity may be chosen as

$$\mathcal{U} = \frac{c_s}{\sqrt{\gamma}} \tag{2.104}$$

resulting in $\theta_0^\star = 1$. Thus, thermal flows with a flexible γ may be simulated using only a very minor modification to the LBM, although limited to cases of isentropic flow.

2.4.1.2 Dual Distribution Function

To simulate thermal flows not limited to the isentropic condition of the previous section, the **dual distribution function (DDF) model** can be used. Here, the equilibrium distribution is formulated for polyatomic molecules, as opposed to the monoatomic assumption made in Section 2.2.3.3. The additional $K = n_{\mathrm{DOF}} - d$ number of DOF corresponding to internal DOF are contained in a vector $\boldsymbol{\eta}$. f^{eq} then reads

$$f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\eta}, t) = \frac{\rho}{(2\pi RT)^{(d+K)/2}} \mathrm{e}^{-\frac{(\boldsymbol{\xi}-\boldsymbol{u})^2+\boldsymbol{\eta}^2}{2RT}}. \tag{2.105}$$

This expression can be separated into the two parts

$$\begin{aligned} f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\eta}, t) = \frac{\rho}{(2\pi RT)^{(d)/2}} \mathrm{e}^{-\frac{(\boldsymbol{\xi}-\boldsymbol{u})^2}{2RT}} \cdot \frac{1}{(2\pi RT)^{K/2}} \mathrm{e}^{-\frac{\boldsymbol{\eta}^2}{2RT}} \\ = f^{\mathrm{eq,mono}}(\boldsymbol{x}, \boldsymbol{\xi}, t) \cdot g^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\eta}, t), \end{aligned} \tag{2.106}$$

where $f^{\mathrm{eq,mono}}$ is exactly the same as the monoatomic version of f^{eq} in Eq. (2.31). Moreover, it can be shown that

$$\int g^{\mathrm{eq}} \, \mathrm{d}^K \eta = 1, \quad \int \boldsymbol{\eta}^2 g^{\mathrm{eq}} \, \mathrm{d}^K \eta = KRT, \tag{2.107}$$

leading to the definition of the two distribution functions

$$\tilde{f}(\boldsymbol{x}, \boldsymbol{\xi}, t) := \int f(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\eta}, t)\, \mathrm{d}^K \eta, \tag{2.108}$$

$$\tilde{g}(\boldsymbol{x}, \boldsymbol{\xi}, t) := \int \frac{\boldsymbol{\eta}^2}{2} f(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\eta}, t)\, \mathrm{d}^K \eta, \tag{2.109}$$

with their respective equilibrium distributions taking the forms

$$\tilde{f}^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, t) = \int f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\eta}, t)\, \mathrm{d}^K \eta = \frac{\rho}{(2\pi RT)^{d/2}} \mathrm{e}^{-\frac{(\boldsymbol{\xi}-\boldsymbol{u})^2}{2RT}}, \tag{2.110}$$

$$\tilde{g}^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, t) = \int \frac{\boldsymbol{\eta}^2}{2} f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\eta}, t)\, \mathrm{d}^K \eta = \frac{K\rho RT}{2(2\pi RT)^{d/2}} \mathrm{e}^{-\frac{(\boldsymbol{\xi}-\boldsymbol{u})^2}{2RT}}. \tag{2.111}$$

The resulting *density* distribution function $\tilde{f}$ is subject to the same constraints and evolution through the Boltzmann equation as the PDF in the standard LBM with the equilibrium distribution also remaining the same: $\tilde{f}^{\mathrm{eq}} = f^{\mathrm{eq,mono}}$. The new second distribution function $\tilde{g}$ is interpreted as a *potential energy* distribution. For the remainder of this section, the $\sim$ notation is dropped for brevity when referring to $\tilde{f}$ and $\tilde{g}$.

The potential energy distribution does also have moments, similar to the moments of f in Eqs. (2.18):

$$\int g\, \mathrm{d}^d \xi = \frac{K}{2} \rho RT, \tag{2.112a}$$

$$\int \boldsymbol{\xi} g\, \mathrm{d}^d \xi = \frac{K}{2} \rho RT \boldsymbol{u}, \tag{2.112b}$$

$$\int \xi_\alpha \xi_\beta g\, \mathrm{d}^d \xi = \frac{K}{2} \rho RT u_\alpha u_\beta + \frac{K}{2} \rho (RT)^2 \delta_{\alpha\beta}. \tag{2.112c}$$

The evolution equation of g reads

$$\frac{\partial g}{\partial t} + \xi_\alpha \frac{\partial g}{\partial x_\alpha} = \Omega_g, \tag{2.113}$$

where

$$\Omega_g = -\frac{1}{\tau_g}(g - g^{\mathrm{eq}}) + \frac{(\boldsymbol{\xi} - \boldsymbol{u})^2}{2\tau_{gf}}(f - f^{\mathrm{eq}}). \tag{2.114}$$

The new relaxation times are related through

$$\frac{1}{\tau_{gf}} = \frac{1}{\tau_f} - \frac{1}{\tau_g}, \tag{2.115}$$

where τ_f is the relaxation time of the BGK operator in the Boltzmann equation. When $\tau_f = \tau_g$, the original BGK model is obtained. The Prandtl number of this model is related to the relaxation time parameters through

$$\mathrm{Pr} = \frac{\tau_f}{\tau_g}. \tag{2.116}$$

The new governing equations are then discretised in order to incorporate into an LBM scheme. If the equilbrium distributions are discretised in velocity space in the same manner as the standard LBM, the low Mach number limit still holds and the only gain is that the temperature is also correctly simulated. There is, however, alternate discretisation schemes, which incorporates the DDF model, successfully simulating even supersonic flow, such as in the paper by *Li and Zhong.* [8]

2.4.2 Non-Reflecting BCs

The general strategy to implement non-reflecting boundary conditions is to solve the **local one-dimensional inviscid (LODI) equations** at the boundary. From these equations, incoming and outgoing characteristics can be identified. By suppressing incoming amplitude variations, the acoustic waves inside the domain are ideally allowed to propagate out of the system without reflection. By solving the LODI relations, updated values of the flow variables ρ and $\boldsymbol{u}$ consistent with the non-reflectiveness are found, which are enforced at the boundary through any Dirichlet type boundary condition of choice. [1], [7], [9]

In a paper by *Izquierdo and Fueyo* [7], the Navier-Stokes characteristic boundary condition (NS-CBC) developed by *Poinsot and Lele* [9] are successfully applied to link-wise boundary conditions for the LBM. The LODI equations are none other than the Euler equations without transverse terms. For an open boundary normal to the x-direction in 2D, they take the form

$$\frac{\partial \rho}{\partial t} + u_x \frac{\partial \rho}{\partial x} + \rho \frac{\partial u_x}{\partial x} = 0 \tag{2.117a}$$

$$\frac{\partial u_x}{\partial t} + u_x \frac{\partial u_x}{\partial x} + \frac{1}{\rho} \frac{\partial p}{\partial x} = 0 \tag{2.117b}$$

$$\frac{\partial u_y}{\partial t} + u_x \frac{\partial u_y}{\partial x} = 0 \tag{2.117c}$$

$$\frac{\partial (\rho E)}{\partial t} + \frac{\partial \Big(u_x (\rho E + p) \Big)}{\partial x} = 0. \tag{2.117d}$$

As the standard LBM is isothermal, the energy equation is not really needed here, but for a thermal model such as the ones described in the previous sections, it would. Also, Izquierdo and Fueyo propose including the entropy characteristic stemming from the energy equation even in the isothermal LBM with a specific heat ratio of $\gamma \approx 1.2$ for best results [7]. Collecting the primitive variables of Eqs. (2.117) into a vector

$$\boldsymbol{m} = (\rho, u_x, u_y, E), \tag{2.118}$$

results in the compact form of Eqs. (2.117):

$$\frac{\partial}{\partial t} \boldsymbol{m} + \boldsymbol{\Gamma}_x \frac{\partial}{\partial x} \boldsymbol{m} = 0. \tag{2.119}$$

The amplitude variations $\boldsymbol{\mathcal{L}}_x$ can then be computed as

$$\boldsymbol{\mathcal{L}}_x = \boldsymbol{\Lambda} \boldsymbol{S} \frac{\partial}{\partial x} \boldsymbol{m}, \tag{2.120}$$

where $\boldsymbol{S}$ is a matrix of left eigenvectors of $\boldsymbol{\Gamma}_x$ and $\boldsymbol{\Lambda}$ a diagonal matrix of the eigenvalues of $\boldsymbol{\Gamma}_x$:

$$\lambda_1 = u_x - c_s, \tag{2.121a}$$

$$\lambda_2, \lambda_3 = u_x, \tag{2.121b}$$

$$\lambda_4 = u_x + c_s. \tag{2.121c}$$

$$\tag{2.121d}$$

Consequently, Eq. (2.119) may be written as

$$\frac{\partial}{\partial t}\boldsymbol{m} + \boldsymbol{S}^{-1}\boldsymbol{\mathcal{L}}_x = 0, \tag{2.122}$$

with

$$\boldsymbol{\mathcal{L}}_x = \begin{bmatrix}\mathcal{L}_{x,1}\\ \mathcal{L}_{x,2}\\ \mathcal{L}_{x,3}\\ \mathcal{L}_{x,4}\end{bmatrix} = \begin{bmatrix}(u_x - c_s)\left(\dfrac{\partial p}{\partial x} - \rho c_s \dfrac{\partial u_x}{\partial x}\right)\\ u_x \dfrac{\partial u_y}{\partial x}\\ u_x\left(c_s^2 \dfrac{\partial \rho}{\partial x} - \dfrac{\partial p}{\partial x}\right)\\ (u_x + c_s)\left(\dfrac{\partial p}{\partial x} + \rho c_s \dfrac{\partial u_x}{\partial x}\right)\end{bmatrix}. \tag{2.123}$$

Note that $\mathcal{L}_{x,3} = 0$ when the standard LBM equation of state $p = c_s^2\rho$ is used. It is here, that the empirical value of $\gamma \approx 1.2$ would be used along with the conventional speed of sound relation

$$c_s^2 = \gamma\frac{p}{\rho}. \tag{2.124}$$

The CBC method involves predicting the change in the primitive variables at the boundary. Therefore, the LODI relations are recast as the time derivatives in terms of the amplitude variations:

$$\frac{\partial \rho}{\partial t} = -\frac{1}{c_s^2}\left(\frac{1}{2}(\mathcal{L}_{x,4} + \mathcal{L}_{x,1}) + \mathcal{L}_{x,2})\right) \tag{2.125a}$$

$$\frac{\partial u_x}{\partial t} = -\frac{1}{2\rho c_s}(\mathcal{L}_{x,4} - \mathcal{L}_{x,1}) \tag{2.125b}$$

$$\frac{\partial u_y}{\partial t} = -\mathcal{L}_{x,2}. \tag{2.125c}$$

[7], [9]

2.4.2.1 Outlet

The Anti-Bounce-Back BC described in Section 2.3.4.3 is used to create a non-reflecting outlet BC. The extrapolation of variables to the wall is done through the formula in Eq. (2.90), here restated with the time dependence made explicit:

$$\phi(\boldsymbol{x}_\mathrm{w}, \hat{t}) \approx \phi(\boldsymbol{x}_\mathrm{b}, t) + \frac{\Delta x}{2}\Big(\phi(\boldsymbol{x}_\mathrm{b}, t) - \phi(\boldsymbol{x}_{\mathrm{b}+1}, t)\Big), \tag{2.126}$$

where $\hat{t} = t + \Delta t/2$ and ϕ any primitive flow variable. Recall that the boundary lies at a distance $|\boldsymbol{x}_\mathrm{w} - \boldsymbol{x}_\mathrm{b}| = \Delta x/2$. To approximate the gradients at the boundary, the following second-order backwards extrapolation is used:

$$\frac{\partial \phi}{\partial x}(\boldsymbol{x}_\mathrm{w}, \hat{t} - \Delta t) \approx \frac{1}{3\Delta x}\Big(8\phi(\boldsymbol{x}_\mathrm{w}, \hat{t} - \Delta t) - 9\phi(\boldsymbol{x}_\mathrm{b}, \hat{t} - \Delta t) + \phi(\boldsymbol{x}_{\mathrm{b}+1}, \hat{t} - \Delta t)\Big). \tag{2.127}$$

For an outlet in the positive x-direction, the incoming characteristic $\mathcal{L}_{x,1}$ would be set to zero for a completely non-reflecting BC, but to be able to enforce a specific pressure p_∞, a linear relaxation model is used instead:

$$\mathcal{L}_{x,1}(\boldsymbol{x}_\mathrm{w}, \hat{t} - \Delta t) = k_1(p(\boldsymbol{x}_\mathrm{w}, \hat{t} - \Delta t) - p_\infty), \tag{2.128}$$

where

$$k_1 = \sigma_1(1 - \mathrm{Ma}^2)\frac{c_s}{L}. \tag{2.129}$$

Here, L is the length from inlet to outlet, Ma is the maximum Mach number in the domain and σ_1 is a tuning parameter which should be in the range $0.58 < \sigma_1 < \pi$. The remaining amplitude variations are calculated using Eq. (2.123). These are then inserted into the expressions for the time derivatives in Eqs. (2.125) and the updated variables on the boundary are approximated using forward Euler:

$$\phi(\hat{t}) \approx \phi(\hat{t} - \Delta t) + \frac{\partial \phi}{\partial t}(\hat{t} - \Delta t). \tag{2.130}$$

[7]

2.4.2.2 Inlet

The CBC can be applied to an inlet very similarly to the outlet. The major difference is that the number of incoming and outgoing characteristics are switched. The incoming characteristics may either be solved for in Eqs. (2.125) imposing the variables and their derivatives at the boundary [9] or using similar relaxation models as in Eq. (2.128) for partial non-reflectiveness [7].

2.4.3 Multiple Relaxation Time Schemes

As mentioned in Section 2.3.5, high Re flows present a challenge in that the grid needs to be very refined to keep the relaxation time τ above the stability limit of $1/2$. By replacing the BGK collision operator with operators using **two relaxation times (TRT)** or **multiple relaxation times (MRT)**, it is possible to decouple the viscosity and stability dependencies. For example in TRT, which has the two relaxation times τ^+ and τ^-, only one of them is connected to the viscosity, leaving the other as a free parameter which can be tweaked for accuracy and stability of the simulation. MRT is even more flexible, but the biggest advantage of TRT is its ease of implementation. By decomposing the PDF (and f^{eq}) into symmetric and anti-symmetric parts

$$f_i^+ = \frac{f_i + f_{\bar{i}}}{2}, \quad f_i^- = \frac{f_i - f_{\bar{i}}}{2}, \tag{2.131}$$

the **force-free Lattice-Boltzmann equation using TRT** reads

$$\begin{aligned} &f_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) \\ &\quad = f_i(\boldsymbol{x}, t) - \frac{\Delta t}{\tau^+}\Big(f_i^+(\boldsymbol{x}, t) - f_i^{+\mathrm{eq}}(\boldsymbol{x}, t)\Big) - \frac{\Delta t}{\tau^-}\Big(f_i^-(\boldsymbol{x}, t) - f_i^{-\mathrm{eq}}(\boldsymbol{x}, t)\Big). \end{aligned} \tag{2.132}$$

Due to the symmetric structure of TRT, the collision step can be performed for only half of the stencil, with the other half following automatically. This makes the TRT implementation just as efficient as BGK. [1]

3
Methods

As part of the thesis work, a code written in Python implementing the basic Lattice-Boltzmann method was supplied. In this chapter, the structure and state of the code as it was received is described as well as the developments made. In Chapter 4, a few test cases are also presented which were instrumental in influencing the direction of the development of the code. The results of these test cases are then presented in Chapter 5.

3.1 Description of the Supplied Code

The code was structured in an object orient fashion with Python classes as objects representing different levels of the problem. At the top level, the main code creates an object called the app, which specifies the specific flow problem to be simulated. Here, a mesh is created, boundary and initial conditions specified and a stencil selected. In the code's initial state, any exporting or plotting of field variables had to also be specified in the app. The most essential object contained in the app object is the lattice object, which in turn contains cell objects. These cell objects are identified through a cell index and contains a vector specifying its position in space, an array pointing to its neighbour cells as well as an array with the corresponding index i of the velocity $\boldsymbol{c}_i$ pointing from the neighbour back to the cell. A diagram of these contents of a cell object is shown in Figure 3.1. If the cell is a boundary cell, it also contains a property class of its specified boundary condition, a normal vector for the boundary and an array listing the index j opposite to i in regard to the normal (as defined in Section 2.3.4.4).

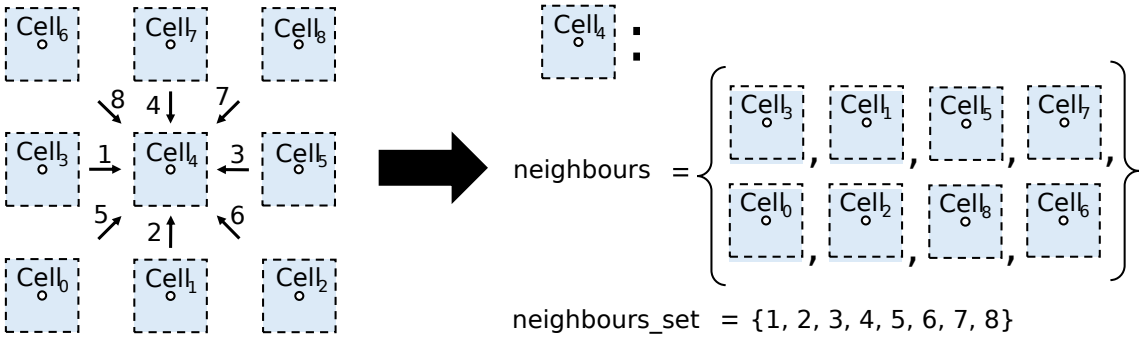


Figure 3.1: Diagram showing how the neighbours and associated velocity directions are stored in a cell.

3.1.1 Lattice Methods

Apart from all of the computational cells, the lattice object also contains arrays of the variables and constants (such as $f_i, f_i^{\mathrm{eq}}, \boldsymbol{u}, \rho, c_s, w_i$). The variable arrays are flattened in space and the values of cell n are stored at index n in the arrays. The lattice class contains a number of methods that can be called on the lattice object to perform the different steps of the LBM algorithm. These methods are (in order of execution during the LBM algorithm)

1. apply_force() – calculates the source term S_i from Eq. (2.84):
$$S_i = \left(1 - \frac{\Delta t}{2\tau}\right) F_i, \quad F_i = w_i \left(\frac{c_{i\alpha}}{c_s^2} + \frac{(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})u_\beta}{c_s^4}\right) F_\alpha. \tag{3.1}$$
2. collision() – applies collision as in Eq. (2.86a):
$$f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) - \frac{\Delta t}{\tau}\Big(f_i(\boldsymbol{x}, t) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t)\Big) + S_i(\boldsymbol{x}, t)\Delta t. \tag{3.2}$$
3. treat_boundary() – calls the apply() method on each boundary cell's BC class to prepare the boundary populations for streaming. The modified populations f_i are stored in the boundary cells as discussed in Section 2.3.4.1.
4. streaming() – streaming is performed as in Eq. (2.86b):
$$f_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i^*(\boldsymbol{x}, t). \tag{3.3}$$
Figure 3.2 contains a more detailed explanation of the code implementation.
5. macroscopic() – compute macroscopic variables through the discrete moments in Eq. (2.85):
$$\rho = \sum_i f_i, \quad \rho\boldsymbol{u} = \sum_i \boldsymbol{c}_i f_i + \frac{\Delta t}{2}\sum_i \boldsymbol{c}_i F_i. \tag{3.4}$$
6. equilibrium() – calculate f_i^{eq} from the macroscopic variables using Eq. (2.57):
$$f_i^{\mathrm{eq}} = w_i\rho\left(1 + \frac{c_{i\alpha}u_\alpha}{c_s^2} + \frac{u_\alpha u_\beta(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})}{2c_s^4}\right). \tag{3.5}$$

```
def streaming(self):
    f2 = np.zeros_like(self.f)
    f2[:] = self.f[:]
    for cell in self.internal_cells:
        for cell_n, i in zip(cell.neighbours, cell.neighbours_set):
            f2[i, cell.index] = self.f[i, cell_n.index]
    self.f[:] = f2
```

Figure 3.2: Code to perform the streaming step. First, a copy of the PDF is created (f2), to avoid data being overwritten. Then, each internal cell is looped over with each cell's neighbours (cell_n) being looped over along with the index i of the velocity $\boldsymbol{c}_i$ pointing from the neighbour to the cell. At each iteration of the loop, the streaming equation in Eq. (2.86b) is applied. Lastly, f2 is assigned back to the PDF variable stored in the lattice object.

3.1.2 Initialisation

First, a rectangular mesh of cells is created in the app class. The cells are initialised with their position vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ and the mesh array containing the cells stores the cell at position $\boldsymbol{x}$ at index $(x_1, x_2, \ldots, x_d)$ (the coordinates are positive integers). The BCs are then specified for the boundary cells followed by an algorithm initialising all of the cells' neighbours and other information contained in each cell. One of the BC types available is called 'dead', which means that the cell is not considered part of the domain. This allows for simulation domains of arbitrary shapes (within the limits of a rectangular grid). The mesh is finally used to create the lattice object where it is flattened as described in Section 3.1.1. Here, all cells except for the ones marked 'dead' receive a unique identifying index. Lastly, the mesoscopic populations are initialised as

$$f_i = f_i^{\mathrm{eq}}(\rho_0, \boldsymbol{u_0}), \tag{3.6}$$

where ρ_0 and $\boldsymbol{u}_0$ are the specified initial conditions.

3.1.3 Iteration

In the main code, the app object is created and then the LBM algorithm is applied through calling of the lattice methods in a loop. A residual is calculated as the difference in the norm of the velocity array before and after a complete iteration. Every few iterations (defined by a variable in the code), when the max iteration is reached or when the residual is less than a specified tolerance, an update method of the app object is called as well as information such as the value of the residual, maximum f, u_α, ρ and measured time for each substep of the iterations are printed to the terminal.

3.2 Needs for Further Development

The most pressing need for further development was the run time as the structure of the code was not optimised for speed in Python and no parallelisation was implemented. The code as it was received had also not properly defined an algorithm to find the normal of all possible corner types – especially not in 3D. Another needed feature was a general exporting algorithm for easier post processing. Finally, no working non-reflecting BC was present in the original code. This section describes the methods used in the development of these features that were initially lacking.

3.2.1 Exporting Mesh Plots

The first development to the code was to write a method that plots the lattice including marking the boundary conditions. The coordinate of every cell is found along with the applied BC. All cells not marked as 'dead' are used to define the lines creating the lattice plot. The BC information is also plotted.

3.2.2 Code Optimisation and Parallelisation

As already mentioned, the code in its initial state was running very slowly. This is because Python is rather slow at looping over objects, as in the example code in Figure 3.2. Considering the code snippet in said figure, what is really happening is that information in one array are copied from certain components to other components in another array. This could also be accomplished by inputting lists of the indices for every copying action. Since the problem at hand is static (no moving geometry), such lists can be saved during an initial loop through the objects creating a map of sorts over the problem. This turns out to be much faster in Python and also lends itself to further optimisation through the Numba library [10].

3.2.2.1 Using the Numba Library

```
@njit(parallel=isParallel)
def streaming(f, f2, streaming_map):
    for ind in prange(streaming_map[0].size):
        i = streaming_map[0][ind]
        cell_index = streaming_map[1][ind]
        cell_n_index = streaming_map[2][ind]
        f2[i, cell_index] = f[i, cell_n_index]
```

Figure 3.3: Optimised code to perform the streaming step. The first line is the Numba decorator indicating that the function is to be optimised. The argument tells Numba to enable automatic parallelisation if the variable 'isParallel' is set to 'True' (by running the code on computers with different numbers of CPU cores, it was found that the parallelisation increased performance if the number of cores were greater than 6). Here, the copy of the PDF (f2) and the PDF itself are created in the streaming method in the lattice class, before being input as arguments into the streaming function of the calc module. Using Numba, it turns out that creating a loop as shown in the figure is the fastest way of running over the lists in the map. The 'prange' function indicates to Numba that the loop may be parallelised. The last line of code performs the same action as the very similar line in Figure 3.2, but the new structure of the code increases the speed of execution substantially. The copied PDF, f2, is assigned back to f in the lattice object, after the function in the figure has finished running.

The Numba library allows for radical speed up of Python code by compiling parts of the code, specified by certain decorators, to optimised machine code, bypassing the Python interpreter completely. Numba also has the capability of automatic parallelisation. To be able to bypass the Python interpreter, however, the parts of the code being optimised by Numba cannot include, or reference any Python objects. [10] This made the LBM code in its initial state extremely unfit for Numba optimisation. But, with the map approach described in Section 3.2.2 being applied to the methods in the lattice class (recall Section 3.1.1) that is called

during iteration, the calculations could be lifted out into a separate 'calc' module, where every function is marked for optimisation with the Numba decorator. Most of these calculation routines could also be further optimised by enabling the automatic parallelisation of Numba. An example of how the streaming code in Figure 3.2 was optimised is shown in Figure 3.3. There is also support for GPU based parallelisation using Nvidia CUDA [10], [11], which could increase performance even further, although this was deemed outside of the scope of this thesis.

3.2.2.2 Speeding Up the Initialisation

Another section of the code which ran quite slowly was the initialisation of the problem, where the lattice is built up and the cell objects gain their lists of neighbours. This step is also accomplished by iterating over every cell and finding the the cells that can be reached by the velocity stencil. This step also accounts for periodic BCs, adding the cells on the opposite side of the domain as neighbours. As the condition of no Python objects being violated, this section of the code is not possible to optimise with Numba. There was, however, room for improvement by manually parallelising the problem using the multiprocessing library [12]. The manual step consist of dividing the entire set of cells into subsets for which the neighbour finding algorithm is then run in separate processes started on separate CPU cores. The multiprocessing processes start entirely new versions of the Python interpreter, and as such, cannot communicate back to the original instance unless a shared memory is set up between the processes [12]. To avoid writing errors caused by multiple processes trying to access the same memory location, this memory is set up such that every process will only access the parts of the memory associated with its allocated subset of cells. After the processes have all finished, there is one loop over all the cells in the main process, where the information gathered from the parallel processes are applied to set the neighbours of every cell. This is much faster than the neighbour finding algorithm and making this optimisation to the initialisation well worth it.

It should be mentioned that the path of multiprocessing was a path also initially tried for the iteration part of the code, but the overhead of creating the processes manually seemed to consume the entire gain of the parallelisation, leading to only marginal speed up at best and even slower code at worst. The lessons learned during this failed attempt were still able to be applied, for instance in the neighbour finding algorithm just described, avoiding this path to have been explored in vain.

3.2.3 Exporting to VTK

The code as it was, came with a few basic cases already defined, but not every case had a proper exporting routine and having to define unique such routines for every new case seemed unnecessarily cumbersome. The solution to this problem was in the form of exporting the macroscopic field variables ρ and $\boldsymbol{u}$ using the legacy vtk file format. The .vtk files consist of a description of the mesh followed by the values of the variables in each cell. There is also a vtk series file which matches every exported vtk file to a specific time step. [13] This can then be imported in external software, such as ParaView, for post-processing like plotting and creating animations. The

only major downside is that the results of the simulation cannot be viewed in real time, although it is still possible to load the new files as they are exported during iteration allowing for analysis of part of the simulation before it has completed. The export routine that was created also includes an option to export in physical units, by specifying the conversion factors (see Section 2.3.5) for Δx and ρ_0 as well as the kinematic viscosity ν of the fluid.

3.2.4 Corner Detection and Handling

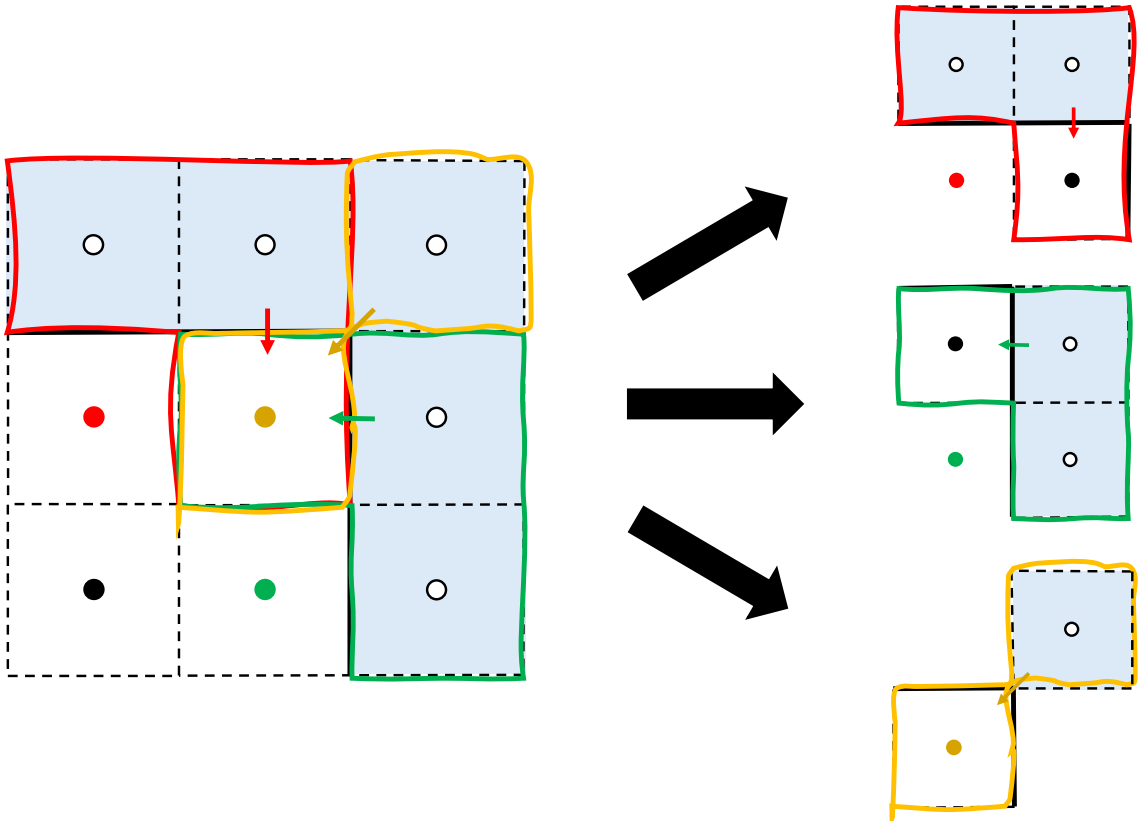


Figure 3.4: Convex corner in 2D being divided into multiple instances of itself. The neighbours associated with each instance is surrounded with strokes of different colours. The node containing the associated BC and the normal vector are also marked in the corresponding colour.

The corner handling of the initial code lacked any possibility of correctly handling any convex corner or edge. The first step towards a consistent implementation was the realisation that a convex corner might actually have multiple different boundary conditions. For example, in 2D, the two edges meeting in the corner might belong to edges with two different BCs, but there is only one boundary node to specify a BC. The solution that was chosen was to introduce a new type of BC for convex corners, where the BC for a certain edge in the corner is taken from the next boundary cell along the same edge. To store these new BCs, the convex corner BC was made to contain multiple copies of the corner cell itself, but each with its own BC, normal and

set of neighbours. This boundary condition is not meant to be used when specifying the problem; the corner cells automatically detect whether they are convex corners or not, relabelling themselves as convex corners and creating the copies of the cell with the correct BCs, normals and neighbours. Instead, the BC specified in the corner cell is used to determine the behaviour for populations pointing directly at the corner, as it is ambiguous from which edge the BC should be taken from, leaving this to be settled by the user of the code. Figure 3.4 illustrates how the BCs, normals and neighbours are divided into multiple instances of the corner cell.

3.2.4.1 Normal Detection in 2D

The algorithm to find the normal direction of a boundary cell in 2D is rather simple, but elegant, and was already featured in the supplied code. Looking at the stencils in Figure 2.1, or in Table 2.1, it can be noted that the shortest velocity vectors are numbered first, followed by increasing magnitude. This can be exploited; by looping over the cell neighbours and stopping as soon as a fluid cell is encountered, the principal directions (velocities with a magnitude of 1) will always be found first (see Figure 3.1 for a reference on how the neighbouring cells are stored). For a straight edge, there is only one fluid cell in the principal directions, and for a concave corner, there is only one fluid cell neighbour in total. Consequently, the normal finding algorithm simply takes the velocity direction from the first encountered fluid cell. However, for a convex corner, this breaks down as there are two fluid cell neighbours in principal directions. The solution in 2D is still simple; just ensure that the principal directions are always completely looped over and add the vectors to find the normal. Counting the number of fluid neighbours found in the principal directions also serves as the method of finding a convex corner in 2D.

3.2.4.2 User Specified Corner Behaviour

In Section 3.2.4, it was mentioned that the BC applied to a convex corner cell determines the behaviour of the cell only in the direction pointing directly at the corner. This is true also for concave corners, where this direction is the only one interacting with the corner BC. There is a specific type of combination of BCs meeting at a concave corner that should receive special attention. If one of the edges meeting in the corner is a symmetry BC, while the other edge is an open or periodic boundary, the symmetry BC should not be used in the corner cell. This is because the diagonal normal direction in the corner will cause the population pointing at it to reflect back to its origin, just like the Bounce-Back BC, instead of the specular reflection of a symmetry BC. The consequences of an incorrect specification is that the free-slip macroscopic behaviour turns into a no-slip one right at these corner points. Figure 3.5 provides a visual demonstration of this scenario.

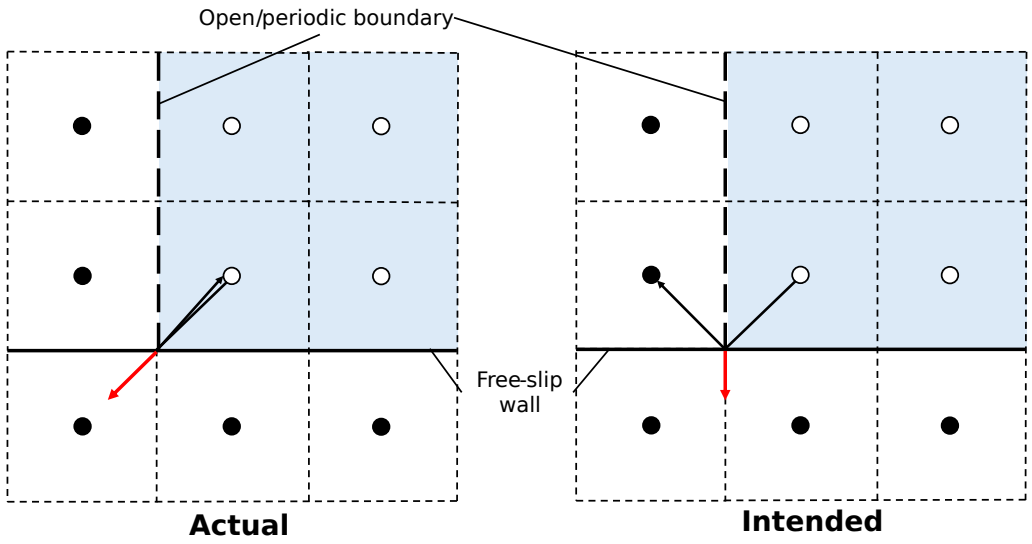
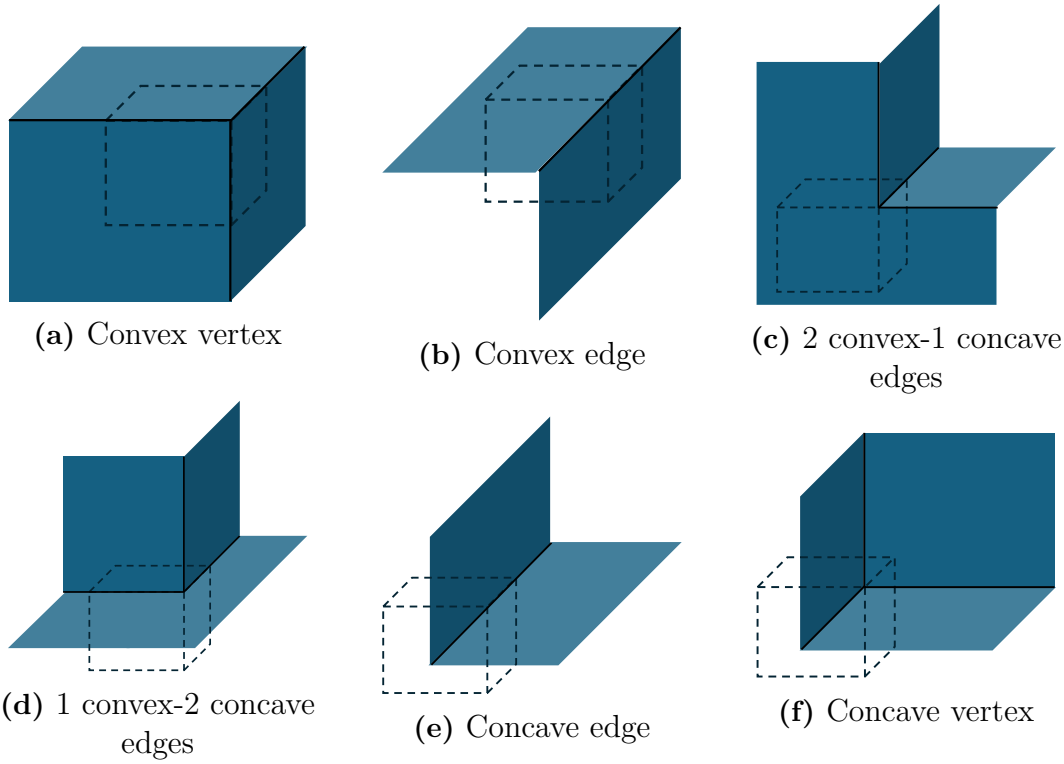


Figure 3.5: Concave corner in 2D with an open or periodic boundary meeting a free-slip (symmetry) wall. The BC in the corner itself is that of the wall. Actual and intended behaviour are compared, with the BC normal displayed as a red arrow. The actual behaviour is identical to BB, or a no-slip wall. The behaviour marked as intended is actually also inconsistent, since the populations cannot simply exit the fluid domain. The solution which produces both intended and consistent behaviour is to set the corner BC the same as the open/periodic boundary. This figure also demonstrates that the Bounce-Back BC can be placed in the corner and produce both intended and consistent behaviour.

3.2.4.3 Corner Identification in 3D

In 3D, the number of possible corner types are increased from two to six. This is because the corner might consist of one or three edges meeting at a point, with every edge being either convex or concave. In Figure 3.6, all six possible 3D corners are shown. Unfortunately, this diversity renders the simple method described in Section 3.2.4.1 unable to find a correct normal for every possible corner type in 3D. Additionally, the direction of the normal, which cells to copy BCs from and how the neighbour cells should be distributed is not entirely trivial. Every face, edge and vertex of a 3D corner should have its own normal and BC. In Table 3.1, every normal along with the associated BC and neighbour cells are listed for the four corner types with at least one convex edge (the entirely concave corner types only have one edge or vertex and thus, only one normal).

The resulting algorithm for identifying all of these 3D corner types consist of a decision tree counting the number of neighbouring fluid cells with each specific length of the velocity vectors in increasing order, stopping when there is only one possibility for the corner type. Since the algorithm also uniquely determines the corner type, every convex type corner are also marked as such when encountered. The D3Q15 stencil had to be disqualified from the code, due to the inability to distinguish every corner type uniquely if the $\sqrt{2}$ velocities are not available in the current stencil. Based on the fact that 27 abscissae are needed in 3D for the Gauss-Hermite quadrature to be exact, this can hardly be considered a great loss.



(a) Convex vertex **(b)** Convex edge **(c)** 2 convex-1 concave edges

(d) 1 convex-2 concave edges **(e)** Concave edge **(f)** Concave vertex

Figure 3.6: Every 3D corner type along with descriptive names.

2 convex-1 concave edges			1 convex-2 concave edges		
normal	BC	neighbours	normal	BC	neighbours
20	0	$\{19\}$	20	0	$\{19\}$
8	6	$\{7, 21\}$	12	2	$\{11, 25\}$
6	8	$\{5, 9, 11, 16, 18, 22, 23, 25\}$	8	6	$\{7, 21\}$
Convex vertex			**Convex edge**		
normal	BC	neighbours	normal	BC	neighbours
20	0	$\{19\}$	8	0	$\{7, 19, 21\}$
12	2	$\{11, 25\}$	4	2	$\{3, 11, 14, 17, 24, 25\}$
10	4	$\{9, 23\}$	2	4	$\{1, 9, 13, 15, 23, 26\}$
8	6	$\{7, 21\}$			
6	8	$\{5, 16, 18, 22\}$			
4	10	$\{3, 14, 17, 24\}$			
2	12	$\{1, 13, 15, 26\}$			

Table 3.1: Normals, BCs and neighbours for 3D corners. The naming and orientation of the corner types is defined in Figure 3.6. The numbering corresponds to vectors in the D3Q27 stencil as defined in Figure 2.1e. The BC and neighbours columns include the directions of the vectors pointing from the corner cell to the cells of interest. For example, the first row indicates that for the normal pointing directly at the corner (20), the BC is taken from the cell itself (0) and the only neighbour is the one opposite of the normal (19).

3.2.5 45° Symmetry BC

In Section 2.3.4.4, where the symmetry BC is presented, the concern regarding the fact that the BC behaves identically to the Bounce-Back BC at corners, meaning that boundaries aligned 45° relative to the lattice results in a stair-step-approximation turning the free-slip into no-slip. To evaluate the effect of this, the channel hump test case defined in Section 4.1 was used.

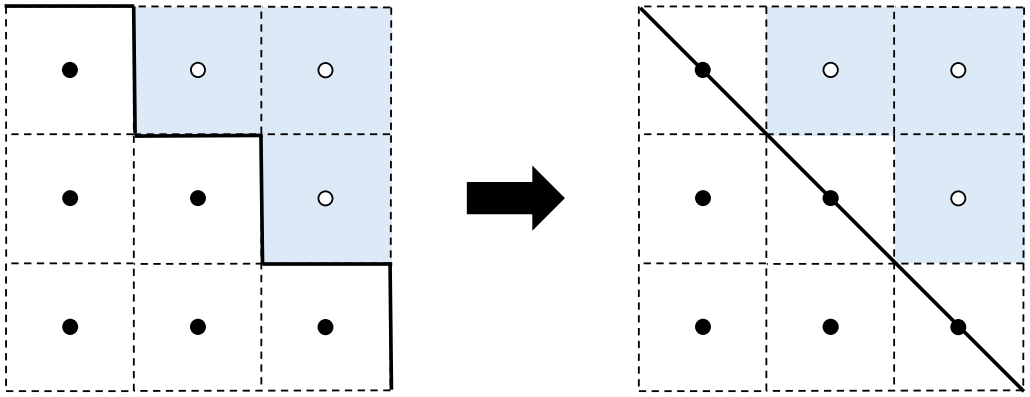


Figure 3.7: Transformation from stair-stepping approach to 45° symmetry BC. The normal and BC is now the same for all of the neighbours.

A method implemented to try to remedy this potential issues, was to introduce a version of the symmetry BC treating convex corners as 45° edges (see Figure 3.7). Since the regular symmetry BC reflects the populations halfway between the nodes in a single time step, the 45° version was made to save do this in two time steps, since the distance is doubled. The populations to be streamed back from the boundary in the next time step are saved in the corner boundary node pointing out of the domain. At the next time step, the pre-collision populations from the copy of the PDF created during streaming are reflected specularly relative to the normal of the corner.

3.2.6 Non-Reflecting BCs

The non-reflecting type CBC-BCs described in Section 2.4.2 were implemented and tested for both the Anti-Bounce-Back pressure outlet BC and the moving wall BB inlet BC using the travelling wave test case. For both the inlet and outlet, the time derivative extrapolation resulted in a positive feedback loop, where small variations were picked up and amplified until the solution diverged. Limited success was found in specific flow cases if a limiter was introduced, where the macroscopic variables at the boundary were not updated if the estimated time derivative was too small. This was far from being universally stable however, and in some cases the limiter had to be turned up to a level where the non-reflective effect was completely nullified. Noting the behaviour at the boundary when the reflections were suppressed did lead to a qualitative observation that the flow variables at the boundary must comply with the wave approaching the boundary to let it propagate out of the system without reflection. This observation sparked the simple idea of using a zeroth order

interpolation to the boundary, where the boundary values are simply approximated by the values in the cell next to the boundary. The idea was refined even further by considering the question:

- Is there an optimal distance to the boundary for which the reflections would be completely suppressed?

The extrapolation method used to estimate values at the boundary can be written on the form

$$\phi(\boldsymbol{x}_\mathrm{w}, t) \approx \phi(\boldsymbol{x}_\mathrm{b}, t) + L_\mathrm{w}\Big(\phi(\boldsymbol{x}_\mathrm{b}, t) - \phi(\boldsymbol{x}_{\mathrm{b}+1}, t)\Big), \tag{3.7}$$

where $L_\mathrm{w} = \Delta x/2$ is the distance $|\boldsymbol{x}_\mathrm{b} - \boldsymbol{x}_\mathrm{w}|$. In the travelling wave test case, defined in Section 4.2, alternate values for L_w which almost completely suppressed the reflected wave were found for a wide range of simulation parameters and combined into an empirical relation for L_w.

3.2.6.1 Accuracy of the New Non-Reflecting Outlet BC

Determining the accuracy of an outlet BC is always of interest. Ideally, the outlet should be accurate enough so that the domain does not have to be extended very far from the region of interest, saving precious computing resources. The new outlet BC described in the last section was first tested on Poiseuille flow, defined in Section 4.3, and then on the cylinder vortex shedding test case, defined in Section 4.4. In the second case, a simulation in a long domain was compared with a shorter domain, where the long domain was considered to give the correct behaviour which should be matched at the outlet of the short domain. The nature of the non-reflecting outlet BC does not allow specification of pressure (or rather density), which might cause it to drift slightly. To alleviate this issue, an ambient pressure is allowed to enter the Anti-Bounce-Back through the ρ_0 parameter in Eq. (2.91). To keep the non-reflectiveness and only try to match the ambient pressure when no waves are propagating through the boundary, a modified expression for ρ_0 was used:

$$\rho_{0,\mathrm{mod}} = \chi\rho_\mathrm{w} + (1-\chi)\rho_0, \tag{3.8}$$

where

$$\chi = \left(\frac{|\rho(\boldsymbol{x}_\mathrm{b}) - \rho(\boldsymbol{x}_{\mathrm{b}+1})|}{\max\left(\rho(\boldsymbol{x}_\mathrm{b}), \rho(\boldsymbol{x}_{\mathrm{b}+1})\right)}\right)^{0.001}. \tag{3.9}$$

3.2.7 Isentropic Thermal Model

Finally, the isentropic thermal model, introduced in Section 2.4.1.1, was implemented. Practically, the only modification needed was to alter the expression to calculate the equilibrium distribution from Eq. (2.57) to the form

$$f_i^\mathrm{eq} = w_i\rho\left(1 + \frac{c_{i\alpha}u_\alpha}{c_s^2} + \frac{\Big(u_\alpha u_\beta + (\theta_0\rho^{\gamma-1} - 1)c_s^2\delta_{\alpha\beta}\Big)\Big(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta}\Big)}{2c_s^4}\right). \tag{3.10}$$

It was tested on the isentropic vortex convection test case defined in Section 4.5.

4

Test Cases

Here, the test cases in which the LBM was evaluated are presented. Additional basic test cases such as Couette flow, backwards facing step, lid-driven cavity, 2D and 3D cavities as well as flow around a box were defined in the code and used for benchmarking of the execution speed and stability of the code but not evaluated for accuracy.

4.1 Channel Hump

A channel with a circular hump is defined in Figure 4.1 with the proportions of the domain being 25 : 4. The inlet is a moving wall Bounce-Back BC, the outlet is an Anti-Bounce-Back BC and the walls of the channel, including the hump, use the symmetric BC. The channel is initialised with the inlet velocity $\boldsymbol{u}_{\mathrm{w}} = (u_{\mathrm{w}}, 0)$, where $u_{\mathrm{w}} = 0.1$ in lattice units, which means Ma ≈ 0.17. The relaxation time is specified through the kinematic viscosity as

$$\tau = \frac{\nu}{c_s^2} + \frac{1}{2}, \tag{4.1}$$

where the viscosity, in turn, is specified by a Reynolds number defined in terms of the hump height h_{hump}:

$$\mathrm{Re} = \frac{u_{\mathrm{w}} h_{\mathrm{hump}}}{\nu}. \tag{4.2}$$

The aim of the test case is to see if inviscid behaviour is achievable, meaning the free slip of the symmetry boundary condition should not create any viscous boundary layer along the walls or over the hump.

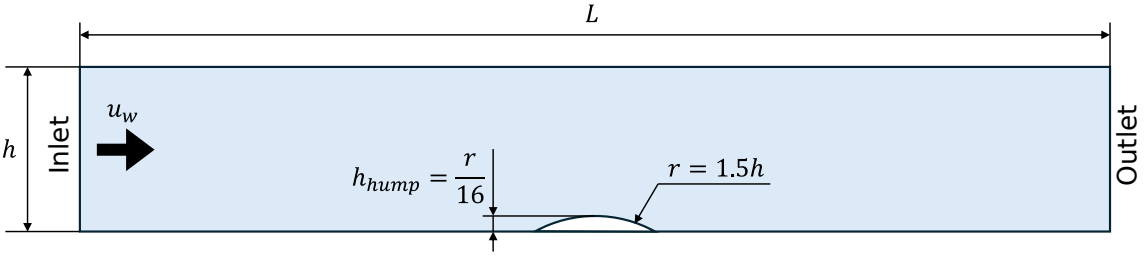


Figure 4.1: Sketch of the channel hump test case.

4.2 Travelling Wave

The test case consist of a long channel with limited height (2000×10 cells), making the case essentially 1D, but it is still defined in 2D to be able to evaluate the outlet BC for a 2D stencil. The inlet is a moving wall Bounce-Back BC, the outlet is an Anti-Bounce-Back BC and the walls of the channel use the symmetric BC. The channel is initialised with a velocity $\boldsymbol{u}_0 = (u_0, 0)$ and a single sinusoidal deviation with amplitude $u' = 0.01u_0$ and period t_p is created at the boundary at $t = 0$. The velocity u_0 is specified through the Mach number $\mathrm{Ma} = u_0/c_s$ and the kinematic viscosity (which also sets the relaxation time τ) is specified through a Reynolds number defined as

$$\mathrm{Re} = \frac{u_0 \cdot (c_s t_\mathrm{p})}{\nu} = \frac{\mathrm{Ma} \cdot c_s^2 t_\mathrm{p}}{\nu}. \tag{4.3}$$

The last rewrite shows that the viscosity is proportional to the ratio Ma/Re. The purpose of this test case is twofold; on the one hand, the evolution of the propagating wave through the long channel is of interest, and on the other hand, the reflecting behaviour of the outlet is tested.

4.3 Poiseuille Flow

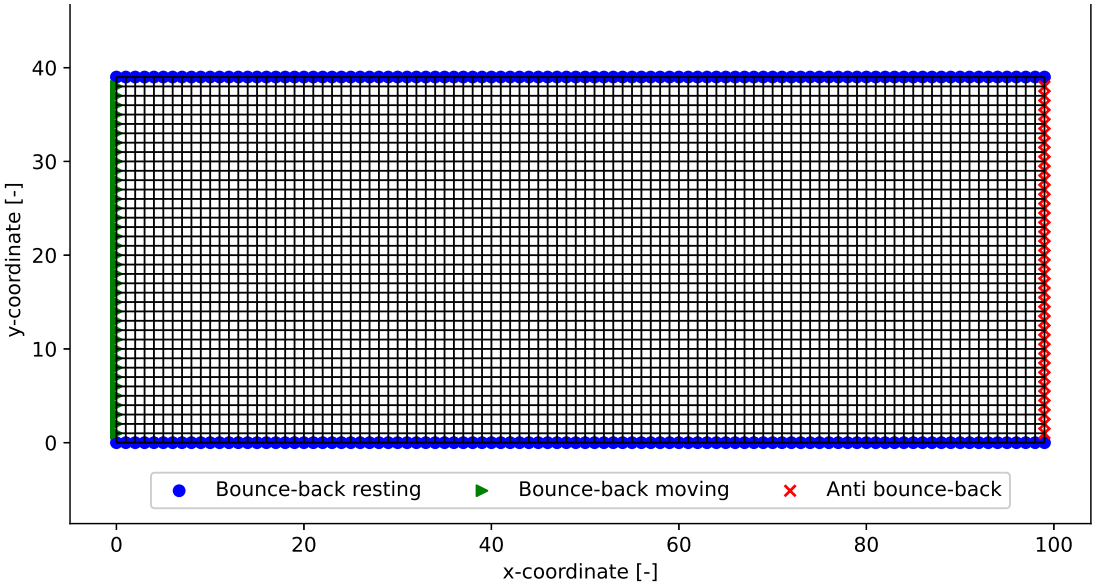


Figure 4.2: The Poiseuille lattice with boundary conditions marked.

The Poiseuille Flow is a test case for viscous flow in a channel which, when fully developed, has the analytical solution

$$u_x(y) = Cy(h - y) \tag{4.4}$$

for the velocity profile, where C is a proportionality constant. The test case is set up according to Figure 4.2. Considering the mass flow balance at the inlet with

a constant velocity $\boldsymbol{u}_\mathrm{w} = (u_\mathrm{w}, 0)$ and the fully developed flow, the proportionality constant can be computed as

$$C = \frac{6}{h^2} u_\mathrm{w}. \tag{4.5}$$

Consequently, the maximum velocity in the centre of the channel is determined by inserting this into Eq. (4.4) and evaluating at $y = h/2$:

$$u_{x,\max} = \frac{3}{2} u_\mathrm{w}. \tag{4.6}$$

The Poiseuille flow has already been used to extensively test the LBM, for example in the book by Krüger et al. [1], so the aim here is to primarily use the case to study how the velocity profile is affected by the outlet BC.

4.4 Cylinder Vortex Shedding

The viscous flow around a cylinder creates an unsteady wake pattern of alternating vortices usually called a *Kármán vortex street* [14]. The geometry of the test case is defined in Figure 4.3 with the dimensions 600×200 cells. The simulation is performed at a Reynolds number of $\mathrm{Re} = 150$, defined in terms of the inlet velocity and diameter of the cylinder and the Mach number for the inlet velocity is $\mathrm{Ma} \approx 0.069$. The purpose of using the test case in this thesis is to compare how the Kármán vortex street exits the domain. As a reference, a domain twice as long is used, which is considered as the 'correct' behaviour for the length of the short domain.

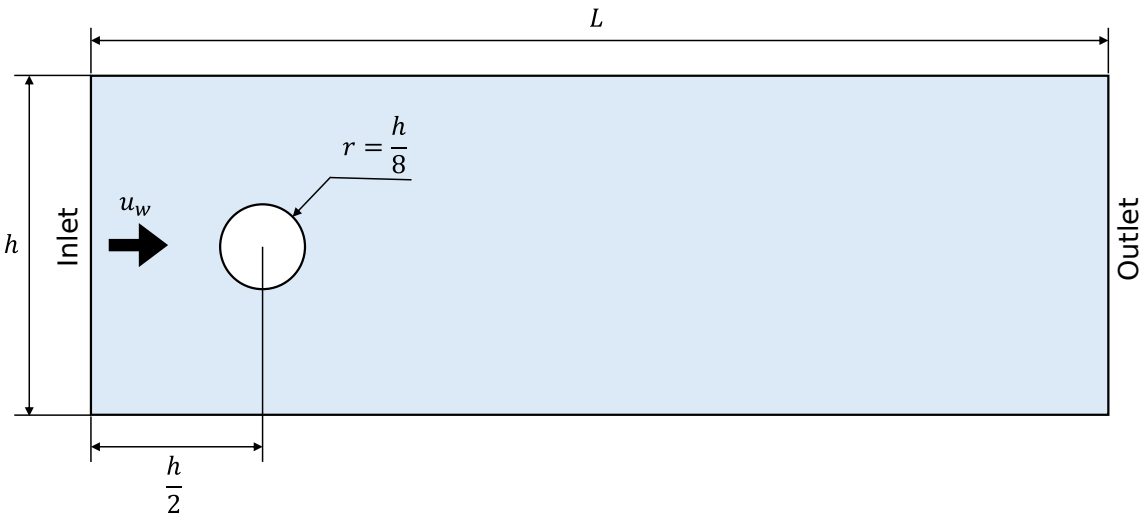


Figure 4.3: Sketch of the cylinder vortex shedding test case.

4.5 Isentropic Vortex Convection

The final test case is used to test the isentropic thermal model described in Section 2.4.1.1. There exists an analytical solution to the full unsteady 2D compressible Euler which introduces a vortex into a homogeneous flow field (with flow variables

denoted with index ∞) which is convected isentropically with the free stream velocity $\boldsymbol{u}_\infty$. The solution is described by the following expressions

$$\frac{u_x(x,y,t)}{c_{s,\infty}} = \frac{u_{\infty,x}}{c_{s,\infty}} - \frac{\mathcal{K}}{2\pi c_{s,\infty}} \bar{y} \mathrm{e}^{\alpha(1-\bar{r}^2)/2}, \tag{4.7a}$$

$$\frac{u_y(x,y,t)}{c_{s,\infty}} = \frac{u_{\infty,y}}{c_{s,\infty}} + \frac{\mathcal{K}}{2\pi c_{s,\infty}} \bar{x} \mathrm{e}^{\alpha(1-\bar{r}^2)/2}, \tag{4.7b}$$

$$\frac{T(x,y,t)}{T_\infty} = 1 - \frac{\mathcal{K}^2(\gamma-1)}{8\alpha\pi^2 c_{s,\infty}^2} \mathrm{e}^{\alpha(1-\bar{r}^2)}, \tag{4.7c}$$

$$\frac{\rho(x,y,t)}{\rho_\infty} = \left(\frac{T(x,y,t)}{T_\infty}\right)^{\frac{1}{\gamma-1}}, \tag{4.7d}$$

where $\mathcal{K}$ and α are the strength and decay of the vortex, respectively, and

$$\bar{x} = x - x_0 - u_{\infty,x} t \tag{4.8a}$$

$$\bar{y} = y - y_0 - u_{\infty,y} t \tag{4.8b}$$

$$\bar{r} = \sqrt{\bar{x}^2 + \bar{y}^2}. \tag{4.8c}$$

With the parameters $\mathcal{K} = 0.125$, $\alpha = 1$, $x_0 = 100$, $y_0 = 100$, $\rho_\infty = 1$ and $u_{\infty,x} = u_{\infty,y} = 0.05$, the vortex starts at $\boldsymbol{x} = (100, 100)$ and is convected to $\boldsymbol{x} = (300, 300)$ in 4000 time steps Δt. [15] The domain is therefore chosen with the dimensions 400×400 cells and the boundary conditions are of the periodic type.

5

Results and Discussion

5.1 Code Development

The mesh plotting routine mentioned in Section 3.2.1 was a good exercise to become familiar with the code structure, and Figure 4.2 showing the Poiseuille test case was created using this method. The optimisations described in Section 3.2.2 resulted in a speed up of the code by a factor of 30.

5.2 Test Cases

Here, the results from the test cases defined in the previous chapter are presented. The structure of this section follows the test cases' corresponding code developments presented in Chapter 3. All plots use the non-dimensional lattice units.

5.2.1 45° Symmetry BC

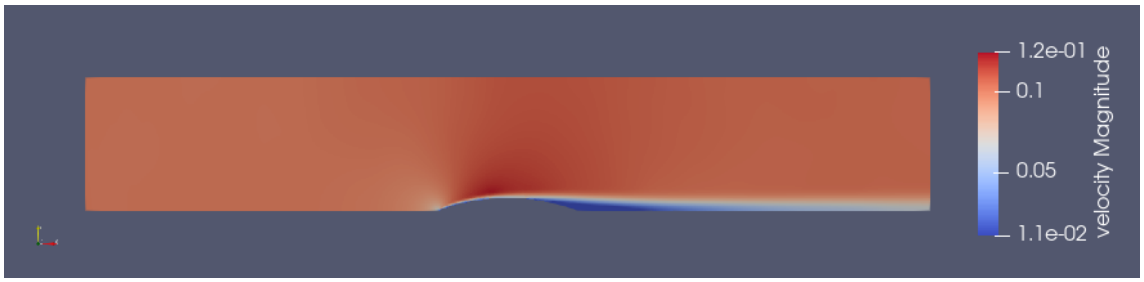


Figure 5.1: Results of the channel hump case at Re = 190.

Figure 5.1, shows that the rough approximation of the hump from Section 4.1 has created a viscous boundary layer, which is undesired in this situation. Figure 5.2 shows the results for the channel hump with the 45° Symmetry BC implemented and in Figure 5.3, a close-up of the hump is compared for the two BCs. It is evident that the new BC performs radically better, but it is still not perfect and a limiting factor in this case was the Reynolds number. Increasing it any further resulted in the grid having to be refined to stay above the stability limit of $\tau > 1/2$. To get around this problem, the TRT or MRT model described in Section 2.4.3 should be considered for implementation in further evaluation beyond the work of this thesis.

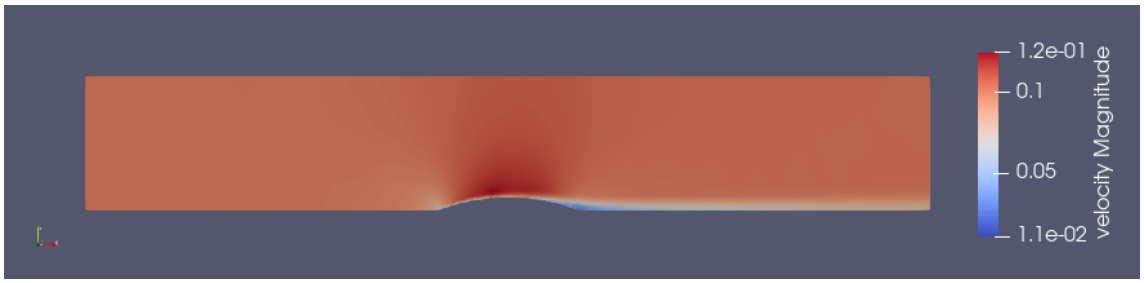
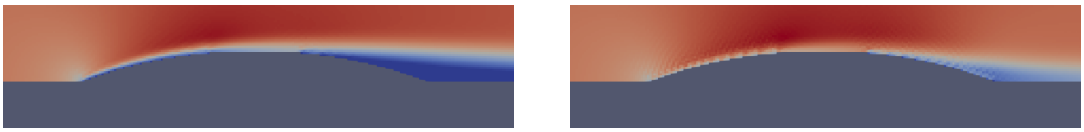


Figure 5.2: Results of the channel hump case with the 45° symmetry BC implemented. The boundary layer is much less pronounced compared to Figure 5.1.



(a) Regular **(b)** 45°

Figure 5.3: Close up comparison of the two channel hump case for the two different symmetry BCs. The boundary layer onset is delayed until the trailing half of the hump in the 45° case while it forms immediately in the regular case.

5.2.2 The Travelling Wave

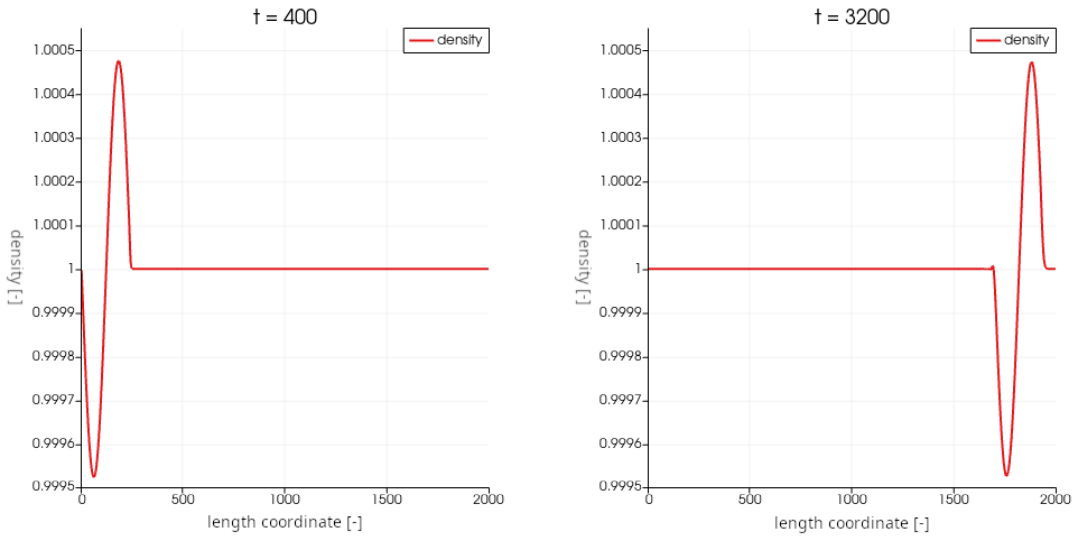


Figure 5.4: Density plots along the centre line of the travelling wave case with $\tau = 0.51$ at two different moments in times.

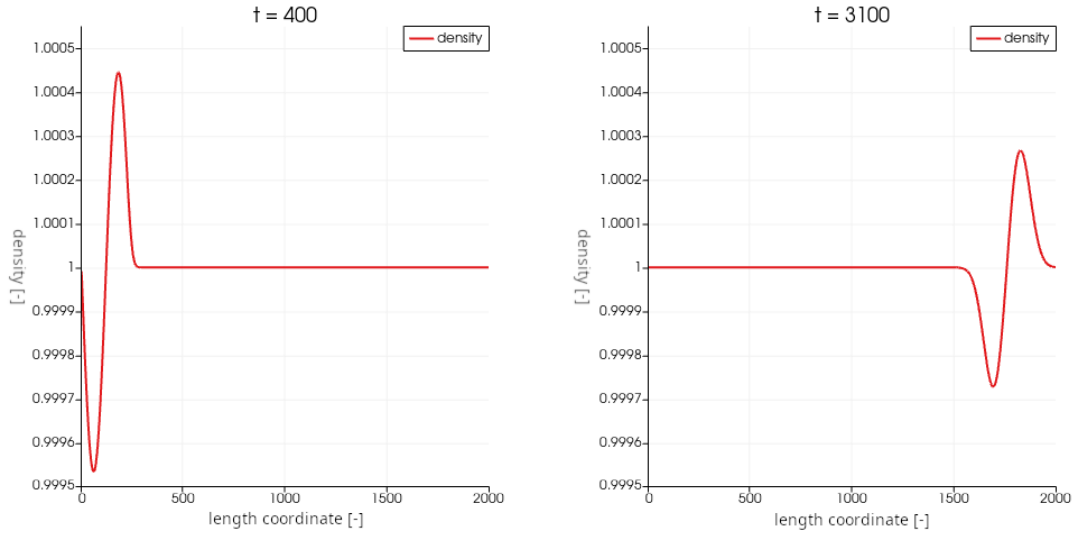


Figure 5.5: Density plots along the centre line of the travelling wave case with $\tau = 1.5$ at two different moments in times.

The travelling wave test case in Section 4.2 was tested at different simulation parameters finding that for low values of τ, the Mach number sensitivity was higher with a limit at around $\mathrm{Ma} = 0.1$, while higher values of τ allowed for Mach number up to around 0.3-0.5. When τ was kept close to $1/2$, to lower the viscosity, the wave travelled the entire length of the domain without any noticeable dissipation or distortion. A case where $\mathrm{Re} = 2000$, $\mathrm{Ma} = 0.05$ and subsequently $\tau = 0.51$ is shown in Figure 5.4. At higher values of τ, the increased viscosity is clearly noticeable through viscous dissipation. Figure 5.5 exhibits this behaviour for the parameters $\mathrm{Re} = 20$ and $\mathrm{Ma} = 0.05$ with resulting $\tau = 1.5$. Pushing further, the viscous damping becomes very strong at values of τ in the range 10-100 and at even higher values of $\tau > 1000$, the damping is paradoxically being reduced. (A plot of this is found in Figure B.1 in Appendix B.) This is because the large τ value enters the LBGK through the denominator in the BGK operator, which essentially kills the collision step of the LBM. This behaviour most likely hold no real physical relevance and such high relaxation times should generally be avoided.

5.2.3 Non-Reflecting BCs

The CBC is an established method for creating non-reflecting BCs [1], [7], [9] and it is a bit unclear why this method proved unstable when implemented in the code for this particular project (as mentioned in Section 3.2.6). The method has even been successfully applied to an Anti-Bounce-Back boundary before, albeit with the MRT model [7]. It does not seem that likely that the stability problems were caused primarily by this discrepancy alone, since divergence problems were encountered for a wide range of τ. As stated before, the only major stability problems in the LBGK single relaxation time model were found at τ close to $1/2$ and too large Mach numbers. Instead, one possible cause is errors in the floating point arithmetic. When subtracting or dividing two similarly sized floating point numbers, the precision is greatly compromised. In the extrapolation steps, there are both subtraction and

division operations that might introduce these types of errors, especially the estimation of gradients when the real gradients are zero. A small error in the estimation creates a non-zero gradient which serves as a feedback loop when the BC is reacting to the resulting ill-posed specification of values at the boundary. This theory would also explain why some success were had when the estimated time derivatives were given a lower limit, since this counteracts the feedback loop being entered at times when the boundary values should remain stable. One strategy that might alleviate or completely solve this problem would be to reformulate the CBC algorithm in such a way that the order of operations in the final expressions for the values entering the boundary condition are chosen to minimise floating point errors.

5.2.3.1 Alternate Strategy

Figure 5.6 shows that the strategy of zeroth order interpolation described in Section 3.2.6 did result in the reflected wave being mostly suppressed. It also proved stable for every flow case it was tested on.

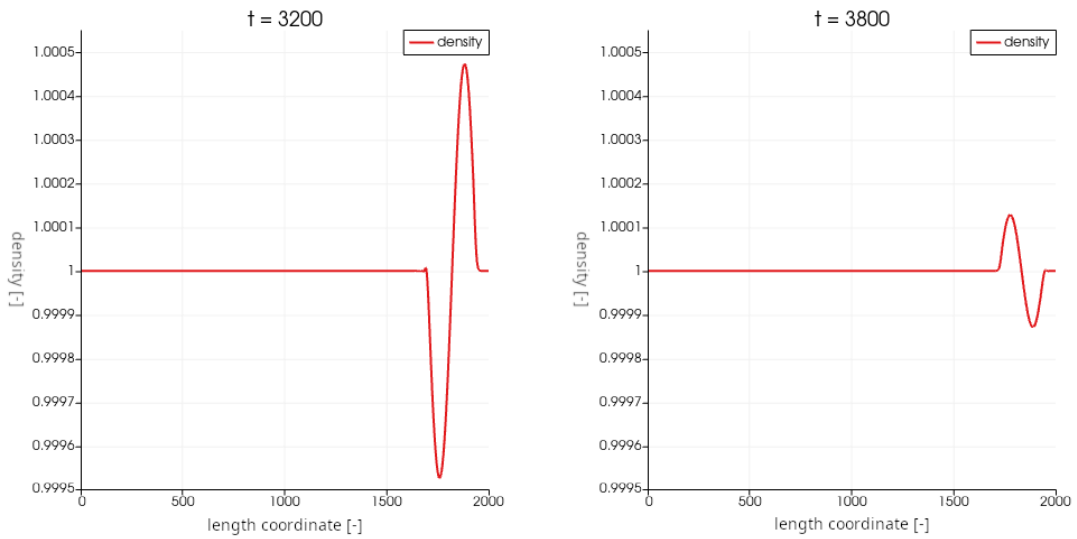


Figure 5.6: Density plots along the centre line of the travelling wave case with $\tau = 0.51$ at two different moments in times.

τ	Ma = 0.01	Ma = 0.02	Ma = 0.05	Ma = 0.1	Ma = 0.2
0.52	0.197	0.195	0.185	0.175	0.146
0.6	0.152	0.150	0.145	0.136	0.117
0.9	-0.018	-0.016	-0.011	-0.005	0.004
1.0	-0.075	-0.072	-0.064	-0.053	-0.034
1.5	-0.357	-0.349	-0.325	-0.288	-0.224
2.1	-0.697	-0.682	-0.638	-0.571	-0.452
2.5	-0.925	-0.904	-0.847	-0.760	-0.604
3.0	-1.207	-1.182	-1.109	-0.996	-0.794

Table 5.1: Optimal values for L_w to three decimals for which the wave in the test case from Section 4.2 was maximally suppressed during reflection at the outlet.

The refined strategy was used on the travelling wave test case, finding values for L_w which almost completely suppressed the reflected wave for a wide range of simulation parameters summarised in Table 5.1. For a fixed Mach number, the relation between τ and L_w is almost perfectly linear. Performing linear regression on the columns in Table 5.1 results in the line equation coefficients shown in Table 5.2.

	Ma = 0.01	Ma = 0.02	Ma = 0.05	Ma = 0.1	Ma = 0.2
a	-0.5662942	-0.5550212	-0.5220819	-0.4718248	-0.3794484
b	0.49168585	0.48335712	0.45795416	0.41956457	0.3447394

Table 5.2: Coefficients for linear regression made on the columns in Table 5.1, where $L_\mathrm{w} = a\tau + b$.

The rows in Table 5.2 are also almost linearly related to the Mach number, although not as close as the relation between between τ and L_w. Performing linear regression once again results in an empirical expression for L_w as a function of τ and Ma:

$$L_\mathrm{w}(\tau, \mathrm{Ma}) = (0.98073536\mathrm{Ma} - 0.5734699)\tau - 0.7715106\mathrm{Ma} + 0.49809503. \quad (5.1)$$

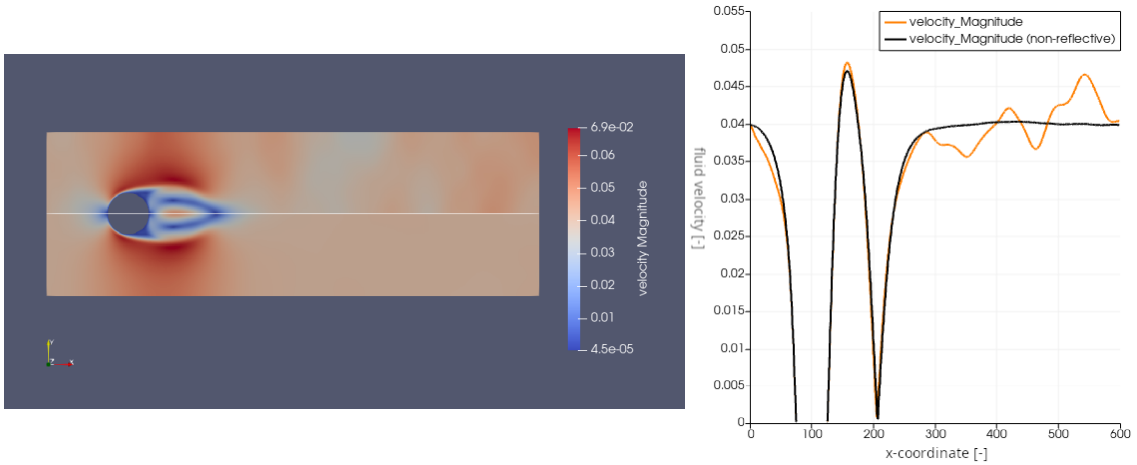
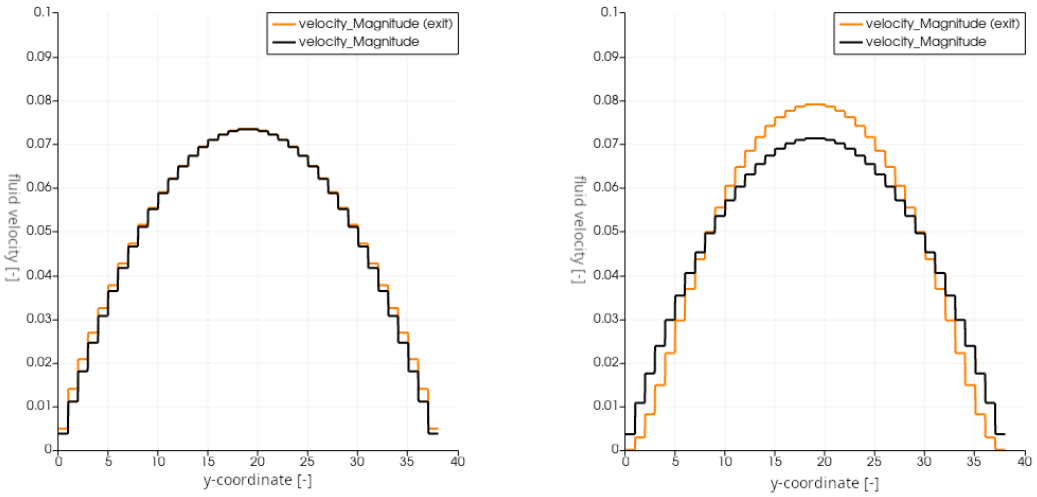


Figure 5.7: The 2D cylinder test case comparing the velocity field for the standard and non-reflective outlet at a moment in time where the wake is still building up. Left: 2D view split along the centre line with non-reflective on the lower half. Right: 1D view along the centre line. Both pictures show that most of the lingering waves of the standard case has exited the system in the non-reflective case.

Eq. (5.1) was implemented and confirmed to be working as intended and in a stable manner for the entire range of τ listed in Table 5.1 and for Mach numbers between $0.001 \leq \text{Ma} \leq 0.3$. Using values of $L_\text{w} < -1.2$ proved unstable, so the function in Eq. (5.1) was limited to never go below this value. This means that the non-reflectiveness deteriorates for $\tau > 3.0$, although in this regime, the viscous dissipation is already high enough to dampen out acoustic waves anyway. In Figure 5.7, the empirical relation in Eq. (5.1), is confirmed to be effective at suppressing reflection of waves generated at initialisation also in the cylinder test case from Section 4.4. In 3D, the formulation with varying L_w also turned out to be unstable, so the recommendation is to set $L_\text{w} = 0$, reducing to the zeroth order version of simply copying values to the boundary. This version remained stable, and simply due to all the additional space for waves to traverse in 3D compared to 2D, they seem to spread out and pose less of a problem. This thesis focused primarily on 2D cases, so further work would is needed for proper evaluation in 3D.

5.2.3.2 Accuracy of the New Non-Reflecting Outlet BC



(a) Original **(b)** Non-reflecting

Figure 5.8: Comparison of the velocity profile for Poiseuille flow at fully developed conditions (black curve) and at the exit (orange curve) for two verisions of the outlet.

When evaluating the accuracy of the new outlet on the Poisuille test case, the fully developed velocity profile is seen for a considerable length of the domain, but close to the outlet it is warped, indicating inaccuracies in the BC. The original Anti-Bounce-Back is not perfect either, though, as seen in the comparison made in Figure 5.8. *Dubois et al.* [16] also notices this and suggest replacing the Anti-Bounce-Back scheme with regular moving Bounce-Back for the oblique links at the boundary. These results were able to be reproduced, although the solution is not deemed as a viable replacement, since this version of the BC caused diverging solutions in most other cases it was evaluated.

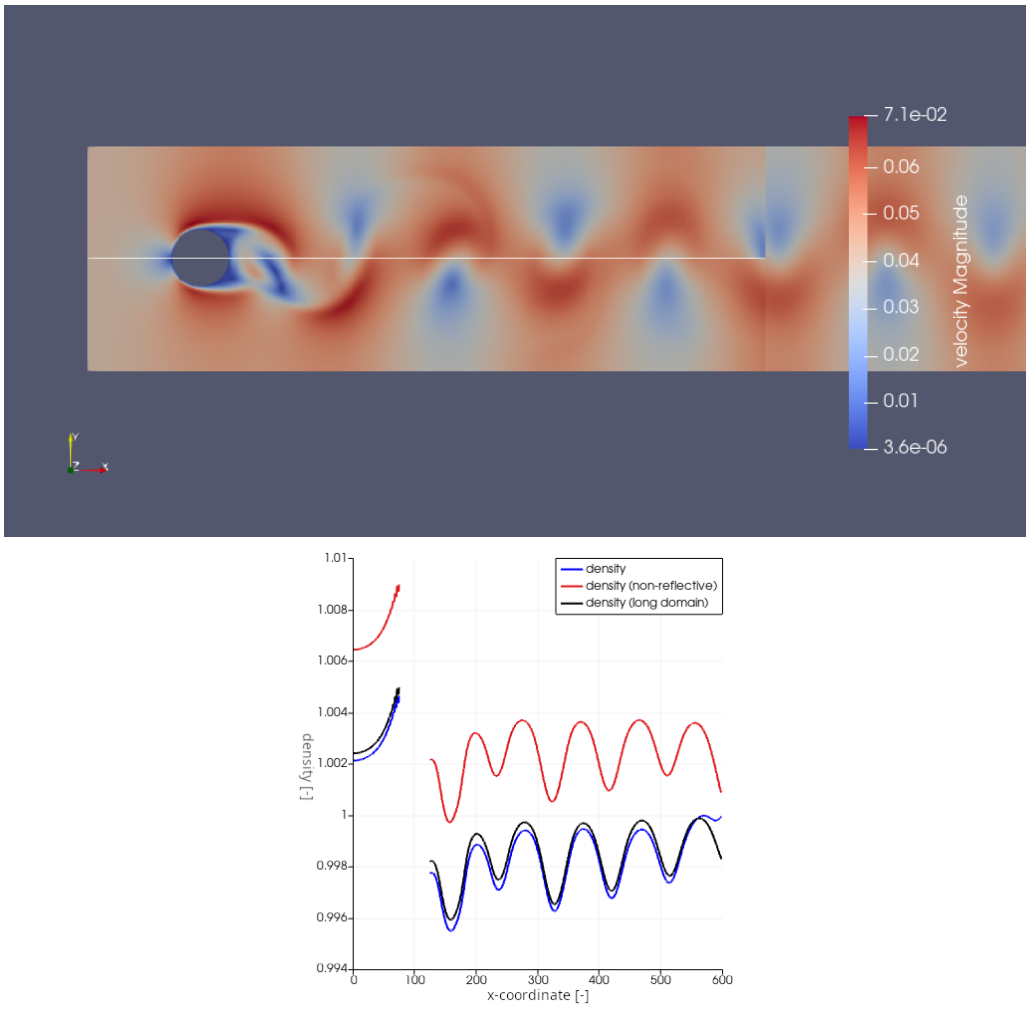


Figure 5.9: Comparison between non-reflecting outlet and standard outlet in the cylinder vortex shedding case. The goal is to match the long domain as closely as possible at the outlet. Above: The short domains are divided along the centre line with the non-reflecting at the bottom. The non-reflecting appears to match the long domain better at the outlet. Below: The density plotted along the centre line. Right at the outlet, the standard outlet does not match very well, but the non-reflecting has drifted and acquired an offset.

In Figure 5.9, the comparison is made between the reflecting and non-reflecting BCs for the cylinder vortex sheddeing test case from Section 4.4, showing that non-reflecting BCs seems to perform better at the outlet. There is a very noticeable drift, though, as discussed in Section 3.2.6.1. Implementing the modified expression for ρ_0 in Eq. (3.8) provided the best match with the long domain for the vortex shedding case, as shown in Figure 5.10. The non-reflective behaviour was also confirmed to remain intact for the travelling wave case with this modification.

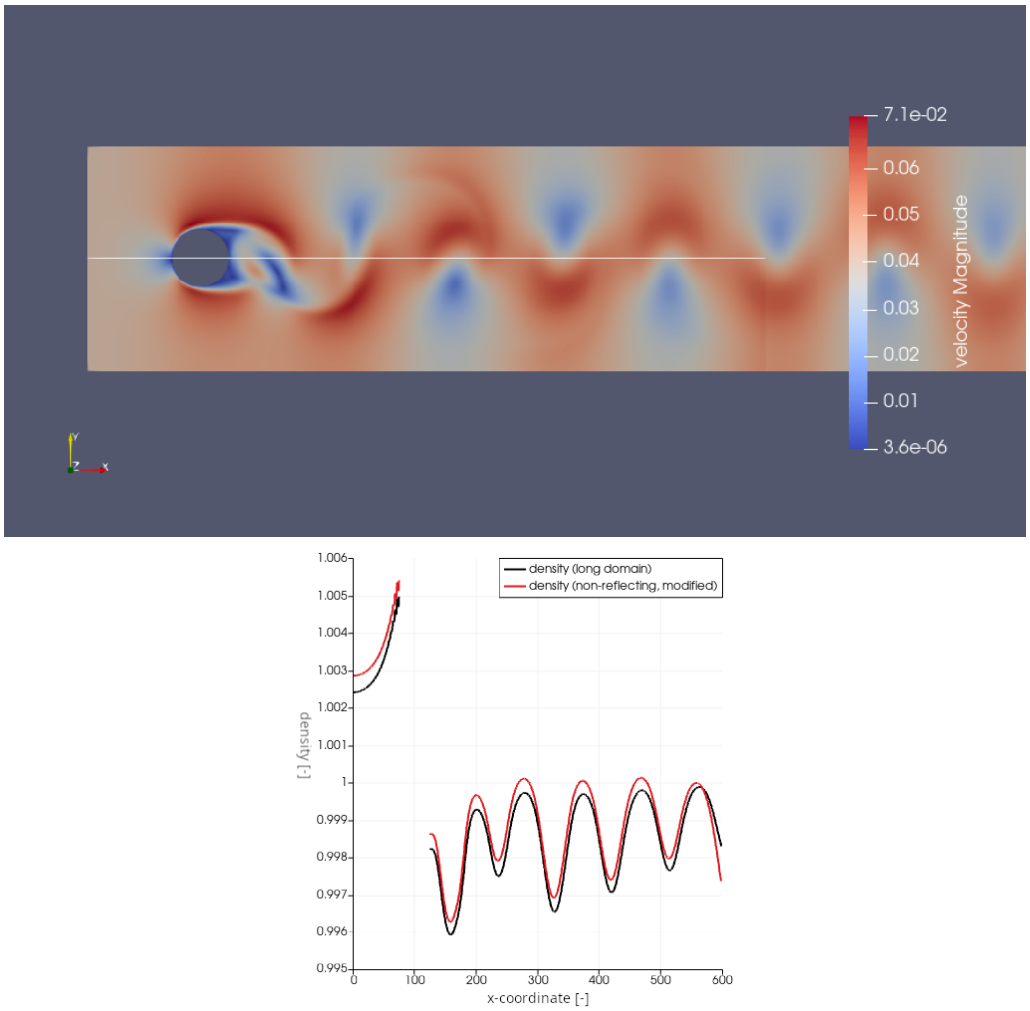


Figure 5.10: Comparison between the modified non-reflecting outlet and the long domain in the cylinder vortex shedding case. Above: The short domain is divided along the centre line with the non-reflecting at the bottom. Below: The density plotted along the centre line. The modified outlet matches the long domain better than any of the versions in Figure 5.9.

5.2.4 Isentropic Thermal Model

The results of the test case in Section 4.5 are shown in Figure 5.11. Looking at the velocity field, the vortex seem to be convected without major distortion or the vortex dissolving. There is noticeable viscous dissipation, however, and the flow parameters were chosen right at the edge of stability, preventing further reduction of the viscosity for that level of mesh refinement. The density field reveals that there are acoustic waves created at the initialisation. This serves as a good demonstration of the difficulty in properly initialising the full mesoscopic picture, as mentioned in Section 2.3.4, and that the equilibrium state is not always a good choice of initial state. It also hints at the energy, density and fluid velocity not being coupled in an entirely correct manner. However, the magnitude of the density fluctuations are small enough to consider the case incompressible and the density assumed constant.

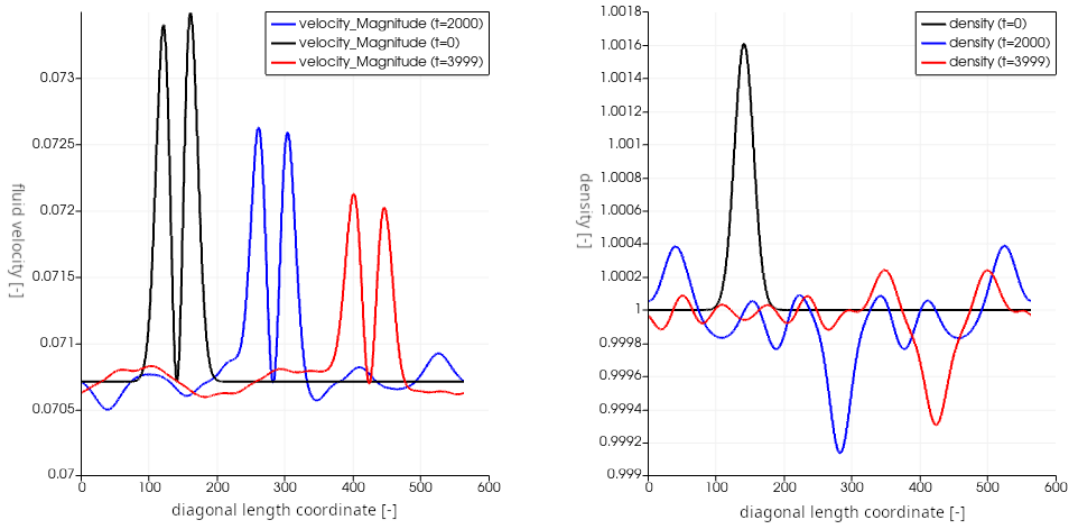


Figure 5.11: Velocity magnitude and density plots at different moments in for the isentropic vortex convection case. The plots are made along a diagonal line aligned with the free stream velocity. The vortex is preserved in shape but reducing in magnitude due to the viscosity with some small acoustic waves as noise in the picture. The density plot shows how the acoustic waves mostly dominate in the density field.

5.3 Further work

The results of this thesis work has gained plenty of insights to consider when using the Lattice-Boltzmann method of which many point to interesting extensions for the method. First of all, the potential for GPU parallelisation could be explored. The MRT or TRT models are also interesting to look into, to allow for higher Re while keeping the required grid refinement in check for stable simulations. Further evaluation of thermal models, such as the DDF model, are of great interest as well. Finally, reformulations of the discretisation along with other proposed models are of interest to unlock the capability of simulating high Mach number flows.

6
Conclusion

The results of this master's thesis can be used to draw the following concluding remarks:

- The standard LBM is an elegant reformulation of the fluid dynamics and performs well within its limitations.
- One of the very promising features is the potential for parallel computing on the GPU.
- To increase the capability of the method, extensions, of which there are many, can be used.
- The method is has the stability condition of $\tau > 1/2$. Close to this limit, the Mach number limit is stricter.
- The link-wise BCs are simple to implement but not always accurate enough.
- The LBM does not suffer from any noticeable numerical dissipation, although it is always viscous, which makes it very difficult to replicate inviscid flow conditions.
- Even for a completely known macroscopic state, initialisation of a simulation is a challenge, since there are multiple possible mesoscopic states corresponding to a single macroscopic one.
- The method in its standard form is weakly compressible, making it suitable for simulating acoustics. This might also be negative since unwanted acoustic noise is easily introduced at initialisation.
- It is possible to create non-reflecting BCs which allow acoustic waves to smoothly exit the simulation domain.

Bibliography

[1] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, *The Lattice Boltzmann Method, Principles and Practice*. Springer, 2017.

[2] J. D. A. Jr., *Modern Compressible Flow, With Historical Perspective*, 3rd ed. McGraw-Hill, 2003.

[3] D. V. Schroeder, *An Introduction to Thermal Physics*. Oxford University Press, 2021.

[4] Wikipedia contributors, *N-sphere — Wikipedia, the free encyclopedia*, [Online; accessed 27-March-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=N-sphere&oldofid=1213963808.

[5] L. Råde and B. Westergren, *Beta, Mathematics Handbook*, 2nd ed. Studentlitteratur, 1993.

[6] E. Buckingham, "On physically similar systems; illustrations of the use of dimensional equations," *Phys. Rev.*, vol. 4, pp. 345–376, 4 Oct. 1914. DOI: 10.1103/PhysRev.4.345. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.4.345.

[7] S. Izquierdo and N. Fueyo, "Characteristic nonreflecting boundary conditions for open boundaries in lattice boltzmann methods," *Phys. Rev. E*, vol. 78, p. 046 707, 4 Oct. 2008. DOI: 10.1103/PhysRevE.78.046707. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.78.046707.

[8] K. Li and C. Zhong, "A lattice boltzmann model for simulation of compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 77, no. 6, pp. 334–357, 2015. DOI: https://doi.org/10.1002/fld.3984. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.3984. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.3984.

[9] T. Poinsot and S. Lelef, "Boundary conditions for direct simulations of compressible viscous flows," *Journal of Computational Physics*, vol. 101, no. 1, pp. 104–129, 1992, ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(92)90046-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999192900462.

[10] Anaconda, Inc., *User manual — numba*, [Online; accessed 11-June-2024], 2024. [Online]. Available: https://numba.readthedocs.io/en/stable/user/index.html.

[11] NVIDIA Corporation, *Cuda toolkit - free tools and training*, [Online; accessed 11-June-2024], 2024. [Online]. Available: https://developer.nvidia.com/cuda-toolkit.

[12] Python Software Foundation, *Multiprocessing — process-based parallelism*, [Online; accessed 11-June-2024], 2024. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html.

[13] L. S. Avila, U. Ayachit, S. Barré, *et al.*, *The VTK User's Guide*, 11th ed. Kitware, Inc., 2010.

[14] T. von Kármán, *Aerodynamics, Selected Topics in the Light of Their Historical Development*, 2nd ed. Dover, 2004.

[15] K. Masatsuka, *I do like CFD, VOL.1, Governing Equations and Exact Solutions*, 2nd ed. Katate Masatsuka, 2013.

[16] F. Dubois, P. Lallemand, and M. M. Tekitek, "On anti bounce back boundary condition for lattice boltzmann schemes," *Computers & Mathematics with Applications*, vol. 79, no. 3, pp. 555–575, 2020, ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2019.03.039. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0898122119301646.

A

Derivations and Calculations

In this appendix, full derivations and calculations deemed too lengthy or detracting from the primary focus to include in the main text are presented.

A.1 Hermite Polynomials to Second Order

The generating formula for the d-dimensional Hermite polynomial of n-th order was presented in Eq. (2.43):

$$\boldsymbol{H}^{(n)}(\boldsymbol{x}) = (-1)^n \frac{1}{\omega(\boldsymbol{x})} \boldsymbol{\nabla}^{(n)} \omega(\boldsymbol{x}), \tag{A.1}$$

where

$$\omega(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}} \mathrm{e}^{-\boldsymbol{x}^2/2}. \tag{A.2}$$

The first two partial derivatives of $\omega(\boldsymbol{x})$ in Eq. (A.1) are:

$$\begin{aligned} \nabla^{(1)}_{\alpha} \omega(\boldsymbol{x}) &= \frac{\partial}{\partial x_\alpha} \omega(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}} \mathrm{e}^{-\boldsymbol{x}^2/2} \cdot (-x_\alpha) = -x_\alpha \omega(\boldsymbol{x}), \\ \nabla^{(2)}_{\alpha\beta} \omega(\boldsymbol{x}) &= \frac{\partial}{\partial x_\alpha} \frac{\partial}{\partial x_\beta} \omega(\boldsymbol{x}) = \partial_\alpha(-x_\beta \omega(\boldsymbol{x})) = -x_\beta \partial_\alpha \omega(\boldsymbol{x}) - \omega(\boldsymbol{x}) \partial_\alpha x_\beta \\ &= (-x_\beta)(-x_\alpha \omega(\boldsymbol{x})) - \omega(\boldsymbol{x}) \delta_{\alpha\beta} = (x_\alpha x_\beta - \delta_{\alpha\beta}) \omega(\boldsymbol{x}). \end{aligned} \tag{A.3}$$

Note that since $\boldsymbol{x}^2 = \sum_{\alpha=1}^{d} x_\alpha^2$, the partial derivative with respect to x_α is

$$\partial_\alpha \boldsymbol{x}^2 = 2x_\alpha. \tag{A.4}$$

Below, the HPs up to second order ($n = 0, 1, 2$) are computed:

$$H^{(0)}(\boldsymbol{x}) = (-1)^0 \frac{1}{\cancel{\omega(\boldsymbol{x})}} \nabla^{(0)} \cancel{\omega(\boldsymbol{x})} = 1, \tag{A.5a}$$

$$H^{(1)}_{\alpha}(\boldsymbol{x}) = (-1)^1 \frac{1}{\omega(\boldsymbol{x})} \nabla^{(1)}_{\alpha} \omega(\boldsymbol{x}) = x_\alpha, \tag{A.5b}$$

$$H^{(2)}_{\alpha\beta}(\boldsymbol{x}) = (-1)^2 \frac{1}{\omega(\boldsymbol{x})} \nabla^{(2)}_{\alpha\beta} \omega(\boldsymbol{x}) = x_\alpha x_\beta - \delta_{\alpha\beta}. \tag{A.5c}$$

A.1.1 HP Series Expansion of the Equilibrium Distribution

The HP series expansion for the equilibrium distribution from Eq. (2.46) truncated to second order and with the substitution $\boldsymbol{\eta} = (\boldsymbol{\xi} - \boldsymbol{u})/\sqrt{\theta}$ in the $\boldsymbol{a}^{(n),\mathrm{eq}}$ coefficient integral reads

$$\begin{aligned} \hat{f}^{\mathrm{eq}}(\rho, \boldsymbol{u}, \theta, \boldsymbol{\xi}) &= \omega(\boldsymbol{\xi}) \sum_{n=0}^{2} \frac{1}{n!} \boldsymbol{a}^{(n),\mathrm{eq}}(\rho, \boldsymbol{u}, \theta) \cdot \boldsymbol{H}^{(n)}(\boldsymbol{\xi}), \\ \boldsymbol{a}^{(n),\mathrm{eq}}(\rho, \boldsymbol{u}, \theta) &= \rho \int \omega(\boldsymbol{\eta}) \boldsymbol{H}^{(n)} \left(\sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}\right) \mathrm{d}^d \eta. \end{aligned} \tag{A.6}$$

To compute the integrals, which share some similarities with the one in the derivation of the Maxwell speed distribution in Section 2.2.3.7, the following results will be of use:

- For a spherically symmetric integrand (such as $\mathrm{e}^{-\boldsymbol{x}^2}$) integrated over the entire d-dimensional $\boldsymbol{x}$-space, the substitution from Eq. (2.24) can be performed:

$$\mathrm{d}^d x = S_{d-1} r^{d-1} \, \mathrm{d}r, \tag{A.7}$$

  where $0 \leq r = |\boldsymbol{x}| < \infty$ and

$$S_{d-1} = \frac{2\pi^{d/2}}{\Gamma(\frac{d}{2})}. \tag{A.8}$$

- For a separable, sufficiently well behaved function

$$f(\boldsymbol{x}) = g(x_\alpha) h(x_1, \ldots, x_{\alpha-1}, x_{\alpha+1}, \ldots, x_d), \tag{A.9}$$

  where $g(x_\alpha)$ is an odd function (such as $x_\alpha^{n_{\mathrm{odd}}}$, where n_{odd} is an odd integer), any integral over f spanning $-\infty < x_\alpha < \infty$ evaluates to zero:

$$\int f(\boldsymbol{x}) \, \mathrm{d}^d x = \cancelto{0}{\int_{-\infty}^{\infty} g(x_\alpha) \, \mathrm{d}x_\alpha} \int h \, \mathrm{d}^{d-1} x = 0. \tag{A.10}$$

- The slightly more general form of the definite integral in Eq. (2.28) found in the formula collection [5] can be written as

$$\int_0^\infty x^{d-1} \mathrm{e}^{-Cx^2} \, \mathrm{d}x = \frac{\frac{1}{2}\Gamma(\frac{d}{2})}{C^{d/2}}, \quad (C, d > 0). \tag{A.11}$$

- The HPs in Eqs. (A.5) with the argument $\boldsymbol{x} = \sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}$ reads:

$$H^{(0)} \left(\sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}\right) = 1, \tag{A.12a}$$

$$H_\alpha^{(1)} \left(\sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}\right) = \sqrt{\theta}\eta_\alpha + u_\alpha, \tag{A.12b}$$

$$\begin{aligned} H_{\alpha\beta}^{(2)} \left(\sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}\right) &= \left(\sqrt{\theta}\eta_\alpha + u_\alpha\right)\left(\sqrt{\theta}\eta_\beta + u_\beta\right) - \delta_{\alpha\beta} \\ &= \theta\eta_\alpha\eta_\beta + \sqrt{\theta}(\eta_\alpha u_\beta + \eta_\beta u_\alpha) + u_\alpha u_\beta - \delta_{\alpha\beta}. \end{aligned} \tag{A.12c}$$

First up, the coefficient $\boldsymbol{a}^{(0),\mathrm{eq}}$:

$$a^{(0),\mathrm{eq}} = \rho \int \omega(\boldsymbol{\eta}) H^{(0)} \left(\sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}\right) \mathrm{d}^d\eta \overset{\text{Eqs. (A.2) \& (A.12a)}}{\downarrow} = \frac{\rho}{(2\pi)^{d/2}} \int \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta$$

$$\overset{\text{Eq. (A.7)}}{\downarrow} = \rho \frac{S_{d-1}}{(2\pi)^{d/2}} \int_0^\infty r^{d-1} \mathrm{e}^{-\boldsymbol{r}^2/2} \, \mathrm{d}r \overset{\text{Eq. (A.11)}}{\downarrow} = \rho \frac{S_{d-1}}{(2\pi)^{d/2}} \frac{\frac{1}{2}\Gamma(\frac{d}{2})}{(\frac{1}{2})^{d/2}} \overset{\text{Eq. (A.8)}}{\downarrow} = \rho \frac{2\pi^{d/2}}{\Gamma(\frac{d}{2})} \frac{\frac{1}{2}\Gamma(\frac{d}{2})}{\pi^{d/2}} = \rho. \quad \text{(A.13)}$$

Secondly, the coefficient $\boldsymbol{a}^{(1),\mathrm{eq}}$:

$$a_\alpha^{(1),\mathrm{eq}} = \rho \int \omega(\boldsymbol{\eta}) H_\alpha^{(1)} \left(\sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}\right) \mathrm{d}^d\eta \overset{\text{Eqs. (A.2) \& (A.12b)}}{\downarrow} = \frac{\rho}{(2\pi)^{d/2}} \int \mathrm{e}^{-\boldsymbol{\eta}^2/2} \left(\sqrt{\theta}\eta_\alpha + u_\alpha\right) \mathrm{d}^d\eta$$

$$= \frac{\rho}{(2\pi)^{d/2}} \left(\sqrt{\theta} \underbrace{\int \eta_\alpha \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta}_{\to 0} + u_\alpha \int \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta \right) \quad \text{(A.14a)}$$

The integrand $\eta_\alpha \mathrm{e}^{-\boldsymbol{\eta}^2/2}$ is odd in η_α and by Eq. (A.10), the first integral on the last line vanishes. The remaining integral is identical to the one in Eq. (A.13):

$$a_\alpha^{(1),\mathrm{eq}} = \frac{\rho u_\alpha}{(2\pi)^{d/2}} \int \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta \overset{\text{Eq. (A.13)}}{\downarrow} = u_\alpha a^{(0),\mathrm{eq}} = \rho u_\alpha. \quad \text{(A.14b)}$$

Finally, the coefficient $\boldsymbol{a}^{(2),\mathrm{eq}}$:

$$a_{\alpha\beta}^{(2),\mathrm{eq}} = \rho \int \omega(\boldsymbol{\eta}) H_{\alpha\beta}^{(2)} \left(\sqrt{\theta}\boldsymbol{\eta} + \boldsymbol{u}\right) \mathrm{d}^d\eta$$

$$\overset{\text{Eqs. (A.2) \& (A.12c)}}{\downarrow} = \frac{\rho}{(2\pi)^{d/2}} \int \mathrm{e}^{-\boldsymbol{\eta}^2/2} \left(\theta\eta_\alpha\eta_\beta + \sqrt{\theta}(\eta_\alpha u_\beta + \eta_\beta u_\alpha) + u_\alpha u_\beta - \delta_{\alpha\beta}\right) \mathrm{d}^d\eta \quad \text{(A.15a)}$$

This integral is split into

$$\frac{\rho\theta}{(2\pi)^{d/2}} \int \eta_\alpha\eta_\beta \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta, \quad \text{(A.15b)}$$

$$\frac{\rho\sqrt{\theta}}{(2\pi)^{d/2}} \left(u_\beta \underbrace{\int \eta_\alpha \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta}_{\to 0} + u_\alpha \underbrace{\int \eta_\beta \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta}_{\to 0} \right) = 0, \quad \text{(A.15c)}$$

$$\frac{\rho(u_\alpha u_\beta - \delta_{\alpha\beta})}{(2\pi)^{d/2}} \int \mathrm{e}^{-\boldsymbol{\eta}^2/2} \, \mathrm{d}^d\eta \overset{\text{Eq. (A.13)}}{\downarrow} = (u_\alpha u_\beta - \delta_{\alpha\beta}) a^{(0),\mathrm{eq}} = \rho(u_\alpha u_\beta - \delta_{\alpha\beta}). \quad \text{(A.15d)}$$

The integrals in Eq. (A.15c) have the same integrands as the vanishing integral in Eq. (A.14a) and thus, sum to zero. In Eq. (A.15d), the integral is identical to the one in Eq. (A.13). The integral in Eq. (A.15b) evaluates to zero for any $\alpha \neq \beta$ by the argument for odd integrands. To keep the index notation, $\eta_\alpha\eta_\beta$ can be replaced

by $\delta_{\alpha\beta}\boldsymbol{\eta}^2/d$, which is equal to $\eta_\alpha\eta_\beta$ precisely when $\alpha=\beta$. The integation can then be completed:

$$\frac{\rho\theta}{(2\pi)^{d/2}}\int \eta_\alpha\eta_\beta \mathrm{e}^{-\boldsymbol{\eta}^2/2}\,\mathrm{d}^d\eta = \frac{\rho\theta}{(2\pi)^{d/2}}\frac{\delta_{\alpha\beta}}{d}\int \boldsymbol{\eta}^2\mathrm{e}^{-\boldsymbol{\eta}^2/2}\,\mathrm{d}^d\eta$$

$$\overset{\text{Eq. (A.7)}}{\downarrow} = \rho\theta\frac{S_{d-1}}{(2\pi)^{d/2}}\frac{\delta_{\alpha\beta}}{d}\int_0^\infty r^{d+1}\mathrm{e}^{-r^2/2}\,\mathrm{d}r \overset{\text{Eq. (A.11)}}{\downarrow} = \rho\theta\frac{S_{d-1}}{(2\pi)^{d/2}}\frac{\delta_{\alpha\beta}}{d}\frac{\frac{1}{2}\Gamma(\frac{d+2}{2})}{(\frac{1}{2})^{(d+2)/2}}$$

$$\overset{\Gamma(z+1)=z\Gamma(z)}{=} \rho\theta\frac{S_{d-1}}{(2\pi)^{d/2}}\frac{\delta_{\alpha\beta}}{d}\frac{\cancel{\frac{1}{2}}\Gamma(\frac{d}{2}+1)}{\cancel{(\frac{1}{2})(\frac{1}{2})^{d/2}}} \overset{\downarrow}{=} \rho\theta\frac{S_{d-1}}{\pi^{d/2}}\frac{\delta_{\alpha\beta}}{\cancel{d}}\frac{\cancel{d}}{2}\Gamma\left(\frac{d}{2}\right)$$

$$\overset{\text{Eq. (A.8)}}{\downarrow} = \rho\theta\frac{\cancel{2\pi^{d/2}}}{\cancel{\Gamma(\frac{d}{2})}}\frac{\delta_{\alpha\beta}}{\cancel{2\pi^{d/2}}}\cancel{\Gamma\left(\frac{d}{2}\right)} = \rho\theta\delta_{\alpha\beta}. \quad \text{(A.15e)}$$

Combining Eqs. (A.15e) and (A.15d) back into Eq. (A.15a) to finally finish the calculation of $\boldsymbol{a}^{(2),\mathrm{eq}}$:

$$a^{(2),\mathrm{eq}}_{\alpha\beta} = \rho(u_\alpha u_\beta - \delta_{\alpha\beta}) + \rho\theta\delta_{\alpha\beta} = \rho\Big(u_\alpha u_\beta + (\theta-1)\delta_{\alpha\beta}\Big). \quad \text{(A.15f)}$$

Before putting it all together, the HPs from Eqs. (A.5) and the coefficients from Eqs. (A.13)-(A.15) are neatly summarised below:

$$\begin{aligned} H^{(0)}(\boldsymbol{x}) &= 1, & a^{(0),\mathrm{eq}}(\rho,\boldsymbol{u},\theta) &= \rho \\ H^{(1)}_\alpha(\boldsymbol{x}) &= x_\alpha, & a^{(1),\mathrm{eq}}_\alpha(\rho,\boldsymbol{u},\theta) &= \rho u_\alpha \\ H^{(2)}_{\alpha\beta}(\boldsymbol{x}) &= x_\alpha x_\beta - \delta_{\alpha\beta}, & a^{(2),\mathrm{eq}}_{\alpha\beta}(\rho,\boldsymbol{u},\theta) &= \rho\Big(u_\alpha u_\beta + (\theta-1)\delta_{\alpha\beta}\Big). \end{aligned} \quad \text{(A.16)}$$

To arrive at the expression for $\hat{f}^{\mathrm{eq}}$ in Eq. (2.49), all terms from Eq. (A.16) are simply inserted into Eq. (A.6):

$$\begin{aligned} \hat{f}^{\mathrm{eq}}(\rho,\boldsymbol{u},\theta,\boldsymbol{\xi}) &= \omega(\boldsymbol{\xi})\sum_{n=0}^{2}\frac{1}{n!}\boldsymbol{a}^{(n),\mathrm{eq}}(\rho,\boldsymbol{u},\theta)\cdot\boldsymbol{H}^{(n)}(\boldsymbol{\xi}) \\ &= \omega(\boldsymbol{\xi})\left(\frac{1}{0!}a^{(0),\mathrm{eq}}H^{(0)}(\boldsymbol{\xi}) + \frac{1}{1!}a^{(1),\mathrm{eq}}_\alpha H^{(1)}_\alpha(\boldsymbol{\xi}) + \frac{1}{2!}a^{(2),\mathrm{eq}}_{\alpha\beta}H^{(2)}_{\alpha\beta}(\boldsymbol{\xi})\right) \\ &= \omega(\boldsymbol{\xi})\rho\left(1 + u_\alpha\xi_\alpha + \frac{1}{2}\Big(u_\alpha u_\beta + (\theta-1)\delta_{\alpha\beta}\Big)\Big(\xi_\alpha\xi_\beta - \delta_{\alpha\beta}\Big)\right). \end{aligned} \quad \text{(A.17)}$$

A.2 Velocity discretisation of the PDF

The HP series expansion coefficients for the PDF reads

$$\boldsymbol{a}^{(n)}(\boldsymbol{x},t) = \int f(\boldsymbol{x},\boldsymbol{c},t)\boldsymbol{H}^{(n)}(\boldsymbol{c})\,\mathrm{d}^d c. \quad \text{(A.18)}$$

Splitting f into an equilibrium and a non-equilibrium part ($f^{\mathrm{neq}} = f - f^{\mathrm{eq}}$) yields

$$\boldsymbol{a}^{(n)}(\boldsymbol{x},t) = \int f^{\mathrm{eq}}(\boldsymbol{x},\boldsymbol{c},t)\boldsymbol{H}^{(n)}(\boldsymbol{c})\,\mathrm{d}^d c + \int f^{\mathrm{neq}}(\boldsymbol{x},\boldsymbol{c},t)\boldsymbol{H}^{(n)}(\boldsymbol{c})\,\mathrm{d}^d c. \quad \text{(A.19)}$$

The first term is just the equilibrium coefficient $\boldsymbol{a}^{(n),\mathrm{eq}}$ and can be integrated exactly using the Gauss-Hermite quadrature rule as done in Eq. (2.53):

$$\boldsymbol{a}^{(n),\mathrm{eq}} = \sum_i \frac{w_i}{\omega(\boldsymbol{c}_i)} f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{c}_i, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i) = \sum_i f_i^{\mathrm{eq}}(\boldsymbol{x}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i). \tag{A.20}$$

The integrand in the second term is multiplied and divided by $\omega(\boldsymbol{c})$:

$$\int \frac{\omega(\boldsymbol{c})}{\omega(\boldsymbol{c})} f^{\mathrm{neq}}(\boldsymbol{x}, \boldsymbol{c}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}) \,\mathrm{d}^d c \tag{A.21}$$

Restating the Gauss-Hermite quadrature rule from Eq. (2.50)

$$\int \omega(\boldsymbol{c}) P^{(N)}(\boldsymbol{c}) \,\mathrm{d}^d c = \sum_{i=0}^{\tilde{n}^d - 1} w_i P^{(N)}(\boldsymbol{c}_i), \tag{A.22}$$

with the condition on the required number of abscissae

$$\tilde{n} \geq \frac{N+1}{2}. \tag{A.23}$$

Comparing Eqs. (A.21) and (A.22), reveals that the Gauss-Hermite rule can be applied if

$$\frac{f^{\mathrm{neq}}(\boldsymbol{x}, \boldsymbol{c}, t)}{\omega(\boldsymbol{c})} \boldsymbol{H}^{(n)}(\boldsymbol{c}) = P^{(N)}(\boldsymbol{c}). \tag{A.24}$$

$\boldsymbol{H}^{(n)}(\boldsymbol{c})$ is already a polynomial with a maximum degree of $n = 2$, since the moments considered are up to second order. The remaining factor of Eq. (A.24) is approximated using a polynomial series expansion:

$$\frac{f^{\mathrm{neq}}(\boldsymbol{x}, \boldsymbol{c}, t)}{\omega(\boldsymbol{c})} \approx P^{(N-n)}(\boldsymbol{c}). \tag{A.25}$$

The degree of this polynomial is limited by the number of abscissae required to integrate f^{eq}. The HP series expansion of the equilibrium distribution is truncated at second order giving a total of $N = 4$. From the condition in Eq. (A.23), $\tilde{n} \geq 5/2$ is obtained, but since $\tilde{n}$ is an integer, the lowest possible value is $\tilde{n} = 3$. To match this degree in Eq. (A.24), up to $N = 5$ may be chosen leaving a maximal degree of $N - n = 3$ to the polynomial series expansion in Eq. (A.25). This is one degree higher than the equilibrium distribution with an error of $\mathcal{O}(u^3)$, giving the error for the discretisation of f^{neq} as one order higher at $\mathcal{O}(u^4)$. Applying the rule then looks like

$$\begin{aligned} \int \frac{\omega(\boldsymbol{c})}{\omega(\boldsymbol{c})} f^{\mathrm{neq}}(\boldsymbol{x}, \boldsymbol{c}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}) \,\mathrm{d}^d c &\approx \int \omega(\boldsymbol{c}) P^{(N)}(\boldsymbol{c}) \,\mathrm{d}^d c = \sum_i w_i P^{(N)}(\boldsymbol{c}_i) \\ &= \sum_i w_i P^{(N-n)}(\boldsymbol{c}_i) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i) := \sum_i f_i^{\mathrm{neq}}(\boldsymbol{x}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i). \end{aligned} \tag{A.26}$$

Combining Eqs. (A.20) and (A.26) into Eq. (A.19) finally yields

$$\boldsymbol{a}^{(n)}(\boldsymbol{x}, t) \int f(\boldsymbol{x}, \boldsymbol{c}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}) \,\mathrm{d}^d c \approx \sum_i f_i(\boldsymbol{x}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i), \tag{A.27}$$

with

$$f_i(\boldsymbol{x}, t) \approx \frac{w_i}{\omega(\boldsymbol{c}_i)} f(\boldsymbol{x}, \boldsymbol{c}_i, t). \tag{A.28}$$

B

Figures and Tables

Additional figures and tables are found in this appendix.

Table B.1: Commonly used stencils in 1-3 dimensions with velocities $\boldsymbol{c}_i$ of magnitude $|\boldsymbol{c}_i|$, weight w_i and numbering i. This is an expanded version of table 2.1.

Stencil	$\boldsymbol{c}_i$	$\lvert\boldsymbol{c}_i\rvert$	w_i	i
D1Q3	(0)	0	$2/3$	0
	(1)	1	$1/6$	1
	(-1)	1	$1/6$	2
D2Q9	$(0,0)$	0	$4/9$	0
	$(1,0)$	1	$1/9$	1
	$(0,1)$	1	$1/9$	2
	$(-1,0)$	1	$1/9$	3
	$(0,-1)$	1	$1/9$	4
	$(1,1)$	$\sqrt{2}$	$1/36$	5
	$(-1,1)$	$\sqrt{2}$	$1/36$	6
	$(-1,-1)$	$\sqrt{2}$	$1/36$	7
	$(1,-1)$	$\sqrt{2}$	$1/36$	8
D3Q15	$(0,0,0)$	0	$2/9$	0
	$(1,0,0)$	1	$1/9$	1
	$(-1,0,0)$	1	$1/9$	2
	$(0,1,0)$	1	$1/9$	3
	$(0,-1,0)$	1	$1/9$	4
	$(0,0,1)$	1	$1/9$	5
	$(0,0,-1)$	1	$1/9$	6
	$(1,1,1)$	$\sqrt{3}$	$1/72$	7
	$(-1,-1,-1)$	$\sqrt{3}$	$1/72$	8
	$(1,1,-1)$	$\sqrt{3}$	$1/72$	9
	$(-1,-1,1)$	$\sqrt{3}$	$1/72$	10
	$(1,-1,1)$	$\sqrt{3}$	$1/72$	11
	$(-1,1,-1)$	$\sqrt{3}$	$1/72$	12
	$(-1,1,1)$	$\sqrt{3}$	$1/72$	13
	$(1,-1,-1)$	$\sqrt{3}$	$1/72$	14

Table B.1: Commonly used stencils in 1-3 dimensions with velocities $\boldsymbol{c}_i$ of magnitude $|\boldsymbol{c}_i|$, weight w_i and numbering i. This is an expanded version of table 2.1.

Stencil	$\boldsymbol{c}_i$	$\lvert\boldsymbol{c}_i\rvert$	w_i	i
D3Q19	$(0,0,0)$	0	$1/3$	0
	$(1,0,0)$	1	$1/18$	1
	$(-1,0,0)$	1	$1/18$	2
	$(0,1,0)$	1	$1/18$	3
	$(0,-1,0)$	1	$1/18$	4
	$(0,0,1)$	1	$1/18$	5
	$(0,0,-1)$	1	$1/18$	6
	$(1,1,0)$	$\sqrt{2}$	$1/36$	7
	$(-1,-1,0)$	$\sqrt{2}$	$1/36$	8
	$(1,0,1)$	$\sqrt{2}$	$1/36$	9
	$(-1,0,-1)$	$\sqrt{2}$	$1/36$	10
	$(0,1,1)$	$\sqrt{2}$	$1/36$	11
	$(0,-1,-1)$	$\sqrt{2}$	$1/36$	12
	$(1,-1,0)$	$\sqrt{2}$	$1/36$	13
	$(-1,1,0)$	$\sqrt{2}$	$1/36$	14
	$(1,0,-1)$	$\sqrt{2}$	$1/36$	15
	$(-1,0,1)$	$\sqrt{2}$	$1/36$	16
	$(0,1,-1)$	$\sqrt{2}$	$1/36$	17
	$(0,-1,1)$	$\sqrt{2}$	$1/36$	18

Table B.1: Commonly used stencils in 1-3 dimensions with velocities $\boldsymbol{c}_i$ of magnitude $|\boldsymbol{c}_i|$, weight w_i and numbering i. This is an expanded version of table 2.1.

Stencil	$\boldsymbol{c}_i$	$\lvert\boldsymbol{c}_i\rvert$	w_i	i
D3Q27	$(0,0,0)$	0	$8/27$	0
	$(1,0,0)$	1	$2/27$	1
	$(-1,0,0)$	1	$2/27$	2
	$(0,1,0)$	1	$2/27$	3
	$(0,-1,0)$	1	$2/27$	4
	$(0,0,1)$	1	$2/27$	5
	$(0,0,-1)$	1	$2/27$	6
	$(1,1,0)$	$\sqrt{2}$	$1/54$	7
	$(-1,-1,0)$	$\sqrt{2}$	$1/54$	8
	$(1,0,1)$	$\sqrt{2}$	$1/54$	9
	$(-1,0,-1)$	$\sqrt{2}$	$1/54$	10
	$(0,1,1)$	$\sqrt{2}$	$1/54$	11
	$(0,-1,-1)$	$\sqrt{2}$	$1/54$	12
	$(1,-1,0)$	$\sqrt{2}$	$1/54$	13
	$(-1,1,0)$	$\sqrt{2}$	$1/54$	14
	$(1,0,-1)$	$\sqrt{2}$	$1/54$	15
	$(-1,0,1)$	$\sqrt{2}$	$1/54$	16
	$(0,1,-1)$	$\sqrt{2}$	$1/54$	17
	$(0,-1,1)$	$\sqrt{2}$	$1/54$	18
	$(1,1,1)$	$\sqrt{3}$	$1/216$	19
	$(-1,-1,-1)$	$\sqrt{3}$	$1/216$	20
	$(1,1,-1)$	$\sqrt{3}$	$1/216$	21
	$(-1,-1,1)$	$\sqrt{3}$	$1/216$	22
	$(1,-1,1)$	$\sqrt{3}$	$1/216$	23
	$(-1,1,-1)$	$\sqrt{3}$	$1/216$	24
	$(-1,1,1)$	$\sqrt{3}$	$1/216$	25
	$(1,-1,-1)$	$\sqrt{3}$	$1/216$	26

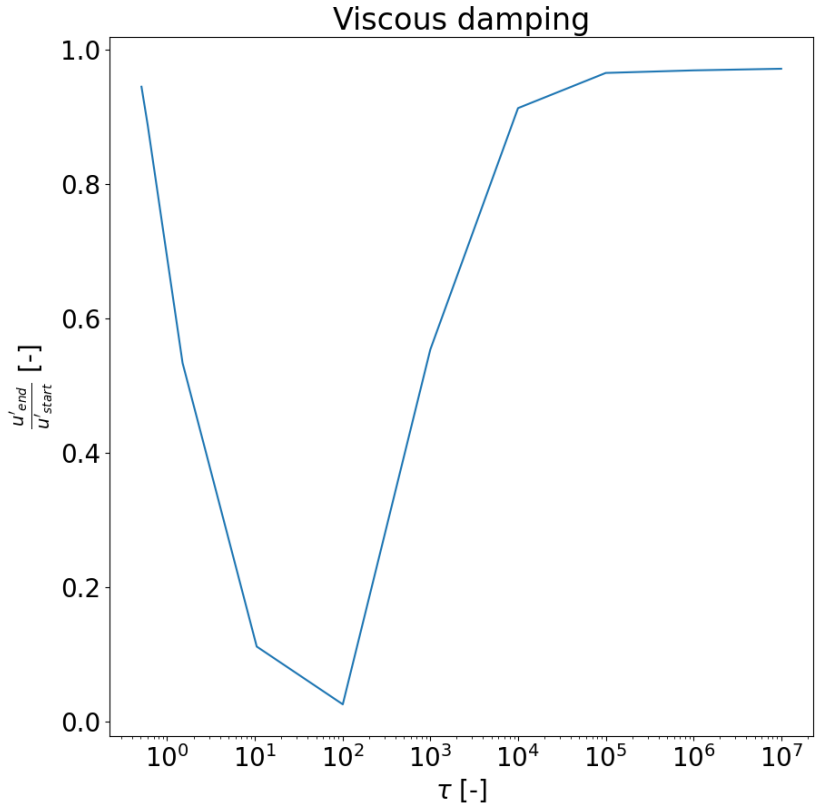


Figure B.1: The viscous damping of the travelling wave case from Section 4.2. The final amplitude u'_{end} is given in terms of τ and normalised by the initial amplitude u'_{start}. The maximal damping occurs at $\tau \sim 100$ before increasing again.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY