Parallel Lattice Boltzmann Method for Convection in Dendritic Solidification

Ivars Krastins

Centre for Numerical Modelling and Process Analysis,

School of Computing and Mathematical Sciences,

University of Greenwich

A thesis submitted in partial fulfilment of the requirements of the University of Greenwich for the degree of Doctor of Philosophy

October 2018

DECLARATION

I certify that the work contained in this thesis, or any part of it, has not been accepted in substance for any previous degree awarded to me, and is not concurrently being submitted for any degree other than that of Doctor of Philosophy being studied at the University of Greenwich. I also declare that this work is the result of my own investigations, except where otherwise identified by references and that the contents are not the outcome of any form of research misconduct.

.....	.....
Ivars Krastins	Dr. Andrew Kao
(Author)	(Supervisor)
.....	.....
Prof. Koulis Pericleous	Dr. Timothy Reis
(Supervisor)	(Supervisor)

DEDICATION

I would like to dedicate this thesis to my family and friends for their everlasting support throughout my PhD.

ABSTRACT

This work focuses on the development, validation and implementation of a parallel lattice Boltzmann method (LBM) for resolving fluid flow in multi-physics problems, such as alloy solidification, focusing on the effects on microstructure evolution. The literature has shown the importance of fluid flow in solidification as it affects the morphology and evolution of the growing dendrites. Because solute flow represents the most time-consuming part of the simulation, state-of-the-art computing allows for only a few cubic millimetres to be simulated, which is far less than the typical size of cast metal components. A purpose-built 3D LBM code is fully coupled to an external cellular automata (CA) solidification solver. It is run in parallel to achieve microstructure solidification on a macroscale. The performance analysis shows that the developed LBM flow solver is several times faster than the finite difference method currently used within the research group. The CA-LBM approach opens the possibility of component-scale microstructural simulations in a practical time frame. To properly model the physical boundaries, a new 3D moment-based boundary method for handling velocity and pressure in LBM is proposed. The capability of the numerical model is demonstrated by replicating experimentally observable physical phenomena during freckle formation in a casting.

ACKNOWLEDGEMENTS

I would like to say thank you to Dr. Andrew Kao for always being my first supervisor, even if not on paper, making sure that I learn from the experience and become a better researcher.

I am grateful to my second supervisor Prof. Koulis Pericleous for trusting in me and giving me the opportunity to conduct a PhD research here in Greenwich.

I would also like to thank my third supervisor Dr. Tim Reis for joining the team at a crucial point and contributing the missing piece of knowledge to my work.

I would like to acknowledge Dr. Matthaios Alexandrakis for preparing me for the PhD journey by going through it first. I was able to learn along the way and be ready for it.

Also, I would like to acknowledge PhD candidate Mr. Teddy Gan for brainstorming during my thesis planning stage.

Lastly, I would like to thank everyone else who made my PhD an unforgettable experience.

CONTENTS

FIGURES

TABLES

LISTINGS

NOMENCLATURE

Symbol	Description	Unit
α	thermal diffusivity	m^2s^{-1}
β	thermal expansion coefficient	K^{-1}
β_C	solute expansion coefficient	$wt.\%^{-1}$
$\mathbf{B}$	magnetic field	T
$\mathbf{c}_i$	discrete velocity vector	-
c	lattice speed	-
c_p	specific heat capacity	$Jkg^{-1}K^{-1}$
c_s	lattice sound speed	-
C	Courant number	-
C_i	collision operator in LGCA	-
D_C	solute diffusivity	-
ΔH	latent heat	Jkg^{-1}
Δt	time step	s/-
Δx	cell size	m/-
$\mathbf{F}$	force	N
$\mathbf{f}$	distribution function column vector	-
F_i	forcing term	-
f_i	particle distribution function	-
f_{liq}	liquid fraction	-
ϕ	general variable	-
g	gravitational acceleration	ms^{-2}
H	enthalpy	J/-
$\mathbf{I}$	identity matrix	-
i, j, k	Cartesian array indices	-
k	thermal conductivity	$Wm^{-1}K^{-1}$
κ	interface curvature	m^{-1}/-
L	domain length	m/-
Λ	*magic* parameter	-

Ma	Mach number	-
$\mathbf{M}$	transformation matrix	-
$\mathbf{m}$	distribution function in moment space	-
m_l	liquidus slope	K wt.%$^{-1}$
μ	dynamic viscosity	Pas
n_i	Boolean type function	-
N_i	ensemble average	-
Nx, Ny, Nz	calculation domain dimensions	-
ν	kinematic viscosity	m^{2}s^{-1}/-
Ω_i	collision operator	-
ω_i	relaxation parameter	-
P, p	pressure	Pa/-
Pe	Péclet number	-
$\mathbf{\Pi}$	momentum flux tensor	-
Π_{ij}	second order velocity moment	-
$\mathbf{Q}$	third order velocity tensor	-
Q_{ijk}	third order velocity moment	-
Ra	Rayleigh number	-
Re	Reynolds number	-
ρ	density	kgm^{-3}/-
$\mathbf{S}$	relaxation matrix	-
S_{ijkl}	fourth order velocity tensor	-
S	speed-up	-
St	Strouhal number	-
t	time	s/-
T	temperature	K/-
T^i	solid-liquid interface temperature	K/-
T_{uc}	undercooling temperature	K/-
τ	relaxation time	-
θ	angle	rad
$\mathbf{U}, \mathbf{u}$	velocity vector	ms^{-1}/-
U_x, U_y, U_z	velocity components	ms^{-1}/-
u, v, w	velocity components	ms^{-1}/-
w_i	lattice weights	-
$\mathbf{x}$	coordinate vector	m/-
x, y, z	Cartesian coordinates	m/-
ξ	velocity vector in phase space	m
ψ	velocity streamfunction	m^{2}s^{-1}
ZXZ	Euler angles	

Acronyms

BC	Boundary condition
BGK	Bhatnagar-Gross-Krook
CA	Cellular automata
CFD	Computational fluid dynamics
CFL	Courant-Friedrichs-Lewy
CPU	Central processing unit
CUDA	Compute unified device architecture
DC	Direct current
DdQq	d-dimensional q-velocity
DHC	Differentially heated cavity
FD(M)	Finite difference (method)
FE(M)	Finite element (method)
FHP	Frisch-Hasslacher-Pomeau
FLOPS	Floating point operations per second
FV(M)	Finite volume (method)
GPU	Graphics processing unit
HPP	Hardy-de Pazzis-Pomeau
HSD	He-Shan-Doolen
I/O	Input/output
KS	Kadanoff-Swift
LBE/M	Lattice Boltzmann equation/method
LGCA	Lattice gas cellular automata
MPI	Message-passing interface
MRT	Multiple relaxation time(s)
NS(E)	Navier–Stokes equation
OpenAcc	Open accelerators
OpenMP	Open multi-processing
PDF	Particle distribution function
PF	Phase-field
PISO	Pressure-implicit with splitting of operators
PIV	Particle image velocimetry
RAM	Random access memory
RBC	Rayleigh–Benard convection
SIMPLE	Semi-implicit method for pressure-linked equations
SRT	Single relaxation time
TESA	Thermoelectric solidification algorithm
TRT	Two relaxation time(s)
UDV	Ultrasound Doppler velocimetry

Chapter 1

INTRODUCTION

One of the ultimate goals for a sustainable future is to create alloys with better material properties. At a fundamental level, this is achieved by improving the microstructure of processes such as casting and welding. Tailoring the microstructure by modifying fluid flow during solidification is a key method to these goals. This can be achieved through the introduction of forces on the liquid, such as stirring or electromagnetically driven forces. Understanding what effect the modified flow has requires both experimental and numerical modelling techniques. However, performing real-time experiments to better understand the convection-driven solidification process is not straightforward. They require special facilities, conditions and attention, not to mention the economic costs. Apart from the technical difficulties of handling molten metals and monitoring the solidification process by means of neutron or X-ray radiographic techniques, a major challenge for *in situ* experiments remains capturing hydrodynamics. Excluding the invasive flow measuring methods, the most common technique is particle image velocimetry (PIV). Nevertheless, PIV, which is an optical visualisation method, requires

particles to be suspended and tracked in the system, which is not always wanted or possible. Ultrasound Doppler velocimetry (UDV) does not require X-rays or suspended particles. It uses ultrasound that reflects off acoustic inhomogeneities within the melt to calculate the local velocities. However, ultrasound is used in solidification for grain refinement, which interferes with dendritic growth and makes UDV an 'invasive' method, and hence unsuitable. The test samples can be analysed post-mortem, that is, after the experiment has finished and the metal has fully solidified, but the flow itself cannot be studied, only its consequences on the microstructure.

An alternative that provides insight into the evolution process of the growing dendrites and allows for a better controlled environment setup is numerical modelling. With the advances in computer capabilities over recent decades, numerical investigations have become commonplace in industrial research and research in general. Despite minimising the costs of experiments and providing a detailed insight in various physical processes, numerical simulations of multi-physics problems face their own constraints and challenges. A combination of, for example, the spatial and temporal resolution, accuracy, stability and available computer memory adds to the total solution time. The calculation domain decomposition and massive parallelisation is a common approach to decreasing the simulation runtime and obtaining results on a millimetre scale. Nonetheless, there is still a gap between the achievable size of the cutting edge numerical simulations and the industrial-size components.

1.1 Thesis overview

This thesis reports on the development of a numerical method that is capable of modelling fluid flow in complex and time-varying geometries and that can be coupled to other solvers to simulate, for example, multi-physics in dynamic multiscale systems in space and time, such as microstructure evolution during alloy solidification. One of the main difficulties in modelling multi-physics problems at the range of time and space scales required by solidification is the cost of simulation. In this respect, modern parallel computer architectures enable the simulation of processes that formerly required weeks to be completed in a few hours. The advantage is further highlighted via the use of numerical techniques and discretisation schemes that take full advantage of parallel solution approaches, for example the use of the lattice Boltzmann method (LBM) for the solution of fluid flow as opposed to the Navier–Stokes equation (NSE) approach. The large-scale simulations require the decomposition of the numerical domain and the use of parallel libraries to handle the inter-processor communications, thus part of thesis' focus is on the parallelisation aspect of the implementation.

While developing the numerical model, considerable effort is put into improving certain aspects of the method, in particular the way the domain boundaries are handled. By doing so, a new method for handling boundaries in 3D is proposed and later tested.

1.2 Thesis contributions

There currently exists within the research group a numerical algorithm TESA or *thermoelectric solidification algorithm* developed by Kao *et al.* [1]. TESA consists

of 4 physics solvers, heat and mass transport, solidification, electromagnetic field and fluid flow, of which the flow solver normally takes up most of the calculation time, 80–90 %. The objective of this work is to replace the current flow solver with a more efficient one, one that is the most suited to massive parallelisation. There are numerous methods with their merits and flaws used by researchers. The task is to choose the best one for the problems considered, in this case, handling fluid flow in complex and dynamic geometries. Because the focus of this work is specific, the general method can also be fine-tuned using application-specific settings. The set objectives and tasks can be summarised as the following two key research questions:

What is the most appropriate and efficient way to model fluid flow during microstructure solidification on a macroscale?

Can the numerical technique be improved to produce more accurate or stable results?

To answer these questions, the relevant literature needs to be reviewed to gain insight into the cutting edge research methods used to model dendritic solidification with convection in large-scale domains. After choosing the best candidate, the method needs to be studied in detail to assess the best setup and its suitability for the application in mind, and improved if necessary. The numerical method in question then needs to be validated against the appropriate benchmark cases before it can be deployed to solve complex scientific or industrial problems.

1.3 Thesis outline

Chapter 1 covers the background knowledge that led to the formation of this thesis. It provides the motivation of the work, and lists several research questions that this thesis seeks to answer. This chapter also contains the structure of the thesis.

Prior-art and state-of-the-art literature is reviewed in Chapter 2, covering topics on the importance of convection in microstructure solidification, employed modelling techniques and solutions to simulating large-scale multi-physics problems. Conclusions are drawn about the best practices to model convection in microstructure solidification on a macroscale.

Chapter 3 is dedicated to the LBM, describing it in detail. This chapter is an extension to the literature review covering the main aspects of the method and recognising parts that will be used in building the optimal model. An important part of the chapter is proposing a new novel 3D moment-based boundary method for the D3Q19 model which is an extension of the 2D Moment Method [2].

Chapter 4 describes the numerical algorithm developed to handle the fluid flow during multi-physics modelling. Both the serial and parallel implementations are described and tested and the performance analysis is conducted comparing different methods and different implementations of the LBM.

Chapter 5 presents the model validation results from various benchmark cases. This chapter also contains the large-scale results from using the developed method.

Conclusions and future work are provided in Chapter 6. The answers to the posed research questions are covered in the conclusions.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

In this chapter the relevant literature on the convection effect on microstructure solidification and the different modelling techniques used to simulate it are covered.

The macroscopic thermophysical properties of materials hugely depend on the solidification process of the liquid melt, which, however, depends on the solute and temperature redistribution around the evolving dendritic structures. The ability to predict the microstructure and hence the macroscopic properties of the solidified material is of great importance to the material processing industry and therefore requires comprehensive research to be conducted in the field of microstructure solidification both experimentally and numerically.

The following sections review important contributions by other investigators relevant to the goals of this work.

2.2 Convection effect on microstructure solidification

In both natural and industrial processes, free convection plays an important role and it cannot be neglected when creating numerical models. The effects of melt convection on dendritic growth have been investigated both numerically and experimentally.

In directional solidification convection affects the morphological stability of the solid-liquid interface, as numerically studied and experimentally observed by Noel *et al.* [3], Jamgotchian *et al.* [4] and Lan and co-workers [5–7].

In single component melts or pure metals, solidifying dendrites emit latent heat and create a thermal boundary layer around the tips. Convection can deform this layer by extracting away the heat and assisting growth or by supporting its build-up and preventing growth. Additionally, in multi-component systems, such as binary or ternary alloys, the solute gets ejected into the melt causing a shift in the phase diagram and modifying the solidification point temperature. Dendritic growth is affected by the way this extra concentration of the solute is dissipated. Convection is almost always present in solidification processes. One of the most common driving forces is gravity, which through density variations leads to natural convection. This type of flow can be introduced through thermo-solutal buoyancy or even through density variations between the solid and liquid giving floating or sinking grains. Buoyant forces either carry it away from the solid-liquid interface or do not let it escape and slow down the solidification process depending on the solute-solvent density ratio.

The first visualisation of the buoyancy effect on the dendrite spacing in di-

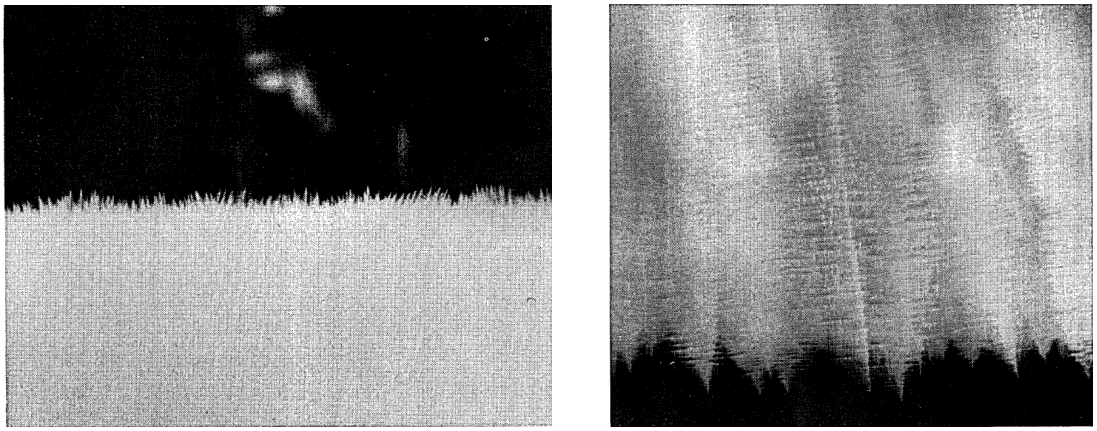


Figure 2.1: The interface and primary dendrite structures in upwards (left) and downwards (right) growth directions [8].

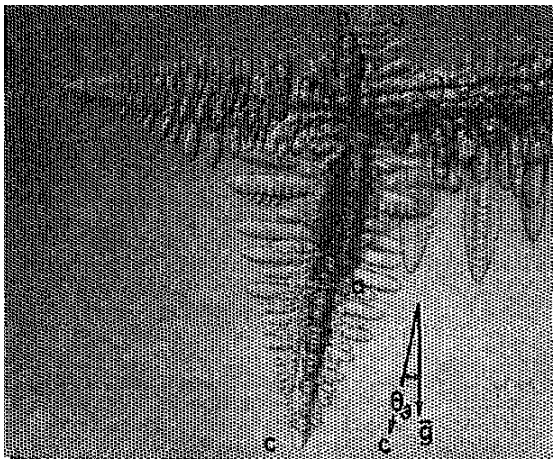


Figure 2.2: Growing dendrite complex affected by gravity [9].

rectional solidification was observed by Burden and Hunt [8]. They noticed that the primary dendrite spacing varies depending on growth direction with respect to gravity, see Figure 2.1. By reversing the thermal gradient, the average spacing coarsened from 0.5 mm to an order of magnitude higher values at 5 mm.

Glicksman and Huang [9] were able to observe the buoyancy effect on a semi-free dendrite growth configuration by using succinonitrile, which is transparent and has a low melting temperature, see Figure 2.2.

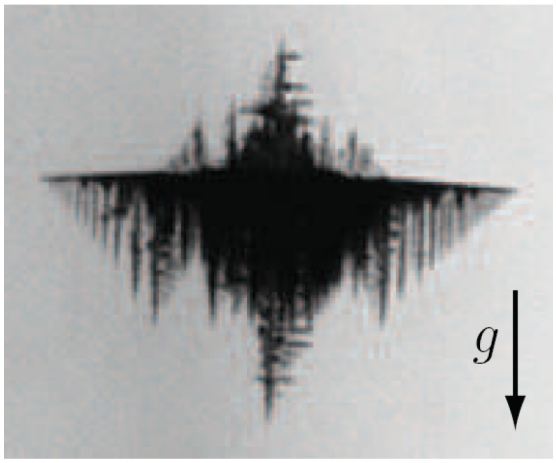


Figure 2.3: Settling of an Ammonium Chloride crystal by Beckermann [10].

Free growing crystals that are under the influence of gravity are difficult to capture. They must be settling down without rotation for the preferential and stunted growth to be observed. It was, however, achieved by Beckermann [10]. Figure 2.3 shows the settling of an Ammonium Chloride (NH_4Cl) crystal with clearly visible differences between the top and bottom sides of the falling equiaxed dendrite.

In directional solidification, the interaction between dendrite growth and convection can become quite intricate. Let us consider a binary alloy system in a quasi-3D differentially heated cavity, where the nucleation starts at the cold bottom wall. An experiment of such a setup has been carried out recently by Shevchenko and co-workers [11; 12]. First, the solidification process of a Ga-25wt.%In alloy under the influence of thermo-solutal convection in a Hele-Shaw cell was studied using X-ray radiography. Second, forced convection was superimposed by the means of a rotating disk of permanent magnets, and the changes to the dendritic structure were investigated. From their results displayed in Figure 2.4, several observations can be made. Gallium enriched plumes are being

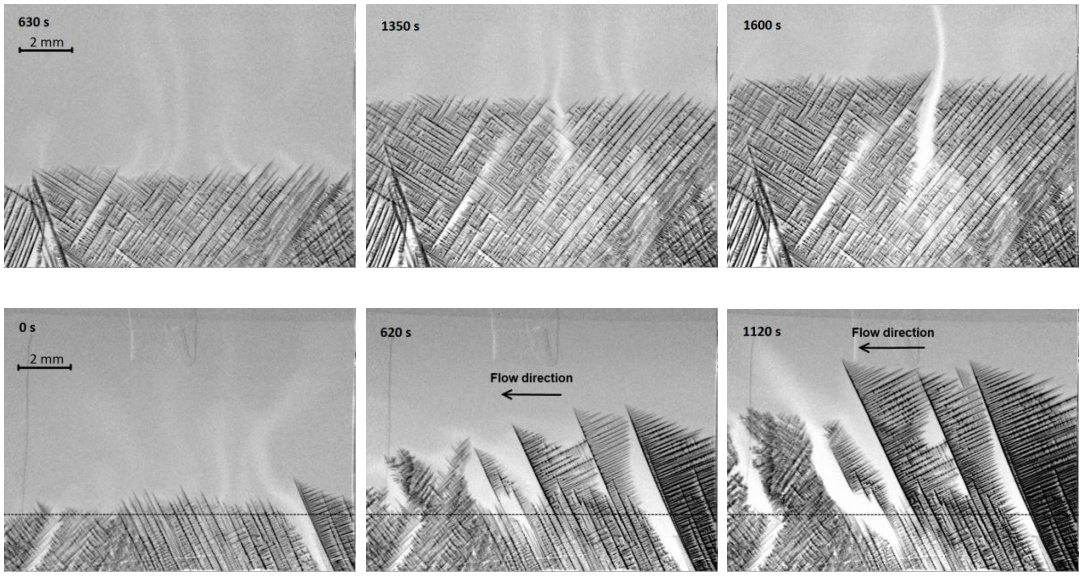


Figure 2.4: Solidification of a Ga-25wt.%In alloy. Natural convection (top) and introduced forced convection (bottom). The dashed line is showing the onset of the electromagnetic force driving the flow [12].

created in the liquid melt transporting the ejected light Gallium up, while in the interdendritic region these solute-rich plumes are creating segregation channels by remelting the already solidified crystals. These segregation channels solidify at a lower temperature and have a different material composition than the rest of the sample. In material science, they are called *freckles* and they are viewed as defects because they compromise the mechanical properties of materials. An example of this is the casting of single-crystal turbine blades using Nickel-based superalloys [13]. The introduced horizontal flow also restructures the segregation channels by eliminating the existing ones and forming new ones [12]. Another aspect to notice from Figure 2.4 is the change in primary dendritic arm spacing when the forced convection is superimposed. The spacing increases approximately 5 times in this case, but it is of course dependent on the magnitude of the external force. Experiments investigating the effect of the imposed fluid flow on the dendritic arm

spacing have been conducted by Steinbach and co-workers [14–16]. Other experiments studying the convection effect on the microstructure have been carried out showing the correlation between the dendrite tip growth velocity and the solute concentration ahead of the tip [17], convection effect on the chimney formation [18] and describing the forced convection effect on dendritic growth pointing out the damping of the local fluctuations of the solute concentration and once more confirming the previous observations of the preferential growth and the primary arm spacing [17; 19; 20].

A lot of research has been done numerically in order to develop a better understanding of the solidification process, namely, microstructure formation, segregation, convection, etc., in columnar [21–23] and equiaxed dendritic growth [24–35] and the transition from one to the other [36–39].

Diepers and Steinbach [21] were the first ones to numerically study the buoyancy effect on the dendritic arm spacing in directional solidification in 2D using the phase-field (PF) method and the control volume scheme for the fluid flow. They concluded that the primary spacing increases/decreases depending on the convection pattern which is crucially affected by the flow direction, see Figure 2.5.

To properly capture the interdendritic flow and investigate the mechanisms of microstructure growth phenomena under natural or forced convection, the use of 3D numerical models is required. Because the dendrites in 2D models are effectively plates that are blocking the solute transport, there is no flow connecting successive interdendritic regions. This was concluded by Yuan and Lee [40] who were the first ones to investigate a case study of columnar dendritic growth with convection in 3D. They used the Imperial College in-house open source software

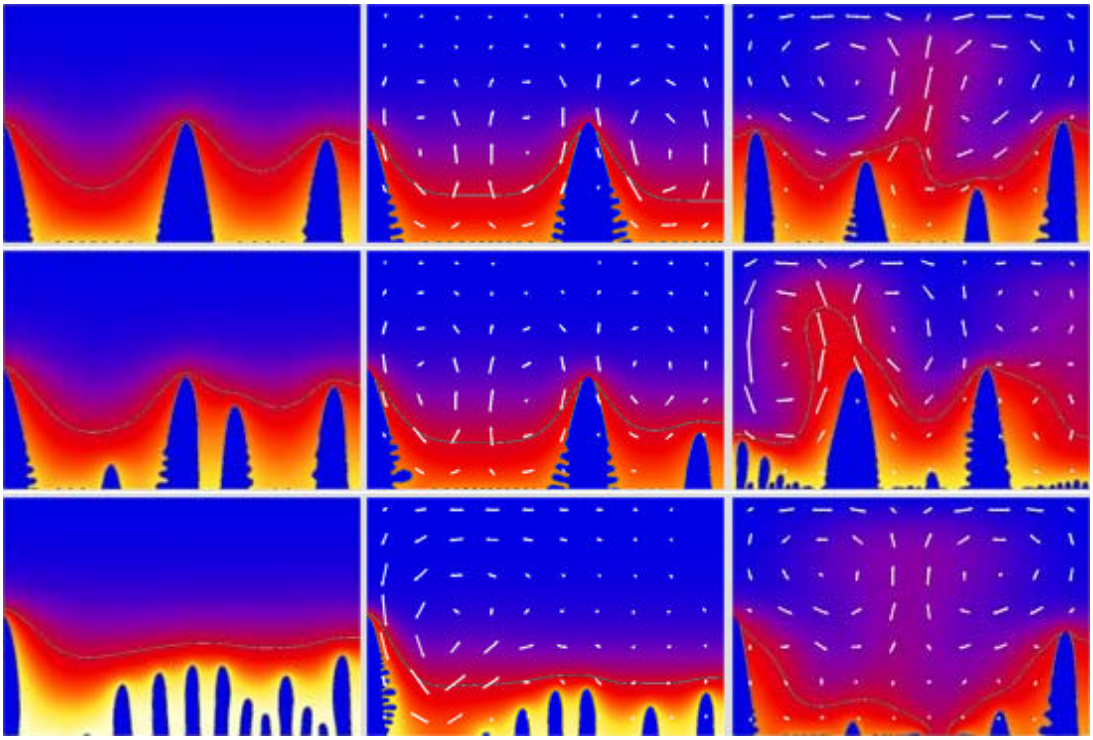


Figure 2.5: Convection effect on primary arm spacing during directional solidification in 2D. No convection (left), downward (middle) and upward buoyancy (right) captured at times $t = 27$ s (bottom), $t = 45$ s (middle), $t = 72$ s (top) [21].

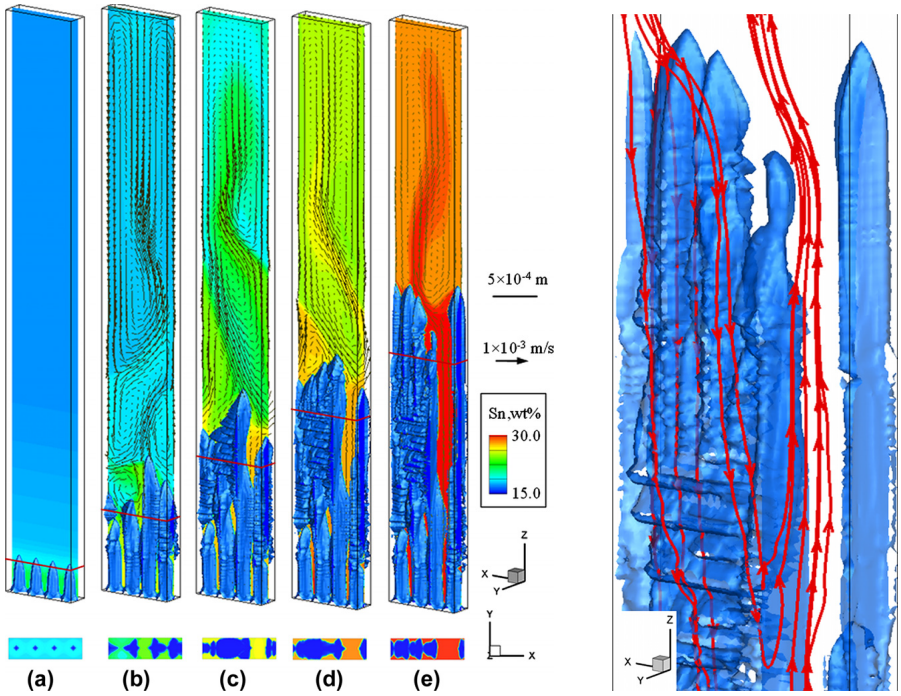


Figure 2.6: Evolution of the dendrite morphology showing Sn concentration profiles (left) and flow pattern (right) at Rayleigh number Ra=46.9 [42].

called µMatIC [41], extending it to include convection. Using the same software, Yuan and Lee studied the freckle initiation mechanism as a consequence of the Rayleigh number, Ra [42], see Figure 2.6. They were able to narrow down the region to Ra = 37.5−46.9 in which the onset of freckle formation occurs in the Pb-Sn alloy. For different alloys the critical Rayleigh number can vary, for example, for Ga–25wt.%In alloy Ra = 150 − 170 [43]. The results showed great promise when compared to the experimental data. Additionally, they concluded that a large density variation in the alloy encourages thermosolutal convection which is directly responsible for the formation of freckles, alongside with remelting, overgrowth and deflection of dendrites.

There is general interest in predicting and controlling the microstructure solid-

ification mechanisms that later affect the macroscopic thermophysical properties of materials. For instance, the effect of thermoelectric magnetohydrodynamics on microstructure evolution has been investigated numerically by Kao *et al.* [44]. In their work, a DC magnetic field is applied externally and it interacts with the thermoelectric currents formed by the temperature gradients at the interface of the growing dendrites creating an electromagnetic force which drives the interdendritic flow and alters the evolution of the dendrites, see Figure 2.7. It increases the primary arm spacing and causes preferential growth of secondary arms. In the melt, however, the solute plumes are being electromagnetically damped slowing down their formation and keeping growth steady which can be valuable for manipulating the microstructure and hence the material properties.

The magnetic field can also affect the morphology of freely growing dendrites, see Figure 2.8. This effect has been numerically investigated by Kao using the inhouse built parallel software TESA [1]. It stands for thermoelectric solidification algorithm and it has been successfully used for various problems in thermoelectric magnetohydrodynamics [45–53].

Several numerical models dealing with convection during solidification have been proposed over time, describing the effect of forced convection on the morphology of a freely growing crystal. The first model was developed by Beckermann *et al.* [24] who used a phase-field method with convection to model the undercooled crystal growth in 2D, see Figure 2.9. The change in the morphology of the dendrite is due to the thermal boundary layer formed by the extraction of the latent heat being reshaped by the fluid flow. A simplified conclusion can be drawn – the thinner the layer, the faster the growth and vice versa. Other works performed in 2D [25–29; 55] have arrived at the same conclusion.

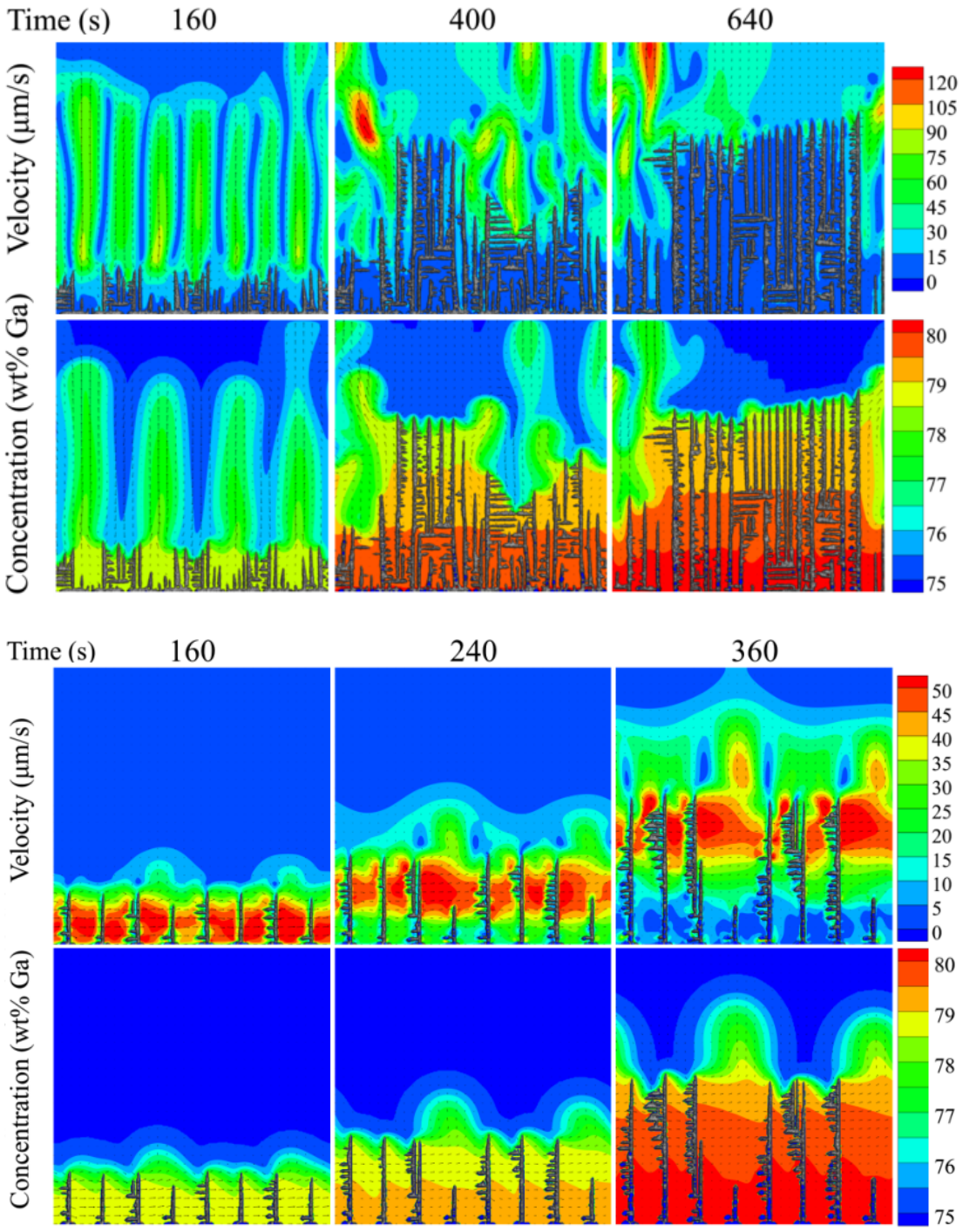


Figure 2.7: Transient evolution of microstructure under the influence of natural convection (top) and an external 1 tesla DC magnetic field (bottom) [44].

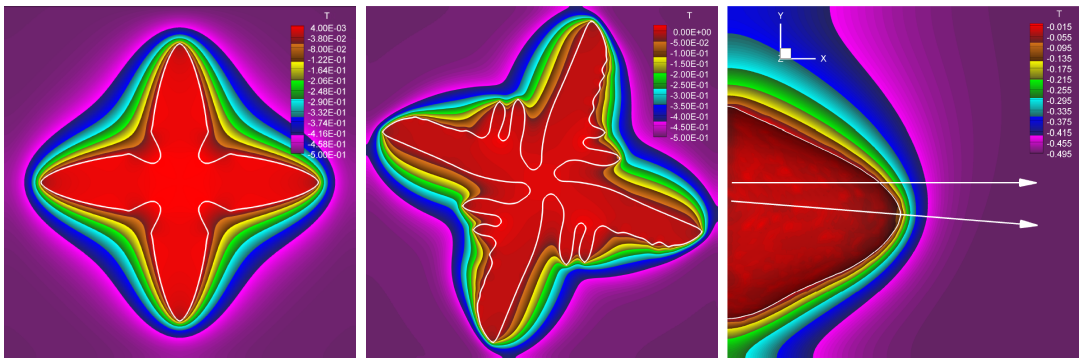


Figure 2.8: Effect of the magnetic field on crystal morphology. No magnetic field (left), $\mathbf{B} = 20$ T in 2D (middle) and in 3D (right) showing deflection of the dendrite tip from preferred direction of growth [54].

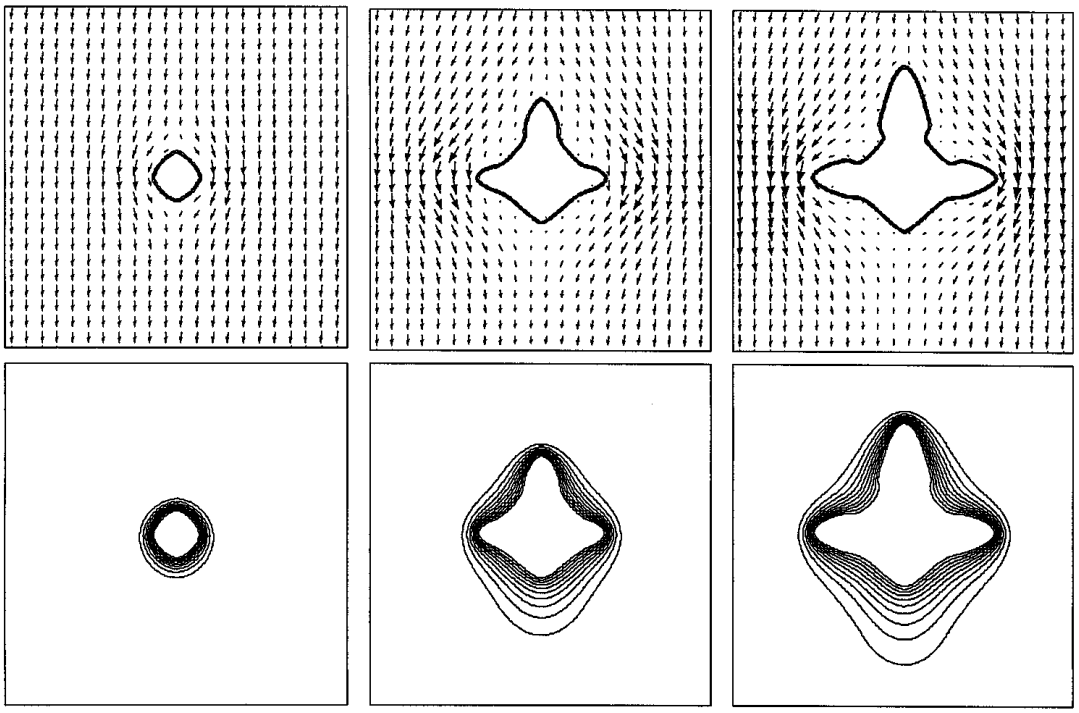


Figure 2.9: Undercooled crystal growth with convection in 2D showing the preferential and stunted growth. Velocity field (top) and isotherms (bottom) [24].

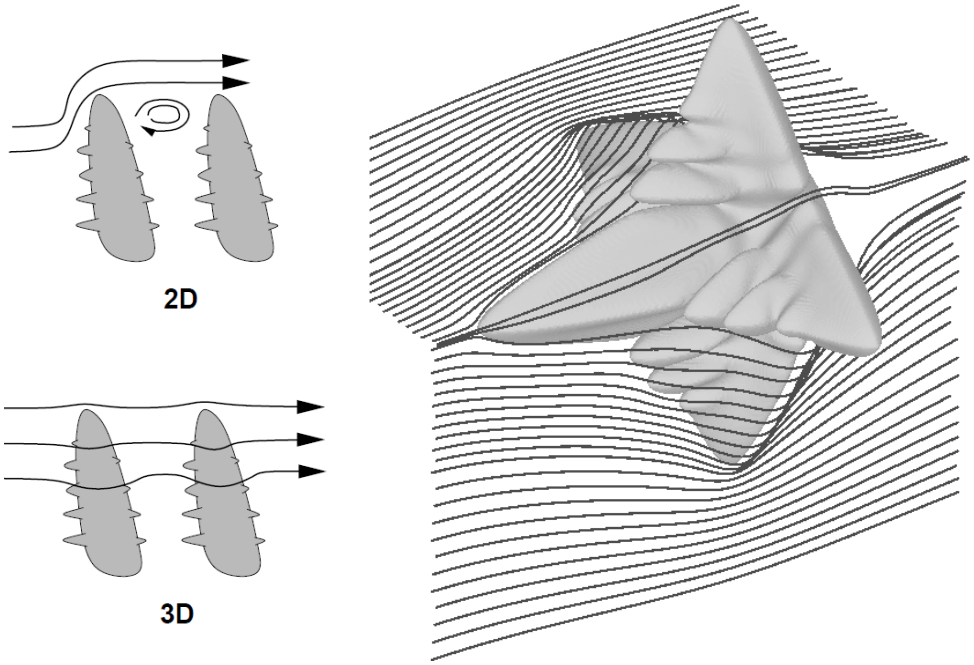


Figure 2.10: Undercooled crystal growth with convection in 3D. Schematic drawing showing the flow pattern in 2D and 3D (left) and the complex shape of the evolving dendrite showing streamtraces (right) [30].

The first 3D phase-field model of a freely growing crystal in fluid flow was developed by Jeong *et al.* [30], followed by several other investigations in the following years [31–35]. There is a fundamental difference between 2D and 3D models – there is more freedom for the flow to pass around the dendrite and carry away the extracted heat or solute boosting the growth of the crystal as shown in Figure 2.10. This difference for constrained and unconstrained dendritic growth has been investigated and quantified by Yuan and Lee [40]. They concluded that the 3D models offer better representation of the solidifying microstructure allowing the melt flow to wrap around the dendrites and not blocking it as it often happens in 2D cases.

2.3 LBM in convection-driven solidification

Since its beginning, the LBM has seen several modifications and improvements in terms of stability and accuracy that enable modelling of turbulent flow, flow in porous media, multi-component, multi-phase and contaminant complex flows and phase-change [56–58]. The simplified treatment of the boundaries in complicated geometries is one of the main reasons why the LBM is a natural choice when it comes to modelling melt flows during dendritic solidification. The other reason for using the LBM, which will be discussed in the next section, is the ease with which the method can be parallelised in comparison to the more traditional finite difference (FD), finite volume (FV) or finite element (FE) method computational fluid dynamics (CFD) techniques.

The first introduction of the LBM to the field of solidification was made by Jiaung *et al.* [59] who used the enthalpy method to solve the phase change problem, and by Miller and co-workers [60; 61] who used the phase-field method to capture the liquid-solid phase transition in the Ga melting problem as well as the anisotropic crystal growth in an undercooled melt with convection. It was already predicted back then that the LBM might become a powerful tool for phase-change problems with complicated boundary conditions due to the method's simplicity, stability and inherent parallelism [59].

Chatterjee and Chakraborty also employed the LBM in combination with the enthalpy method to simulate a generic laser surface melting process in 3D using the D3Q19 lattice [62] and convection-diffusion transport in an undercooled crystal growth in 2D [63] concluding that the implementation of the adapted enthalpy-porosity scheme is much simpler compared to the phase-field-based LB

models due to their small grid spacing. That did not stop Medvedev and Kassner to investigate an undercooled crystal growth in shear flow [64] by using a composite LB-PF method which, as they pointed out, has potential of generalisation to 3D.

Semma and co-workers studied the application of the LBM in Ga melting problem adopting a simple piece-wise function between the solid fraction and temperature [65; 66] again pointing out the potential of the scheme to be used in linking micro/macro aspects in 3D.

Sun and co-workers [67] proposed a CA-LB model to simulate dendritic solidification in 2D. Because the structures of both methods are very similar (LB originated from the lattice gas cellular automata), it seems natural to couple these methods together. They used the LBM to describe the mass and momentum transport in an undercooled crystal growth. Later they included the heat transfer to simulate single and multi-dendritic growth of binary alloys with melt convection [68], but recently they investigated the effect of melt convection on multi-dendritic growth without considering temperature differences in the simulation domain [69].

Yin *et al.* [70] also used the CA-LB method to simulate solidification at the microscale in 2D. They compared the efficiency of the CA-LB model against the FE-CA model and concluded that the CA-LB method is much more efficient when fluid flow is being considered.

Talati and Taghilou [71] used the LBM to model the phase-change material solidification. They did a comparison of the different methods used in simulations and observed that the LBM outperformed the FV method within the framework of ANSYS Fluent 14 confirming the good efficiency of the LBM.

Rojas *et al.* proposed the PF-LB model to simultaneously simulate growth and motion of a free dendrite under shear flow in 2D [72]. Their implementation only requires a single Cartesian grid and describes the motion of the solid without altering the shape.

Sun and co-workers continued working on the CA-LB method expanding it to 3D to model directional solidification of binary alloys. They investigated tip-splitting of the dendrite tips caused by high solidification rates [73] and studied the bubble formation in dendritic growth [74]. In their studies they employed the popular D3Q19 lattice to describe the mass and momentum transport, however their spatial step and time interval were chosen as 0.5 µm and 0.1 µs which is typical for the phase-field method.

Recently Liu and He developed an enthalpy-based LBM with multiple relaxation times (MRT) for modelling phase change in metal foams in 2D [75]. They used the volumetric LB scheme proposed by Huang and Wu [76] to accurately realise the no-slip velocity condition in the diffusive interface. The solidification/melting case results obtained with their proposed model were compared to those of the FD/FV methods showing good agreement. Further observations were made, namely, that the use of the MRT scheme suppresses the interface oscillations that are present in the Bhatnagar-Gross-Krook (BGK) model and that the bounce-back scheme introduces oscillations in the streamlines near the interface while the streamlines obtained by volumetric LB scheme remain smooth.

2.4 Parallelisation and large-scale solidification modelling

Parallelisation as an effective solution to large-scale 3D problems involving geometrical complexity has been considered for several decades, offering a way of bringing down the computational time to a level that could be of interest to engineers [77; 78]. An early attempt to parallelise multi-physics models involving solidification was made by McManus *et al.* [79]. They implemented the solidification and parallel model into the software tool PHYSICA [80] which uses FV methods on unstructured meshes. Because the parallelisation in PHYSICA is realised using the master/slave communication model, the initial mesh parsing as well as the input/output routines can be very time-consuming for large simulations, which might be disadvantageous.

Jeong *et al.* [30] used a parallel adaptive finite element algorithm in 3D to describe the fluid flow effect on dendritic growth. Their phase-field method for the solidification and the semi-implicit approximated projection method for the fluid flow were both parallelised using message-passing service routines that handled input/output, communication between processors and other tasks. With the parallel setup they were able to achieve a parallel efficiency of 90 % running the code on 32 processors, meaning that the runtime was 28.8 times faster compared to the serial implementation. Importantly they noted that the fluid flow calculations consume approximately 80–90 % of the runtime, which makes it the bottleneck of the numerical algorithm.

George and Warren used a parallel finite difference algorithm in 3D to simulate a single dendrite growth using the phase-field method [81]. The $500 \times 500 \times 500$

grid was partitioned with the help of the message passing interface (MPI). Using the MPI-based libraries for the distributed array managing, they calculated that it would take approximately 10 days at the time to run a simulation of $1000 \times 1000 \times 1000$ elements in parallel on 32 processors. The limiting factor for the simulation is the small time step that is bounded by a stability criterion, $\Delta t < \Delta x^2/8D$, where Δx is the spatial step and D is the diffusion coefficient for the problem. Consequently, stable phase-field simulations are very time-consuming due to the method's sub-micron spatial step size.

Nestler [82] used a combination of MPI routines and OpenMP compiler directives for the parallel execution of their finite difference phase-field algorithm in order to model dendritic growth, grain structures and alloy solidification. The motivation for the parallelisation reportedly is the simulation time of the 3D domains when, for example, it takes 48 hours to fully solidify a domain of $80 \times 80 \times 60$ numerical cells using a serial algorithm.

Wang *et al.* [83] investigated the use of processor virtualisation to parallelise the level-set method for solving solidification problems by the means of adaptive MPI. They found that the solver performance was better when the number of virtual processors (that the domain has been decomposed into) was larger than the number of physical processors due to improved cache performance and optimised communication and computation.

Guo *et al.* [84] developed a parallel multi-grid method to solve a case of multiple solidifying dendrites using a fully coupled phase-field method involving thermo-solutal transport. They employed OpenMP in conjunction with MPI to provide a second level of parallelism, and concluded that the parallel hybrid approach can efficiently utilise supercomputing resources allowing to overcome

the major drawback of the phase-field method that is the computational time.

Numerical algorithms can also be parallelised on graphical processing units (GPUs) using computer unified device architecture (CUDA), for example. CUDA is a parallel computing platform developed by NVIDIA that enables the utilisation of the GPU resources in tandem with the central processing unit (CPU) for general purpose computing. It is widely used in image and video processing, computational sciences, finance sector, as well as for artificial intelligence [85]. The NVIDIA GPUs power 5 out of 7 of the world's fastest supercomputers whose performance can be measured in the order of 100 PFLOPS or 10^{17} floating point operations per second. It is the chosen platform for 17 out of the world's top 20 most energy-efficient supercomputers [86], including TSUBAME 3.0 – the latest version of the TSUBAME supercomputer.

The first large-scale phase-field simulation of dendritic solidification was performed by Shimokawabe *et al.* [87] who used 1156 GPUs on TSUBAME 2.0 supercomputer to model dendritic growth in binary alloy solidification with a $768 \times 1632 \times 3264 \approx 4$ billion element mesh and grid spacing $\Delta x = 0.75$ µm. In addition, to test the weak scaling, they fully utilised the TSUBAME 2.0 supercomputer with 4000 GPUs and 16000 CPUs by simulating 277 billion element mesh and achieving 1 PFLOPS in performance. Unfortunately, mesh sizes like these are essential to model microstructural multi-dendritic patterns using the phase-field method on a macroscale.

Takaki *et al.* [88] also used the TSUBAME 2.0 supercomputer to model the dendrite selection process during directional solidification of a binary alloy, see Figure 2.11. Their phase-field model had 64 billion elements but the physical size of only 29 mm^3.

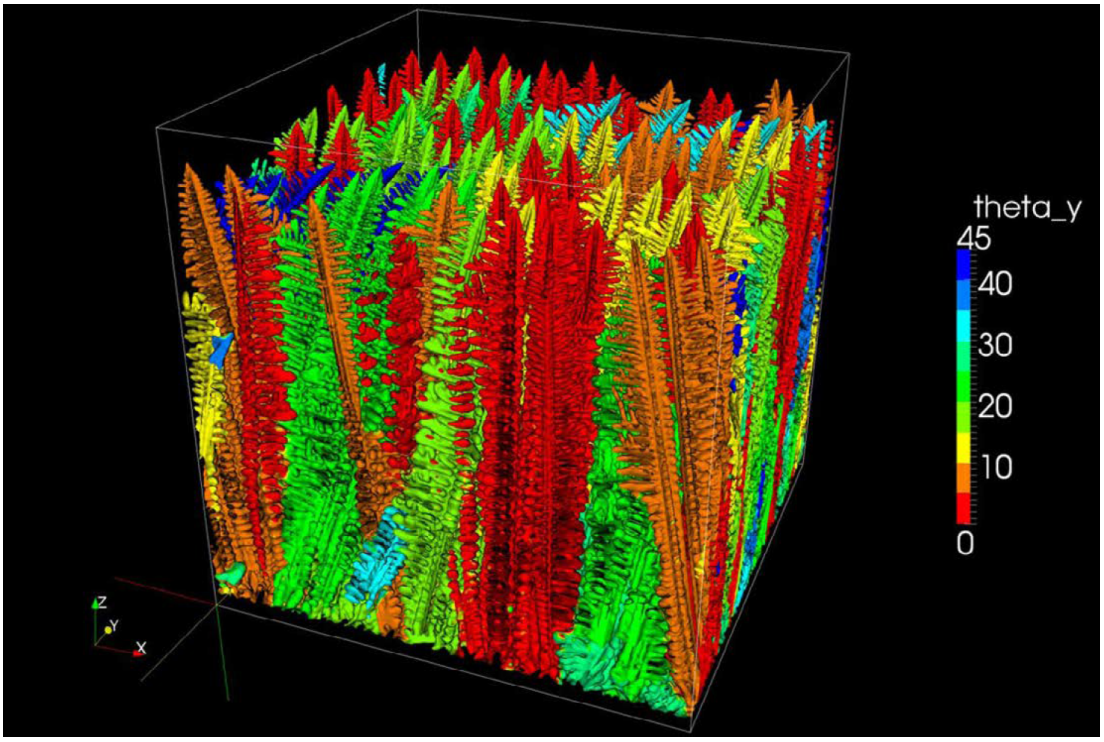


Figure 2.11: Competitive dendritic growth colour marked by the inclination angle θ using the PF-LB method [88].

Takaki *et al.* accelerated the PF-LB algorithm proposed by Rojas *et al.* [72] using CUDA C to enable the utilisation of the GPU resources [89]. They successfully simulated the dendrite growth in shear flow as well as during settling in 2D on a single GPU and predicted the possibility of simulating the settling of dendrites in large 3D computational domains by parallelising the code for the use of multiple GPUs.

Sakane *et al.* [90] performed a large-scale PF-LB simulation of a dendrite growth in forced convection on the TSUBAME 2.5 supercomputer. They used the popular D3Q19 lattice to model fluid flow and they observed that the inclusion of convection increases the calculation time by a factor of 4.4, from 51 to 223 minutes. It means that the flow solver takes around 80 % of the total runtime. While the increase in the calculation time might seem large, the total time of 223 minutes is very reasonable for a billion cell simulation. Moreover, they successfully simulated multiple dendrites in forced convection using a 7 billion cell mesh with the physical domain size still being relatively small – 0.0005 mm^3. State-of-the-art computing with sub-micron grid spacing that allows for the simulated physical domains on the order of only a few cubic millimetres is still far less than the typical size of cast metal components.

The first 3D parallel CA-LB model for dendritic solidification was proposed by Eshraghi *et al.* [91] who used the D3Q15 lattice to model the mass transport in the solute-driven single dendrite growth. Because both methods exhibit local characteristics, they offer good computational efficiency and parallel scalability with potential of leading to large-scale 3D simulations of microstructure evolution on a macroscopic level. That is exactly what they achieved a couple years later by simulating a physical domain of 1 mm^3 using 36 billion grid points [92], see Figure

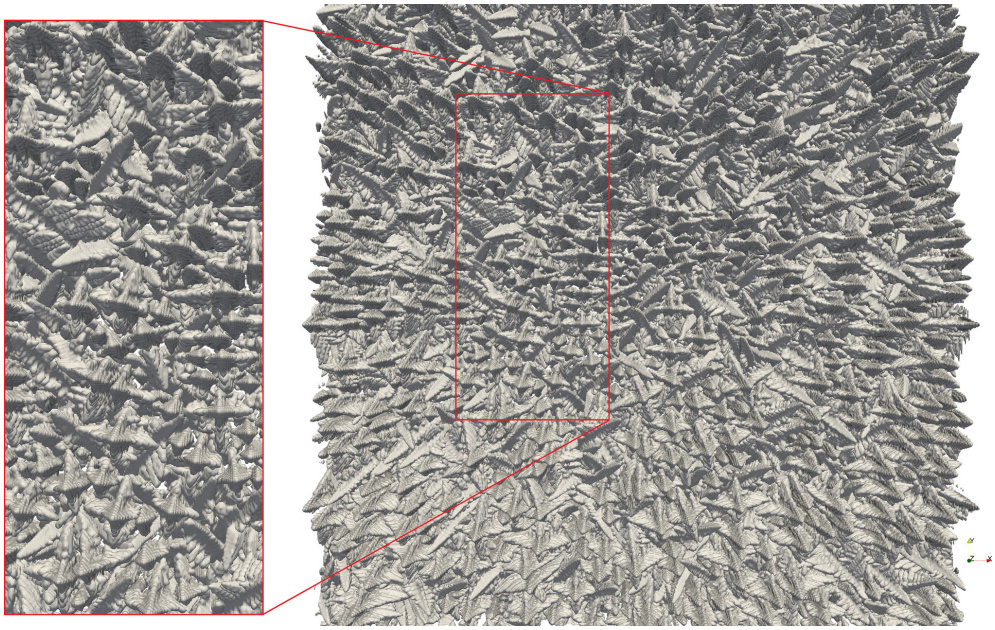


Figure 2.12: The growth of 3D columnar dendritic microstructure using the CA-LB method [92].

2.12. The simulation of the growth of 3D columnar dendritic microstructure with the mesh size of $\Delta x = 0.3$ µm was performed on 6400 CPU cores of the Stampede supercomputer. Millimetre-scale size was achieved by Jelinek *et al.* [93] who simulated the effect of melt convection on more than 3000 nuclei, however the simulation was in 2D.

A comprehensive study of the morphological differences introduced by the flow, including a comparison between 2D and 3D models, has been conducted by Eshraghi *et al.* [94] using the CA method for the cell capturing and the LBM for forced convection.

Recently Alexandrakis [95] achieved a 4 cm^3 domain size in directional solidification with no flow. Alexandrakis used a CA method with grid spacing of $\Delta x = 10$ µm which allowed to reach macroscopic sizes. The method is based on the open source code µMatIC [36; 96–98].

2.5 Conclusion

The relevant literature to this thesis was reviewed in this chapter, and several conclusions can be drawn. A lot of experimental and numerical research has been conducted over the years describing solidification process of liquid melts. The studies have highlighted the effect that convective heat and mass transport has on the evolution of dendritic growth. Whether it is dendrite arm spacing, tip velocities, preferential growth direction or re-/formation of segregation channels, they all can have an effect on the macroscopic properties of the solidified materials. Therefore, it is important to include fluid flow when modelling microstructure solidification. Despite it being computationally expensive and requiring handling of the complicated geometries of the evolving microstructures, with the advances in computational technologies, it has become possible to simulate large-scale domains approaching the size of entire small components. The numerical method that shows the best promise to bridge the gap between micro- and macroscale is the CA-LB. The CA method can handle cell sizes of $O(10\ \mu m)$ compared to the sub-micron length offered by the widely used PF method. The LBM, on the other hand, is faster and more stable than the conventional CFD methods, it can easily handle complex geometries and because of its spatial locality property it can be massively parallelised for large-scale simulations.

Chapter 3

LATTICE BOLTZMANN METHOD

3.1 Introduction

The lattice Boltzmann method is a mesoscopic approach to continuum physics, which has its origins in the kinetic theory. The key advantages of the LBM over conventional CFD methods are relative ease in coupling, particle interaction locality, linear advection term, no Poisson solver for pressure, easy handling of complex boundaries. The locality property of the method allows for easy parallelisation.

From the literature review and based on the reasons above, the lattice Boltzmann method has been chosen to calculate the fluid flow in physical systems within this research. The main concepts of the method are explained in the following sections.

3.2 History

The LBM has originated from the lattice gas cellular automata (LGCA), where the fluid or gas is treated as a collection of particles. The evolution process of the LGCA into the LBM has been recorded in detail by several authors [56; 99–101]. According to Boghosian [99], the first attempt to reproduce hydrodynamics using a lattice gas model was made by Kadanoff and Swift (KS) in 1968 [102], before that, lattice gases were used to model ferromagnets. In the KS 2D liquid gas model, particles can move diagonally from site to site on a regular Cartesian grid via advection or diffusion, without violating any conservation rules. The possible velocities a single particle occupying a lattice site can have are expressed as

$$\mathbf{c}_i = c(\cos(\theta_i), \sin(\theta_i)), \text{ with } \theta_i = \frac{\pi}{4}(2i-1) \text{ for } i = 1-4, \tag{3.1}$$

where $c = \frac{\delta x}{\delta t}\sqrt{2}$ is the lattice speed, δx is the lattice spacing, δt is the time step and $\sqrt{2}$ is the diagonal distance between two lattice sites. The KS model is able to conserve mass, momentum and energy, but due to lack of symmetry it suffers from a strong lattice anisotropy when modelling the decay of sound waves, for example. It has to be noted that the original idea at the time was to study the statistical physics of fluids, and not to offer a new CFD method.

Half a decade later in 1973, Hardy, de Pazzis and Pomeau (HPP) proposed the first cellular automata (CA) model [103], see Figure 3.1. Similar to the KS model, the so-called HPP model also employs a 2D Cartesian square lattice, but the allowed discrete velocities are different (3.2).

$$\mathbf{c}_i = c(\cos(\theta_i), \sin(\theta_i)), \text{ with } \theta_i = \frac{\pi}{2}(i-1) \text{ for } i = 1-4. \tag{3.2}$$

Here $c = \frac{\delta x}{\delta t}$, as particles were allowed to move along the gridlines. A maximum of four particles can reside at each lattice site at any given time as long as they all have different velocities. Mathematically, the state of a site can be represented by a Boolean type function,

$$n_i(\mathbf{x}, t) = \begin{cases} 0, & \text{absence of a particle} \\ 1, & \text{presence of a particle} \end{cases}, \tag{3.3}$$

for a particle with velocity $\mathbf{c}_i$ at coordinate $\mathbf{x}$ and time t. In the HPP model, particles can collide with each other or stream freely in the direction of their velocities. The collision only occurs when the incoming two particles have opposite velocities, otherwise the process does not get triggered, see Figure 3.2. The kinetic equation,

$$n_i(\mathbf{x} + \mathbf{c}_i\delta t, t + \delta t) = n_i(\mathbf{x}, t) + C_i(n(\mathbf{x}, t)), \tag{3.4}$$

describes the streaming and collision dynamics. Here, C_i is the collision operator, which encodes particle anihilation, conservation or emission for the velocity direction i, and it can be written as

$$C_i(n_i(\mathbf{x}, t)) = \begin{cases} -1, & \text{particle anihilation} \\ 0, & \text{particle conservation} \\ 1, & \text{particle emission} \end{cases}. \tag{3.5}$$

The conservation requirements for mass and momentum can be imposed on the collision C_i as

$$\sum_i C_i = 0, \qquad \sum_i \mathbf{c}_i C_i = 0. \tag{3.6}$$

Normally a unit time step, $\delta t = 1$, is used when working in lattice units, so then

(3.4) simplifies to

$$n_i(\mathbf{x} + \mathbf{c}_i, t + 1) = n_i(\mathbf{x}, t) + C_i(n(\mathbf{x}, t)). \tag{3.7}$$

Hydrodynamic variables such as mass density and momentum density can be calculated from the ensemble average of the occupation state number defined as $N_i = \langle n_i \rangle$, where $N_i \in [0, 1]$, in the following way:

$$\rho = \sum_i N_i, \qquad \rho\mathbf{u} = \sum_i \mathbf{c}_i N_i. \tag{3.8}$$

Through the Chapman−Enskog expansion the hydrodynamic equation can be derived, see Section 3.4. However, the Navier–Stokes equations are not recovered correctly. The reason being the same as that for the KS model − insufficient symmetry of the square lattice.

The grid anisotropy problem was finally solved when the importance of symmetry was recognised. It was done by Frisch, Hasslacher and Pomeau [104], and by Wolfram [105] at the same time in 1986. As it turns out, the six-fold symmetric lattice in 2D allows us to recover the fourth rank viscous term tensor from the hydrodynamic equation. The so-called FHP model, named after its authors, employs a hexagonal lattice, see Figure 3.1 with discrete velocities defined as

$$\mathbf{c}_i = c(\cos(\theta_i), \sin(\theta_i)), \text{ with } \theta_i = \frac{\pi}{3}(i - 1) \text{ for } i = 1 - 6. \tag{3.9}$$

Just like in the HPP model, only one particle with velocity $\mathbf{c}_i$ may occupy a lattice site. This exclusion principle leads to Fermi−Dirac distribution for the

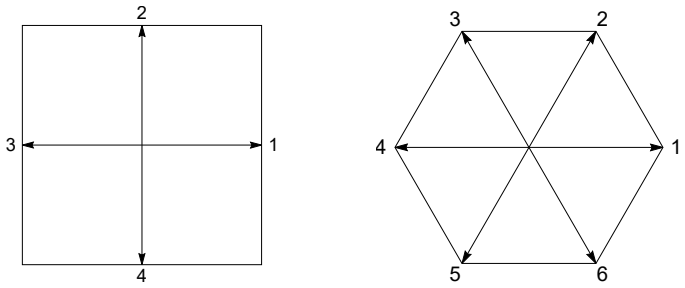


Figure 3.1: First LGCA models shown with numbered discrete velocities: HPP model (left) and FHP model (right).

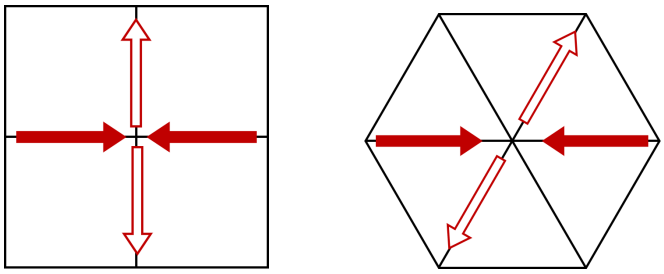


Figure 3.2: Collision process for HPP model (left) and FHP model (right).

mean occupation number N_i,

$$N_i^{eq}(Q_i) = \frac{1}{1+\exp(Q_i)}, \tag{3.10}$$

where Q_i is a linear combination of collision invariants, and the local equilibrium can be written as [106],

$$N_i^{eq} = \frac{\rho}{6}\left[1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + G(\rho)\frac{(\mathbf{c}_i\mathbf{c}_i - c_s^2\mathbf{I}) : \mathbf{uu}}{c_s^4}\right], \tag{3.11}$$

where c_s is the lattice speed of sound, and $G(\rho)$ is a function of the density ρ. Chapman−Enskog analysis allows us to recover the hydrodynamic equations in the incompressible limit, which means that the density variations are much

smaller than the reference density, $\delta\rho \ll \rho$. The equations are,

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0, \\ \frac{\partial \mathbf{u}}{\partial t} + g(\rho)\mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla P + \nu(\rho)\nabla^2 \mathbf{u}, \end{aligned} \tag{3.12}$$

where $g(\rho) = (\rho - 3)/(\rho - 6)$ and $\nu(\rho)$ is another function of density. Because the coefficient in front of the convective term has to be equal to one, but the coefficient in (3.12) $g(\rho) \neq 1$ it leads to the violation of the Galilean invariance, which is characteristic for LGCA models with Fermi−Dirac equilibrium distribution function [100]. One can rescale time, $t \to t/g(\rho)$, however this fix is only valid for small Mach numbers as the pressure depends on the velocity, which is unphysical.

Interestingly, the FHP cannot be extended into 3D in a straightforward way. Instead one has to go to 4D to find a regular lattice that meets the symmetry requirements, and then project down onto 3D space [106]. Such lattice is the face-centered hypercube, and it was proposed by d'Humieres, Lallemand and Frisch in 1986 [107]. The lattice contains 24 discrete velocities, some of which are not the same length and some of which have a multiplicity of two because of the projection. Having 24 velocities per site means that there are $2^{24} \approx 17$ million different possible states of occupancy, which is a lot more than $2^6 = 64$ for the FHP model in 2D. So, the collision rule can no longer be easily worked out, and the necessity for a better way of describing collision processes arises.

Another shortcoming of LGCA is the statistical noise attributed to the Boolean nature of the method. It was dealt with in 1988 when McNamara and Zanetti proposed the use of a single-particle distribution function $f_i = \langle n_i \rangle$, which has a real value between 0 and 1 [108]. The corresponding collision operator also loses its Boolean type, $\Omega_i(f) = \langle C_i(n) \rangle$, and the governing equation for the particle

dynamics can be written as

$$f_i(\mathbf{x} + \mathbf{c}_i\delta t, t + \delta t) - f_i(\mathbf{x}, t) = \Omega_i(f(\mathbf{x}, t)), \tag{3.13}$$

The first step towards a simpler collision was made by Higuera and Jimenez in 1989 when they proposed a linear approximation of the collision operator [109]. They expanded the distribution function around its local equilibrium as

$$f_i = f_i^{eq} + f_i^{neq}, \tag{3.14}$$

where f_i^{neq} is the non-equilibrium distribution function, so that the collisions could be simplified to

$$\Omega_j(f) = \frac{\partial \Omega_j}{\partial f_i}(f_i - f_i^{eq}), \tag{3.15}$$

where $\partial\Omega_j/\partial f_i$ is the collision matrix.

The last simplification to the collision operator was made in the early 1990s [110–113] when a single-relaxation-time (SRT) collision scheme was adopted from the BGK model [114] in kinetic theory,

$$\Omega_i(f) = -\frac{1}{\tau}(f_i - f_i^{eq}), \tag{3.16}$$

where τ is the relaxation time of the particle to its local equilibrium, and the equilibrium function can be generally written as

$$f_i^{eq} = w_i\rho\left[1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{u^2}{2c_s^2}\right], \tag{3.17}$$

where w_i are the weights for a particular lattice. Table 3.1 lists a couple of them.

The final result is a method:

- with no statistical noise,
- with a linear advection term,
- with inherent parallelism due to simple local collisions,
- that is Galilean-invariant up to the second order,
- whose virtual particle velocities at equilibrium obey the Maxwell−Boltzmann distribution at low Mach numbers,
- that leads to the macroscopic Navier−Stokes equation via multiscale analysis, see Section 3.4.

All these properties have made it a serious contender for an alternative numerical approach to the other CFD methods.

3.3 2D and 3D lattices

As discussed in the previous section, lattices need to meet symmetry requirements for macroscopic variables to satisfy continuum equations. In this chapter, the lattices chosen for 2D and 3D numerical simulation in this thesis will be described.

In terms of the lattice choice, the main objective from a physical perspective is of course to use a lattice that allows to recover the Navier−Stokes equation. However, from the computational point of view the fewer variables to do calculations on the better for the efficiency of the algorithm [115]. The symmetry

requirements are given for the lattice velocity set as [115–117]:

$$\begin{aligned}
\sum_i w_i &= 1, \\
\sum_i w_i c_{i\alpha} &= 0, \\
\sum_i w_i c_{i\alpha} c_{i\beta} &= c_s^2 \delta_{\alpha\beta}, \\
\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} &= 0, \\
\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} &= c_s^4 (\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}), \\
\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} c_{i\epsilon} &= 0.
\end{aligned} \tag{3.18}$$

In the 2D HPP and FHP models all the velocity vectors $\mathbf{c}_i$ have equal lengths c_i and weights w_i. However, they will differ for Cartesian lattices with more than four velocities such as D2Q9. Different weights are introduced for different velocities in order to maintain the lattice isotropy. The DdQq notation is used in this thesis to represent a d-dimensional lattice with q velocities [113]. The weights and velocities of the most popular lattices are given in Table 3.1. In the context of this thesis, only 2D and 3D velocity sets with a rest particle or a zero velocity are considered here because they offer better accuracy and numerical stability [118; 119]. The 2D and 3D velocity sets are shown in Figures 3.3 and 3.4.

The most popular model in 2D is the nine-velocity lattice D2Q9, see Figure 3.3. It is the smallest Cartesian lattice in 2D that satisfies the symmetry requirements in (3.18) and can be used to solve Navier−Stokes type problems.

As can be seen from Table 3.1, three dimensions offer more choices in selecting a lattice suitable for hydrodynamic problems, the most popular of them being the D3Q19 model. It offers a good balance between efficiency and stability. While D3Q15 is obviously the more efficient model with ~ 20 % less variables compared to D3Q19, the trade-off between efficiency and stability comes at a price. The

Table 3.1: Properties of velocity sets.

Model	Lattice velocities	Weights	Sound speed
D2Q9	$(0,0)$ $(\pm1,0),(0,\pm1)$ $(\pm1,\pm1)$	$4/9$ $1/9$ $1/36$	$1/\sqrt{3}$
D3Q15	$(0,0,0)$ $(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$ $(\pm1,\pm1,\pm1)$	$2/9$ $1/9$ $1/72$	$1/\sqrt{3}$
D3Q19	$(0,0,0)$ $(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$ $(\pm1,\pm1,0),(\pm1,0,\pm1),(0,\pm1,\pm1)$	$1/3$ $1/18$ $1/36$	$1/\sqrt{3}$
D3Q27	$(0,0,0)$ $(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)$ $(\pm1,\pm1,0),(\pm1,0,\pm1),(0,\pm1,\pm1)$ $(\pm1,\pm1,\pm1)$	$8/27$ $2/27$ $1/54$ $1/216$	$1/\sqrt{3}$

D3Q15 model more than others might experience numerical instabilities [119], for example the checkerboard effect [120], which is an occurrence of unphysical regular patterns in the calculation domain. This instability is more expressed for flows at high Reynolds numbers [121] and not for low Re flows or Stokes flows where $\mathrm{Re} \ll 1$. In fact, D3Q15 has been successfully used in large-scale simulations of dendritic solidification capturing mass transport [91; 92] and momentum transport [94].

On the other hand, D3Q27 offers the best stability out of the three due to its greater lattice isotropy, but it is the least efficient one with nearly twice as many variables as D3Q15 and half as many as D3Q19. Because of its high isotropy, D3Q27 is best suited for turbulent flows – high Reynolds number flows, where the non-linear effects are dominant [122].

Prioritising efficiency while maintaining good stability leads towards the choice of the D3Q19 lattice. Also, when considering 2D physical problems, one might

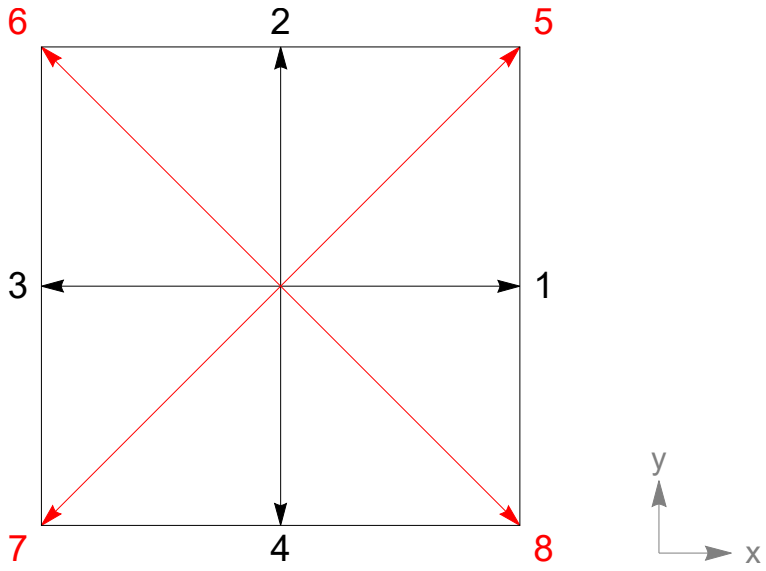


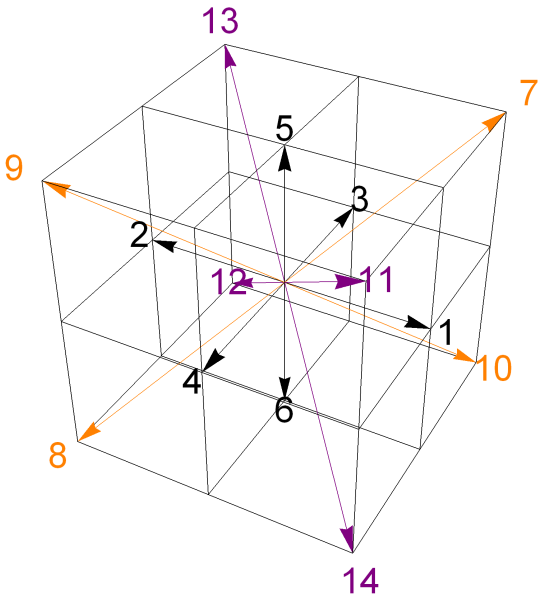
Figure 3.3: D2Q9 lattice. Rest particle velocity is not shown.

carry out calculations using the D2Q9 lattice instead of D3Q19 to minimise computational cost. No information is lost in the process.

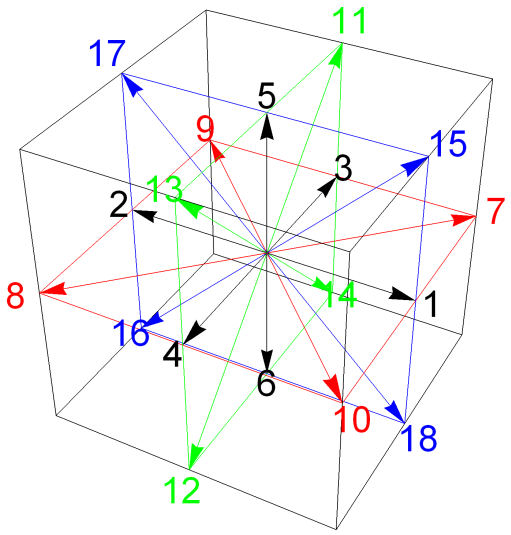
3.4 From lattice Boltzmann to Navier–Stokes

The fact that the collective microscopic behaviour of the LBM leads to that on a macroscopic scale described by the Navier–Stokes equations can be shown by performing a multiscale analysis. One such method that has been an integral part of all the major comprehensive literature on the topic of the LBM [100; 101; 115; 116] is the Chapman–Enskog expansion, where f_i is expanded formally about a small parameter ϵ as

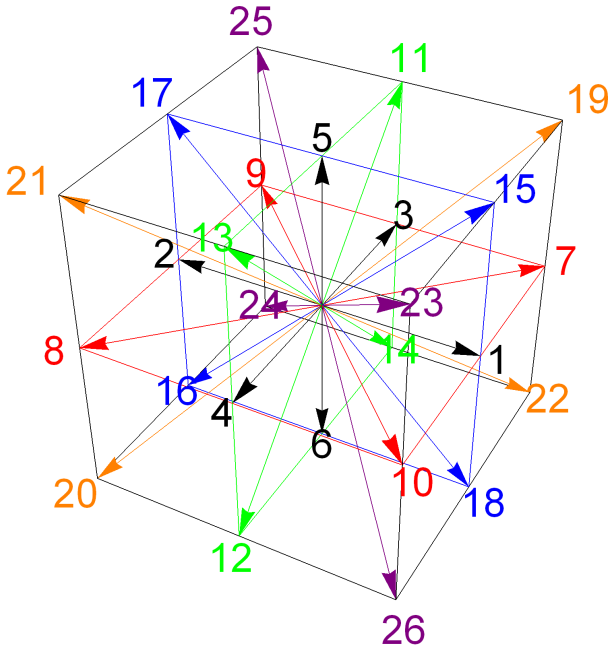
$$f_i = f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + \dots \ , \tag{3.19}$$

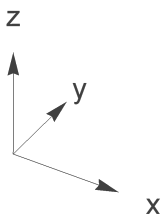


D3Q15



D3Q19





D3Q27

Figure 3.4: D3Q15, D3Q19 and D3Q27 lattices. Rest particle velocity is not shown. The D3Q19 colouring style is borrowed from [123].

where $f_i^{(0)} = f_i^{eq}$ and the ϵ terms contribute towards the non-equilibrium part of the distribution function:

$$f_i = f_i^{eq} + \epsilon f_i^{neq}. \tag{3.20}$$

The time scales in the temporal derivative are separated into convection, t_1, and diffusion scale, t_2, where the latter is assumed to be much slower than the former.

$$\frac{\partial}{\partial t} = \epsilon \frac{\partial}{\partial t_1} + \epsilon^2 \frac{\partial}{\partial t_2}. \tag{3.21}$$

The spatial derivative becomes

$$\frac{\partial}{\partial x} = \epsilon \frac{\partial}{\partial x_1}. \tag{3.22}$$

The last thing left is to expand the discrete velocity Boltzmann equation (DVBE) (3.13). For simplicity, the BGK collision term (3.16) is used in this derivation. After performing a second order Taylor series expansion in time and space and neglecting the higher order terms, (3.13) becomes

$$\left[\frac{\partial}{\partial t} + \mathbf{c}_i \cdot \nabla + \frac{\delta_t}{2} \left(\frac{\partial}{\partial t} + \mathbf{c}_i \cdot \nabla \right)^2 \right] f_i = -\frac{1}{\tau \delta_t} (f_i - f_i^{eq}). \tag{3.23}$$

Applying the expansions from (3.19), (3.21) and (3.22) to (3.23) and separating the terms of different orders of ϵ leads to

$$\epsilon: \qquad \left(\frac{\partial}{\partial t_1} + \mathbf{c}_i \cdot \nabla_1 \right) f_i^{(0)} = -\frac{1}{\tau \delta_t} f_i^{(1)}, \tag{3.24}$$

$$\epsilon^2: \qquad \frac{\partial}{\partial t_2} f_i^{(0)} + \left(\frac{\partial}{\partial t_1} + \mathbf{c}_i \cdot \nabla_1 \right) \left(1 - \frac{1}{2\tau} \right) f_i^{(1)} = -\frac{1}{\tau \delta_t} f_i^{(2)}. \tag{3.25}$$

Mass continuity and Euler's equation can be derived by multiplying (3.24) by 1 and $\mathbf{c}_i$ and taking the zeroth order moment as

$$\frac{\partial \rho}{\partial t_1} + \nabla_1 \cdot (\rho \mathbf{u}) = 0 \tag{3.26}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t_1} + \nabla_1 \cdot \mathbf{\Pi}^{(0)} = 0 \tag{3.27}$$

where $\mathbf{\Pi}^{(0)}$ is the zeroth order momentum flux tensor expressed as

$$\mathbf{\Pi}^{(0)} = \sum_i \mathbf{c}_i \mathbf{c}_i f_i^{(0)} = \rho \mathbf{u}\mathbf{u} + \rho c_s^2 \mathbf{I}. \tag{3.28}$$

On the other scale, the same equations are derived from (3.25) and take the following form,

$$\frac{\partial \rho}{\partial t_2} = 0 \tag{3.29}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t_2} + \left(1 - \frac{1}{2\tau}\right) \nabla_1 \cdot \mathbf{\Pi}^{(1)} = 0, \tag{3.30}$$

where $\mathbf{\Pi}^{(1)}$ is the first order momentum flux tensor expressed as

$$\mathbf{\Pi}^{(1)} = \sum_i \mathbf{c}_i \mathbf{c}_i f_i^{(1)}. \tag{3.31}$$

The term $\mathbf{\Pi}^{(1)}$ is also present on the right hand side when (3.24) is multiplied by $\mathbf{c}_i \mathbf{c}_i$,

$$\frac{\partial \mathbf{\Pi}^{(0)}}{\partial t_1} + \nabla_1 \cdot \mathbf{Q}^{(0)} = -\frac{1}{\tau \delta_t} \mathbf{\Pi}^{(1)}, \tag{3.32}$$

where

$$\mathbf{Q}^{(0)} = \sum_i \mathbf{c}_i \mathbf{c}_i \mathbf{c}_i f_i^{(0)} = \rho c_s^2 [\mathbf{u}\boldsymbol{\delta}]_{\alpha\beta\gamma} + O(u^3). \tag{3.33}$$

where $[\mathbf{u}\boldsymbol{\delta}]_{\alpha\beta\gamma} = u_\alpha\delta_{\beta\gamma} + u_\beta\delta_{\alpha\gamma} + u_\gamma\delta_{\alpha\beta}$. After some substitutions and neglecting the terms of order $O(u^3)$, one can find that

$$\mathbf{\Pi}^{(1)} = -\tau\rho c_s^2\delta_t(\nabla_1\mathbf{u} + \nabla_1\mathbf{u}^T). \tag{3.34}$$

Combining the conservation equations on both scales leads to the macroscopic mass continuity and weakly compressible Navier–Stokes equations,

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0, \tag{3.35}$$

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) = -\nabla p + \nabla\cdot(\rho\nu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)), \tag{3.36}$$

where the pressure p and the kinematic viscosity ν are expressed as

$$p = \rho c_s^2, \tag{3.37}$$

$$\nu = c_s^2\left(\tau - \frac{1}{2}\right)\delta_t. \tag{3.38}$$

The term *weakly compressible* means that the incompressible Navier–Stokes equation is recovered in the low Mach number limit at small velocities because the terms of order $O(\mathrm{Ma}^2)$ and $O(u^3)$ are omitted in the process. These modelling errors will be discussed in the next section. Neglecting the small density variations due to the weak compressibility, the continuity equation and the incompressible NSE can be obtained:

$$\nabla\cdot\mathbf{u} = 0, \tag{3.39}$$

$$\frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}. \tag{3.40}$$

3.5 Stability and accuracy

3.5.1 Stability

In CFD, a typical indicator of stability for explicit schemes is the so-called Courant-Friedrichs-Lewy (CFL) condition, which depending on the evolution equation of the system is

$$C = \frac{u\Delta t}{\Delta x} < 1 \qquad \text{or} \qquad C = \frac{\nu\Delta t}{\Delta x^2} < 1. \tag{3.41}$$

Here the ratios are said to be less than unity, but in reality they are kept well below unity to ensure stability and convergence for 2D and 3D problems. The first condition is for advection problems that can be expressed simply as

$$\frac{\partial\phi}{\partial t} + u\frac{\partial\phi}{\partial x} = 0, \tag{3.42}$$

and the second one is for diffusion problems that can be expressed as

$$\frac{\partial\phi}{\partial t} + \nu\frac{\partial^2\phi}{\partial x^2} = 0. \tag{3.43}$$

Inequalities given in (3.41) simply state that the physical information cannot propagate faster than it is supported by a given lattice with set Δx and Δt. Otherwise, the information is captured incorrectly and lost leading to instabilities. The CFL condition is typically applied to finite difference methods and hyperbolic partial differential equations (PDEs) in particular. Similarly, the LBE is derived from the DVBE, which is essentially a stiff hyperbolic PDE with a source term.

Luckily for the LBE, the second order derivative that describes diffusion is re-

placed by the relaxation of the traceless stress tensor to the local equilibrium. This leaves one with only the advective CFL condition to satisfy. However, because of the lattice space-time relation $\Delta x_i = c_i \Delta t$, (3.41) is automatically satisfied through the low Mach number limitation $C = \mathrm{Ma} \ll 1$ [116].

Non-negativity of the equilibrium distribution function is a sufficient linear stability condition for the BGK and TRT (*two relaxation time*) collision schemes [124]. The bounds for a distribution function are generally $0 < f_i < 1$, but exceeding the lower bound in particular may give rise to instabilities. Stable numerical solutions can still be obtained with negative populations present, but it is advised not to fully trust the result as it might be corrupted [115].

The typical value for the sound speed in lattice Boltzmann is often $c_s = \frac{1}{\sqrt{3}}$, however this value is not a universal constant. It can be determined from the lattice weights of the particular velocity sets, see (3.18), and it has been confirmed that the choice of $c_s = \frac{1}{\sqrt{3}}$ gives the most stable results [2; 121].

The range of the relaxation time τ has a strict lower limit of $\tau > \frac{1}{2}$, which ensures that the fluid viscosity stays positive through the relation (3.38). For the D2Q9 BGK models, the region near $\tau = \frac{1}{2}$ is restrictive for the maximum allowed velocity. The closer τ gets to the value $\frac{1}{2}$, the smaller the maximum velocity must be chosen to keep the simulation stable [125]. On the other side, the combination of BGK and the standard bounce-back scheme has an error that depends on the value of τ − the bigger the relaxation time the larger the error on the boundaries, see Section 3.6.1.

The TRT collision scheme, which is discussed in Section 3.7.2, has two parameters, τ^+ and τ^-, where τ^+ is linked to the fluid viscosity and τ^- is a free parameter that can be tuned for stability and accuracy. Stability of TRT is

controlled by the combination of the both parameters:

$$\Lambda = \left(\tau^{+} - \frac{1}{2}\right)\left(\tau^{-} - \frac{1}{2}\right). \tag{3.44}$$

Λ is normally called the magic parameter, and it governs the stability and accuracy of the TRT model. It has been found that $\Lambda = \frac{1}{4}$ provides the best stability [124; 126], $\Lambda = \frac{1}{6}$ provides the best accuracy for pure diffusion problems [126], $\Lambda = \frac{1}{12}$ provides the best accuracy for pure advection problems [126; 127], $\Lambda = \frac{3}{16}$ places the wall in the bounce-back scheme exactly in the middle between the fluid and boundary nodes [128],

Although stable solutions can be obtained at relatively large velocities, the maximum velocity in lattice Boltzmann is constrained by the low Mach number limit, $\mathrm{Ma} = \frac{u_{\max}}{c_s} \ll 1$, in which the Navier−Stokes equation is recovered. It implies that the maximum velocity must be much smaller than the lattice sound speed, $u_{\max} \ll c_s$. Ideally, one would not exceed $u_{\max} < 0.17$ ($\mathrm{Ma} < 0.3$ [101]) when selecting the maximum fluid velocity, but typically values around $u_{\max} = 0.1$ are chosen for convenience.

3.5.2 Accuracy

The lattice Boltzmann equation (LBE) is second order accurate in space and time. However, the accuracy of the method can be degraded if the boundary condition scheme is chosen poorly. An example of this is a Poiseuille channel flow calculated using the BGK collision operator and the standard bounce-back scheme to realise the no-slip condition on the walls. The standard bounce-back scheme is only first order accurate [129], and it can compromise the overall accuracy of the method.

Most of the boundary schemes are second order accurate, the modified and half-way bounce-back, Zou and He non-equilibrium bounce-back, the Moment Method and others, see Section 3.6. One way to measure the accuracy of the general method used in calculations, is by using the relative L_2 error norm,

$$L_2 = \sqrt{\frac{\sum_i (\phi_i - \phi_i^*)^2}{\sum_i \phi_i^{*2}}}, \tag{3.45}$$

where ϕ_i^* is the exact solution at the calculation domain node i. Simple problems normally have an analytical solution. However, if the problem does not have it, then one can use the field values obtained on the finest grid in order to check the convergence rate and, therefore, confirm the spatial accuracy of the method.

It was shown in Section 3.4 that the LBE leads to a weakly compressible NSE. The weak compressibility comes from the fact that the terms of order $O(u^3)$ and $O(\mathrm{Ma}^2)$ are being neglected in the derivation process. Also, the equilibrium distribution function (3.17) only consists of terms up to the second order in velocity, but that is down to the isotropy of the lattice and the type of problems being calculated. For the physical problems discussed in this thesis, the characteristic velocity and the Mach number are both small, which means that the compressibility error $O(\mathrm{Ma}^2)$ and the cubic $O(u^3)$ error are both negligible.

3.6 Boundary conditions

Boundary conditions (BCs) are a crucial part of finding the correct solution to a physical problem, the other obviously being the differential equation, or a set of them, that governs the evolution of the dynamical system of interest. Different boundary conditions give different unique solutions to a boundary value problem.

One simple example of this is a flow between two parallel infinite plates. If both plates are at rest, $\mathbf{u}_{top\ wall} = 0$ and $\mathbf{u}_{bottom\ wall} = 0$, there is no flow, and the steady-state solution is zero everywhere, $\mathbf{u}_{solution} = 0$. Now, if the top wall is moving at a constant speed, $\mathbf{u}_{top\ wall} = (u_{const}, 0)$, the steady-state solution for the flow changes accordingly to satisfy both boundary conditions simultaneously, which results in a velocity gradient between the plates. So, it is important to set the boundary conditions correctly in order to get the desired outcome.

In the following sections, different popular approaches on how to handle the boundaries in terms of the LBM will be discussed. Starting off with simple kinetic schemes, such as bounce-back and modified bounce-back rules, and then moving on to hydrodynamic schemes, like non-equilibrium bounce-back and Moment Method. Closing up the section, an extension into 3D is proposed for the 2D moment-based method. Numerical simulations are also carried out using the newly derived 3D Moment Method, and the results showing great potential are covered in Sections 5.2 and 5.3.

3.6.1 Kinetic style boundary schemes

Because the LBM originated from the LGCA method, initially the boundary conditions were imposed on the particle distribution functions directly. Not counting the periodic type boundaries, the first velocity boundary condition was implemented using the so-called bounce-back scheme [105; 130]. The idea of the method is that the incoming particle velocities on the boundary are reflected back to the neighbouring nodes where they came from to realise the no-slip BC for velocity, see Figure 3.5. Its simple nature suggests that it would be perfect for flows in complex geometries.

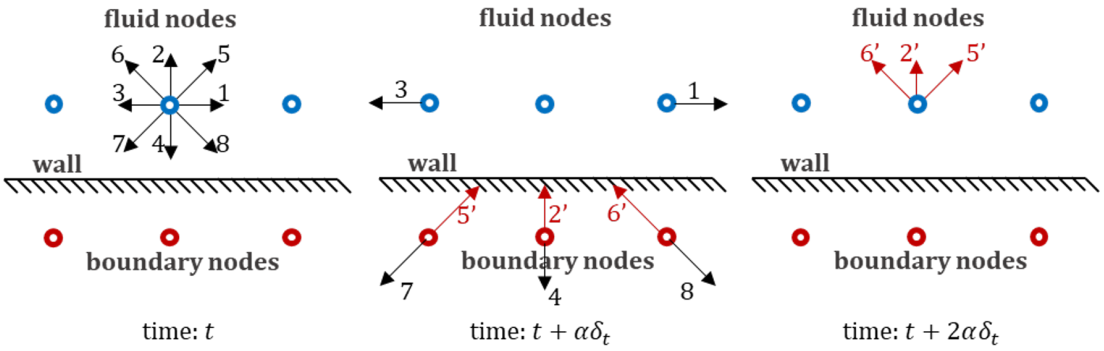


Figure 3.5: Bounce-back schemes visualised for the D2Q9 lattice. $\alpha = 1/2$ for half-way bounce-back and $\alpha = 1$ for the standard bounce-back

However, it was later found that the bounce-back rule suffers from a non-physical slip velocity on the boundary due to the placement of the solid wall [129]. For bounce-back schemes the actual position of the wall is somewhere in between the solid and liquid nodes. It varies with the relaxation time τ, or with the viscosity ν through the relation (3.38). Also, the standard 'on-node' bounce-back scheme is only first order accurate in space [129]. The slip velocity on the boundary for 2D Poiseuille flow is given by

$$u_{\text{slip}} = \frac{2(2\tau - 1)(4\tau - 3) - 6n}{3n^2} u_c, \tag{3.46}$$

where u_c is the maximum velocity in the center of the channel, and n is the number of nodes across the width of the channel. It is obvious from (3.46) that the standard bounce-back scheme, also sometimes referred to as full-way bounce-back, in general has a non-zero slip velocity and it is first order accurate due to the $O(n^{-1})$ term [129].

The accuracy can be improved by choosing, for example the modified bounce-

back scheme, where the slip velocity is given by [129]

$$u_{\text{slip}} = \frac{16\tau(\tau - 1)}{3n^2}\, u_c. \tag{3.47}$$

It is of second order accuracy in space, however the slip velocity is generally still non-zero.

Another second order scheme to mention here is the half-way bounce-back, where the wall is placed in the middle between the solid and fluid nodes, hence the name *half-way*.

Although the position of the wall in the bounce-back case can be adjusted by changing the relaxation time to get rid of the slip velocity, one would rather prefer to leave the choice of the relaxation time value for stability purposes and not error fixing.

3.6.2 Non-equilibrium bounce-back

Probably the most popular hydrodynamic scheme is the one proposed by Zou and He [131]. It allows us to specify pressure and velocity values directly on the boundaries, meaning that macroscopic variables, such as density and velocity are part of the derivation of the BCs for PDFs.

$$\rho = \sum_i f_i, \qquad \rho\mathbf{u} = \sum_i \mathbf{c}_i f_i. \tag{3.48}$$

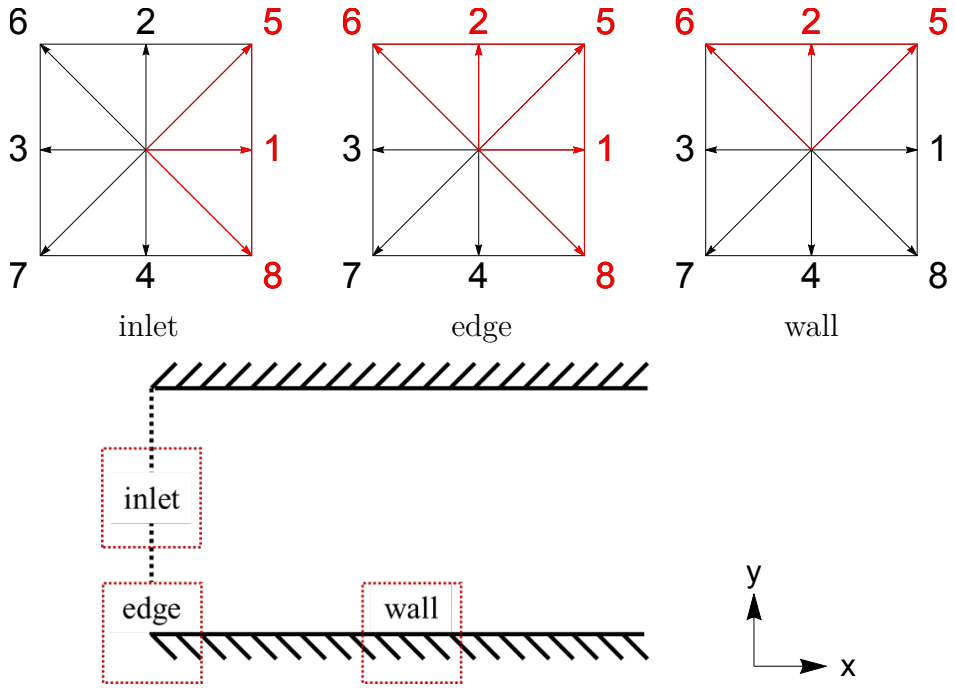


Figure 3.6: Scheme showing the unknown distribution functions at the inlet, edge and wall boundaries. Rest velocity is not shown.

Expanding the equations (3.48),

$$\begin{aligned} \rho &= \sum_i f_i = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8, \\ \rho u_x &= \sum_i f_i c_{ix} = f_1 - f_3 + f_5 - f_7 + f_8 - f_6, \\ \rho u_y &= \sum_i f_i c_{iy} = f_2 - f_4 + f_5 - f_7 + f_6 - f_8, \end{aligned} \tag{3.49}$$

and using the assumption for bounce-back of the non-equilibrium distribution functions at the inlet, see Figure 3.6,

$$f_1 - f_1^{eq} = f_3 - f_3^{eq}, \qquad f_5 - f_5^{eq} = f_7 - f_7^{eq}, \qquad f_8 - f_8^{eq} = f_6 - f_6^{eq}, \tag{3.50}$$

the unknown distribution functions after streaming at the pressure inlet can be

derived,

$$\begin{cases} f_1 &= f_3 + \frac{2}{3}\rho_0 u_x, \\ f_5 &= f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_0 u_x, \\ f_8 &= f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_0 u_x, \end{cases} \tag{3.51}$$

where u_x is found through the consistency condition formed from (3.49):

$$u_x = 1 - \frac{1}{\rho_0}\Big(f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)\Big). \tag{3.52}$$

Conditions for velocity inlet, outlet and other boundaries can be derived similarly.

Furthermore, corner nodes at the inlet and outlet need special attention. Depending on the corner of interest (bottom left inlet node in Figure 3.6), unknown distribution functions can be found as

$$\begin{cases} f_1 &= f_3, \\ f_2 &= f_4, \\ f_5 &= f_7, \\ f_6 &= \frac{1}{2}\Big(\rho_0 - f_0 - 2(f_3 + f_4 + f_7)\Big), \\ f_8 &= \frac{1}{2}\Big(\rho_0 - f_0 - 2(f_3 + f_4 + f_7)\Big). \end{cases} \tag{3.53}$$

Needless to say that f_6 and f_8 are both equal here. Corner nodes should be treated with care because ill-defined boundary conditions at the corners can easily give rise to the numerical instabilities, such as the checkerboard effect.

Extension to 3D

In their paper [131], Zou and He also briefly discuss the derivation process of the unknown PDFs on pressure boundary for the D3Q15 lattice, see Figure 3.4. Using the bounce-back rule for the non-equilibrium part of the distribution function and modifying the tangential momentums, as proposed in [132], they find that

$$\begin{cases} f_1 &= f_2 + \frac{2}{3}\rho_0 u_x, \\ f_7 &= f_8 + \frac{1}{12}\rho_0 u_x - \frac{1}{4}\Big((f_3 - f_4) + (f_5 - f_6)\Big), \\ f_{10} &= f_9 + \frac{1}{12}\rho_0 u_x - \frac{1}{4}\Big((f_3 - f_4) - (f_5 - f_6)\Big), \\ f_{11} &= f_{12} + \frac{1}{12}\rho_0 u_x + \frac{1}{4}\Big((f_3 - f_4) - (f_5 - f_6)\Big), \\ f_{14} &= f_{13} + \frac{1}{12}\rho_0 u_x + \frac{1}{4}\Big((f_3 - f_4) + (f_5 - f_6)\Big), \end{cases} \tag{3.54}$$

where u_x is determined from the consistency condition as

$$u_x = 1 - \frac{1}{\rho_0}\Big(f_0 + f_3 + f_4 + f_5 + f_6 + 2(f_2 + f_8 + f_9 + f_{12} + f_{13})\Big). \tag{3.55}$$

The first attempt, to the author's knowledge, of applying the non-equilibrium bounce-back rule to a D3Q19 lattice was made by Kutay *et al.* [133] who used it to define a pressure inlet and outlet with the normal flow being unknown, see Table 3.2.

Hecht and Harting later proposed a general way of imposing the flow boundary condition for the D3Q19 lattice [134]. Allowing the tangential velocities to be specified at the velocity boundary, in contrast to [133; 135], and using the non-equilibrium bounce-back rule, they introduce two new unknown variables

Table 3.2: Pressure boundary description for the D3Q19 lattice [133].

Inlet	
Known components	$\rho, u_y = 0, u_z = 0,$ $f_0, f_2, f_3, f_4, f_5, f_6, f_8, f_9, f_{11}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}$
Unknowns	$u_x, f_1, f_7, f_{10}, f_{15}, f_{18}$
Relations	$u_{x(in)} = 1 - \frac{f_0+f_3+f_4+f_5+f_6+f_{11}+f_{12}+f_{13}+f_{14}+2(f_2+f_8+f_9+f_{16}+f_{17})}{\rho_{in}}$ $f_1 = f_2 + \frac{1}{3}\rho u_x$ $f_7 = f_8 - \frac{1}{4}(f_3 - f_4) + \frac{1}{6}\rho u_x$ $f_{10} = f_9 + \frac{1}{4}(f_3 - f_4) + \frac{1}{6}\rho u_x$ $f_{15} = f_{16} - \frac{1}{4}(f_5 - f_6) + \frac{1}{6}\rho u_x$ $f_{18} = f_{17} + \frac{1}{4}(f_5 - f_6) + \frac{1}{6}\rho u_x$
Outlet	
Known components	$\rho, u_y = 0, u_z = 0,$ $f_0, f_1, f_3, f_4, f_5, f_6, f_7, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{18}$
Unknowns	$u_x, f_2, f_8, f_9, f_{16}, f_{17}$
Relations	$u_{x(out)} = 1 - \frac{f_0+f_3+f_4+f_5+f_6+f_{11}+f_{12}+f_{13}+f_{14}+2(f_1+f_7+f_{10}+f_{15}+f_{18})}{\rho_{out}}$ $f_2 = f_1 - \frac{1}{3}\rho u_x$ $f_8 = f_7 + \frac{1}{4}(f_3 - f_4) - \frac{1}{6}\rho u_x$ $f_9 = f_{10} - \frac{1}{4}(f_3 - f_4) - \frac{1}{6}\rho u_x$ $f_{16} = f_{15} + \frac{1}{4}(f_5 - f_6) - \frac{1}{6}\rho u_x$ $f_{17} = f_{18} - \frac{1}{4}(f_5 - f_6) - \frac{1}{6}\rho u_x$

that represent the transverse momentum corrections to account for any tangential flows. Setting the velocities to U_x, U_y and U_z the system of the unknown distribution functions at the west face velocity boundary reads

$$\begin{cases} f_1 &= f_2 + \dfrac{1}{3}\rho U_x, \\ f_7 &= f_8 + \dfrac{\rho}{6}(U_x + U_y) - N_y^x, \\ f_{10} &= f_9 + \dfrac{\rho}{6}(U_x - U_y) + N_y^x, \\ f_{15} &= f_{16} + \dfrac{\rho}{6}(U_x + U_z) - N_z^x, \\ f_{18} &= f_{17} + \dfrac{\rho}{6}(U_x - U_z) + N_z^x, \end{cases} \tag{3.56}$$

where u_x is determined from the consistency condition as

$$\rho = f_0+f_3+f_4+f_5+f_6+f_{11}+f_{12}+f_{13}+f_{14}+2(f_2+f_8+f_9+f_{16}+f_{17})+\rho U_x, \quad (3.57)$$

and the transverse momentum corrections N_y^x and N_z^x are expressed as

$$\begin{cases} N_y^x &= \frac{1}{2}\Big(f_3 + f_{11} + f_{14} - (f_4 + f_{12} + f_{13})\Big) - \frac{1}{3}\rho U_y, \\ N_z^x &= \frac{1}{2}\Big(f_5 + f_{11} + f_{13} - (f_6 + f_{12} + f_{14})\Big) - \frac{1}{3}\rho U_z. \end{cases} \quad (3.58)$$

The end result is a boundary condition scheme that is explicit, local, shows second order accuracy, does not depend on the relaxation time, and allows one to specify exact on-site velocity values at the boundary in all directions. In addition, if a no-slip condition for the velocity is imposed, then no numerical slip can be observed in contrast to the standard bounce-back scheme.

Although the results from Hecht and Harting are promising, their scheme itself for treating boundaries in 3D might seem overcomplicated:

- applying the non-equilibrium bounce-back rule and then modifying the tangential distribution functions by introducing the transverse momentum corrections;
- recalculating the resting particle distribution function at the pressure boundary edges;
- correcting the slip along the convex edges.

While these corrections are overcomplicated, they are all a necessary part of the scheme. Lastly, in order to judge whether the strain rate tensor $\Pi_{\alpha\beta}$ is set up

correctly at the boundaries, they use observations from the results, which is fine, however one would preferably choose to set conditions for the stress components directly.

3.6.3 Moment analysis of boundary conditions

By using the moment grouping, a method introduced by Bennett [2], one can analyse the boundary conditions from a hydrodynamic point of view. It has already been successfully applied to the bounce-back rule, diffuse [136] and specular reflection and Zou and He velocity and pressure boundary conditions in 2D [2], but it is yet to be used to analyse hydrodynamic schemes in 3D, namely the one proposed by Hecht and Harting.

So far in 2D, the moment analysis has revealed that Zou and He non-equilibrium bounce-back rule applied on the wall nodes essentially imposes conditions for both momentums, ρU_x and ρU_y, and a third order moment Q_{xxy} (or Q_{xyy} depending on the direction), which might seem a bit odd. Similarly, for an open boundary, the analysis has revealed that the conditions are being imposed onto both momentums, ρU_x and ρU_y, and a third order moment Q_{xyy} (or Q_{xxy}). The consistency condition allows to set the density, ρ_0, instead of the momentum at the pressure boundary.

For the bounce-back rule, the moment analysis has shown that the conditions at the wall are only being set for the normal momentum and both third order moments Q_{xxy} and Q_{xyy}. That gives an insight on why this scheme is suffering from having a slip velocity at the walls − there is no condition being imposed on the tangential velocity.

For the first time, Hecht and Harting 3D boundary conditions are analysed

here using the moment grouping. Inserting (3.58) into (3.56) gives the resulting system to be investigated:

$$\begin{cases} f_1 &= f_2 + \frac{1}{3}\rho U_x, \\ f_7 &= f_8 + \frac{\rho}{6}(U_x + U_y) - \frac{1}{2}(f_3 + f_{11} + f_{14} - f_4 - f_{12} - f_{13}) + \frac{1}{3}\rho U_y, \\ f_{10} &= f_9 + \frac{\rho}{6}(U_x - U_y) + \frac{1}{2}(f_3 + f_{11} + f_{14} - f_4 - f_{12} - f_{13}) - \frac{1}{3}\rho U_y, \\ f_{15} &= f_{16} + \frac{\rho}{6}(U_x + U_z) - \frac{1}{2}(f_5 + f_{11} + f_{13} - f_6 - f_{12} - f_{14}) + \frac{1}{3}\rho U_z, \\ f_{18} &= f_{17} + \frac{\rho}{6}(U_x - U_z) + \frac{1}{2}(f_5 + f_{11} + f_{13} - f_6 - f_{12} - f_{14}) - \frac{1}{3}\rho U_z. \end{cases} \tag{3.59}$$

These are the expressions of the incoming particle distribution functions for an open boundary. To find out what conditions are being set for which hydrodynamic moments, the moments are expressed in terms of the distribution functions, where the unknowns are substituted with their respective expressions from (3.59). The moments for the D3Q19 velocity set are defined in (3.63). This gives the following:

$$
\begin{pmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ \Pi_{xx} \\ \Pi_{yy} \\ \Pi_{zz} \\ \Pi_{xy} \\ \Pi_{xz} \\ \Pi_{yz} \\ Q_{xxy} \\ Q_{xxz} \\ Q_{xyy} \\ Q_{xzz} \\ Q_{yyz} \\ Q_{yzz} \\ S_{xxyy} \\ S_{xxzz} \\ S_{yyzz} \end{pmatrix} = \begin{pmatrix} f_0 + f_3 + f_4 + f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + \\ +2(f_2 + f_8 + f_9 + f_{16} + f_{17}) + \rho U_x \\ \rho U_x \\ \rho U_y \\ \rho U_z \\ 2(f_2 + f_8 + f_9 + f_{16} + f_{17}) + \rho U_x \\ f_3 + f_4 + 2(f_8 + f_9) + f_{11} + f_{12} + f_{13} + f_{14} + \frac{1}{3}\rho U_x \\ f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + 2f_{16} + 2f_{17} + \frac{1}{3}\rho U_x \\ f_4 - f_3 + 2(f_8 - f_9) - f_{11} + f_{12} + f_{13} - f_{14} + \rho U_y \\ f_6 - f_5 - f_{11} + f_{12} - f_{13} + f_{14} + 2(f_{16} - f_{17}) + \rho U_z \\ f_{11} + f_{12} - f_{13} - f_{14} \\ f_4 - f_3 - f_{11} + f_{12} + f_{13} - f_{14} + \rho U_y \\ f_6 - f_5 - f_{11} + f_{12} - f_{13} + f_{14} + \rho U_z \\ \frac{1}{3}\rho U_x \\ \frac{1}{3}\rho U_x \\ f_{11} - f_{12} + f_{13} - f_{14} \\ f_{11} - f_{12} - f_{13} + f_{14} \\ 2(f_8 + f_9) + \frac{1}{3}\rho U_x \\ 2(f_{16} + f_{17}) + \frac{1}{3}\rho U_x \\ f_{11} + f_{12} + f_{13} + f_{14} \end{pmatrix} \tag{3.60}
$$

Five moments that do not contain any trace of the distribution functions are the three momentums, ρu_x, ρu_y and ρu_z, and two third order moments, Q_{xyy} and Q_{xzz}. So, the conditions are being set for the momentums as $\rho u_x = \rho U_x$, $\rho u_y =$

ρU_y and $\rho u_z = \rho U_z$, and for the third order moments as $Q_{xyy} = \frac{1}{3}\rho U_x$ and $Q_{xzz} = \frac{1}{3}\rho U_x$, where $\frac{1}{3}\rho U_x$ is an equilibrium approximation of the higher order moment with the terms $O(u^3)$ being omitted, see (3.33). While the selection of the former three moments makes sense, the last two conditions seem questionable because they both state the same and do not have a simple physical interpretation. At flat boundaries, they are not the preferred velocity moments to impose conditions on. Unfortunately, that seems to be the common theme among the methods involving some variation of the kinetic bounce-back rule.

The moment analysis has shown that when the boundary conditions have kinetic origins, they lack rigidness in a physical sense on a macroscopic level. This is avoided by the moment-based boundary method, which imposes conditions directly onto the hydrodynamic moments. It is described in more detail in the next section.

3.6.4 Moment Method

According to Guo and Shu [101] the first hydrodynamic scheme for velocity boundary conditions was proposed by Noble *et al.* [137] who used hydrodynamic moments, more precisely the velocity to solve for the unknown distribution functions at the boundaries. Their motivation was simple and valid − the bounce-back boundary condition has a relaxation time dependent slip, and it cannot be easily generalised to mass inflows or moving walls. In addition, they wrote in their paper:

> Rather than developing a technique that maintains a discrete particle momentum balance, the hydrodynamic approach seeks to maintain a specified velocity profile on the boundaries. During each time step in

> the LBM procedure, the particle distribution at each node is modified by collision, forcing, and streaming. The goal of the hydrodynamic approach is to prescribe this process in such a fashion that the desired velocity conditions are satisfied at the end of the time step. [137]

They were employing the hexagonal D2Q7 lattice similar to the one shown in Figure 3.1, but with a rest velocity. On the boundary, only two distributions functions are unknown (for example, f_2 and f_3 on the bottom wall), so two conditions are required to solve them. Other lattices, such as D2Q9 and D3Q19 have more unknown functions, therefore they require more independent moments. Luckily, for the D2Q7 lattice, one needs to look no further than (3.61). As shown in Table 3.3, there are exactly two linearly independent moments available when considering pressure and velocities.

$$\begin{cases} \rho &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6, \\ \rho u_x &= f_1 - f_4 + \dfrac{1}{2}(f_2 - f_3 + f_6 - f_5), \\ \rho u_y &= \dfrac{2}{\sqrt{3}}(f_2 + f_3 - f_5 - f_6), \end{cases} \tag{3.61}$$

Table 3.3: Unknown moments combinations at the bottom wall of D2Q7.

#	Moments	Unknown f combinations
1	$\rho, \rho u_y$	$f_2 + f_3$
2	ρu_x	$f_2 - f_3$

Depending on what conditions are being applied at the wall, the solution for the incoming particle distribution functions at the boundary can be as simple as

$$\begin{cases} f_2 &= \rho\left(\dfrac{u_y}{\sqrt{3}} + u_x\right) - f_1 + f_4 + f_5, \\ f_3 &= \rho\left(\dfrac{u_y}{\sqrt{3}} - u_x\right) + f_1 - f_4 + f_6. \end{cases} \tag{3.62}$$

Later the list of hydrodynamic moments, Table 3.3, was expanded to include the energy [138].

The more general moment-based method for imposing hydrodynamic boundary conditions was proposed recently by Bennett [2]. He used the fact that since there is a one-to-one linear mapping from the distribution functions to its moments ($\mathbf{m} = \mathbf{M}\mathbf{f}$), this mapping can be inverted ($\mathbf{f} = \mathbf{M}^{-1}\mathbf{m}$). One can switch between moments $\mathbf{m}$ and particle distribution functions $\mathbf{f}$ very easily and therefore can impose a condition on all $\mathbf{m}$ to find all $\mathbf{f}$. At a boundary, not all the moments are independent, but the idea is to impose conditions on linearly independent moments only and convert this into the particle basis to find the unknown (incoming) distribution functions. The intention is to use as far as possible the hydrodynamic moments only because we are simulating hydrodynamics.

It has been successfully applied to various physical systems, where the exact BCs must be employed [136; 139–142]. Furthermore, its stability and accuracy has also been commented upon briefly [143]. Their results show that the method is second order accurate for velocity and pressure, which matches the accuracy of the LBM. Another important finding is that the Moment Method in combination with the BGK collision operator works very well in the region of low to moderate Reynolds numbers, but more sophisticated collision operators (*e.g.*, two- or multiple-relaxation-time) are preferable for multidimensional flows at high Reynolds numbers.

The merits of the Moment Method are listed below. The Moment Method is:

- exact – concrete hydrodynamic conditions can be exactly specified on the boundaries, whether it is pressure, momentum or momentum flux. Because of this, there is no relaxation time dependent numerical slip on the zero

velocity boundaries, for example. Other methods, such as bounce-back and diffuse reflection do not fully possess this property [144].

- on-site – conditions are imposed directly on the boundary nodes meaning that for example setting a no-slip condition gives precisely a zero velocity at the wall nodes. That is contrary to the bounce-back and diffuse reflection rule, where the wall is placed somewhere in-between the nodes.

- local – only the information from the boundary cell is used in the calculations. The method can be parallelised easily for efficient computing. This is in contrast to the interpolation/extrapolation schemes, where the information from the interior fluid node is required.

- second order accurate – the method does not degrade the accuracy of the lattice Boltzmann method.

- straightforward – the idea and the implementation of the method is relatively simple. The unknown distribution functions are calculated using the hydrodynamic moments that the boundary conditions are imposed on. The trickiest part might be finding a physical interpretation for the higher order moments that may be needed at edge or corner boundaries, see Section 3.6.5. Complexity wise it is not as simple as the bounce-back rule, however the concept is simpler and more straightforward than other methods that involve a mixture of the bounce-back rule, hydrodynamic moments, momentum corrections and other modifications to the distribution functions.

- correct for continuum flows – setting boundary conditions directly for the hydrodynamic moments on a macroscale seems more reasonable than de-

scribing virtual particle-wall interactions borrowed from the kinetic theory.

The last argument is along the lines of what Sam Bennett wrote in his thesis:

> Given the analysis of the previous chapters, it is clear that kinetic style boundaries are not appropriate for the D2Q9 system. When dealing with the 9 moment truncated system, rather than the full set of infinitely many moments, a key link with the continuous Boltzmann equation has been lost. Principally, the length of the grid spacing is much larger than the mean free path, and physical effects at a scale smaller than the hydrodynamic scale are not recreated. Therefore, the conditions at the boundary should be viewed as macroscopic, not microscopic. [2]

Because the derivation process of the Moment Method for the D2Q9 lattice is fully covered in [2] and several later works, it will be skipped here. Next, we extend for the first time the Moment Method to three dimensions and explicitly derive conditions from this method for the D3Q19 model.

3.6.5 Moment Method for the D3Q19 model

Similar to D2Q9 having 9 independent moments, the D3Q19 model has exactly 19 independent moments, which are all listed in (3.63). Starting from the zeroth velocity moment, which is otherwise known as density, and going all the way to the third and fourth order moments, whose physical interpretation are not as clear.

$$
\begin{aligned}
\rho &= \sum_i f_i = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 + f_9 + \\
&\quad + f_{10} + f_{11} + f_{12} + f_{13} + f_{14} + f_{15} + f_{16} + f_{17} + f_{18} \\
\rho u_x &= \sum_i f_i c_{ix} = f_1 - f_2 + f_7 - f_8 - f_9 + f_{10} + f_{15} - f_{16} - f_{17} + f_{18} \\
\rho u_y &= \sum_i f_i c_{iy} = f_3 - f_4 + f_7 - f_8 + f_9 - f_{10} + f_{11} - f_{12} - f_{13} + f_{14} \\
\rho u_z &= \sum_i f_i c_{iz} = f_5 - f_6 + f_{11} - f_{12} + f_{13} - f_{14} + f_{15} - f_{16} + f_{17} - f_{18} \\
\Pi_{xx} &= \sum_i f_i c_{ix}^2 = f_1 + f_2 + f_7 + f_8 + f_9 + f_{10} + f_{15} + f_{16} + f_{17} + f_{18} \\
\Pi_{yy} &= \sum_i f_i c_{iy}^2 = f_3 + f_4 + f_7 + f_8 + f_9 + f_{10} + f_{11} + f_{12} + f_{13} + f_{14} \\
\Pi_{zz} &= \sum_i f_i c_{iz}^2 = f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + f_{15} + f_{16} + f_{17} + f_{18} \\
\Pi_{xy} &= \sum_i f_i c_{ix} c_{iy} = f_7 + f_8 - f_9 - f_{10} \\
\Pi_{xz} &= \sum_i f_i c_{ix} c_{iz} = f_{15} + f_{16} - f_{17} - f_{18} \\
\Pi_{yz} &= \sum_i f_i c_{iy} c_{iz} = f_{11} + f_{12} - f_{13} - f_{14} \\
Q_{xxy} &= \sum_i f_i c_{ix}^2 c_{iy} = f_7 - f_8 + f_9 - f_{10} \\
Q_{xxz} &= \sum_i f_i c_{ix}^2 c_{iz} = f_{15} - f_{16} + f_{17} - f_{18} \\
Q_{xyy} &= \sum_i f_i c_{ix} c_{iy}^2 = f_7 - f_8 - f_9 + f_{10} \\
Q_{xzz} &= \sum_i f_i c_{ix} c_{iz}^2 = f_{15} - f_{16} - f_{17} + f_{18} \\
Q_{yyz} &= \sum_i f_i c_{iy}^2 c_{iz} = f_{11} - f_{12} + f_{13} - f_{14} \\
Q_{yzz} &= \sum_i f_i c_{iy} c_{iz}^2 = f_{11} - f_{12} - f_{13} + f_{14} \\
S_{xxyy} &= \sum_i f_i c_{ix}^2 c_{iy}^2 = f_7 + f_8 + f_9 + f_{10} \\
S_{xxzz} &= \sum_i f_i c_{ix}^2 c_{iz}^2 = f_{15} + f_{16} + f_{17} + f_{18} \\
S_{yyzz} &= \sum_i f_i c_{iy}^2 c_{iz}^2 = f_{11} + f_{12} + f_{13} + f_{14}
\end{aligned}
\tag{3.63}
$$

These moments are used in calculating the incoming particle distribution functions at the local domain boundaries. There are five unknown distribution functions at every face boundary, nine unknowns at every edge boundary and twelve unknowns at every corner boundary. It means that five, nine and twelve linearly

independent moments are required at every face, edge and corner, respectively, to solve for the unknown distribution functions. However, not all of the moments in (3.63) are linearly independent. In fact, they can be placed into groups of unique combinations of distribution functions for any given boundary whether it is at the face, the edge or the corner. Next, the derivation process for each of these different cases will be described, distinguishing between velocity and pressure type boundaries.

Face velocity

So, from the list (3.63), five hydrodynamic moments are chosen to impose a boundary condition on the face for velocity.

If, for example, the west boundary is chosen (see Figure 3.7), the unknown incoming PDFs at the west face are f_1, f_7, f_{10}, f_{15} and f_{18}. Grouped up moments and their corresponding combinations of PDFs are shown in Table 3.4. Moments that are not listed in Table 3.4 do not contain the information of the unknown functions. They are Π_{yz}, Q_{yyz}, Q_{yzz} and S_{yyzz}. By looking at their respective expressions in (3.63) one can confirm that they do not contain the unknown functions of interest.

The moments in a row are not linearly independent so only one moment can be picked from each row to impose a constraint on it and to solve the system for the unknowns at the boundary. The aim is to pick hydrodynamic moments only and avoid selecting the higher order moments as much as possible because they do not have a clear physical meaning.

For the velocity boundary it is logical to select the three momentums, $\rho u_x, \rho u_y$ and ρu_z. The remaining two moments are chosen to be the momentum fluxes,

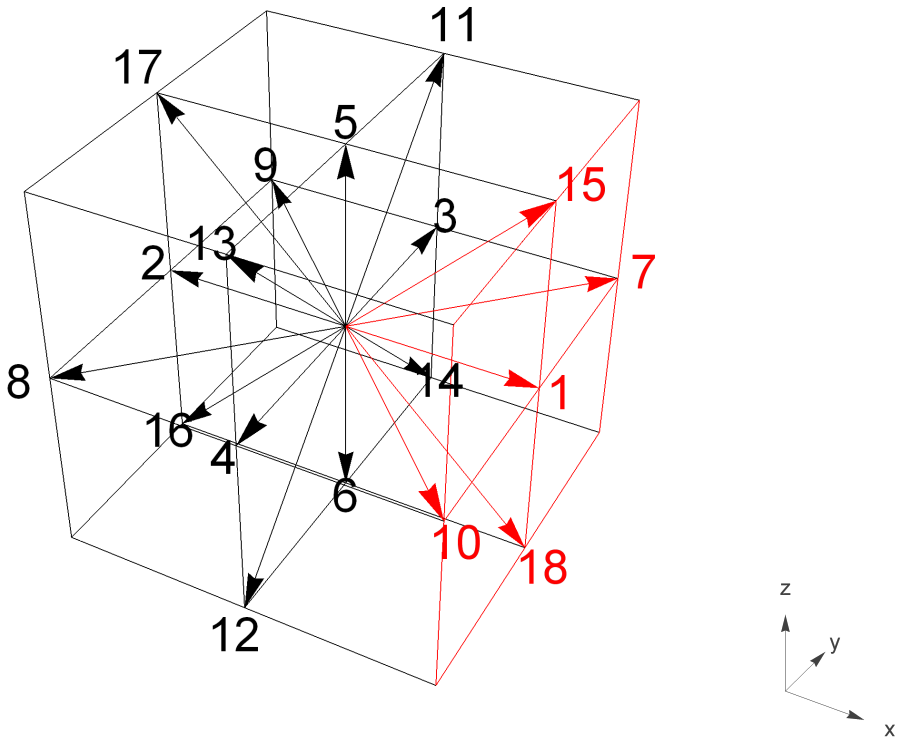


Figure 3.7: Unknown incoming distribution functions (red) at the west face boundary.

Π_{yy} and Π_{zz} due to a simpler physical interpretation compared to the higher order moments. Now that there are five linearly independent equations for the five unknowns, the system can finally be solved. Before solving it, the momentum fluxes need to be defined. Using the first two terms from the Chapman−Enskog multiscale expansion, see Section 3.4, the momentum flux is approximated as

$$\Pi_{yy} = \Pi_{yy}^{eq} + \epsilon\Pi_{yy}^{(1)} + O(\epsilon^2), \qquad \Pi_{zz} = \Pi_{zz}^{eq} + \epsilon\Pi_{zz}^{(1)} + O(\epsilon^2). \tag{3.64}$$

Replacing the terms in the above expressions with (3.28) and (3.34) gives

$$\Pi_{yy} = \frac{\rho}{3} + \rho u_y^2 - \frac{2\rho\tau}{3}\frac{\partial u_y}{\partial y}, \qquad \Pi_{zz} = \frac{\rho}{3} + \rho u_z^2 - \frac{2\rho\tau}{3}\frac{\partial u_z}{\partial z}. \tag{3.65}$$

Table 3.4: Moment combinations at the west face boundary.

#	Moments	Unknown f combinations
1	$\rho, \rho u_x, \Pi_{xx}$	$f_1 + f_7 + f_{10} + f_{15} + f_{18}$
2	$\rho u_y, \Pi_{xy}, Q_{xxy}$	$f_7 - f_{10}$
3	$\rho u_z, \Pi_{xz}, Q_{xxz}$	$f_{15} - f_{18}$
4	$\Pi_{yy}, Q_{xyy}, S_{xxyy}$	$f_7 + f_{10}$
5	$\Pi_{zz}, Q_{xzz}, S_{xxzz}$	$f_{15} + f_{18}$

For simple boundaries, such as velocity inlet, slip and no-slip walls moving with a constant velocity, the tangential velocity derivatives in (3.64) can be discarded giving the following expressions for the momentum fluxes at the velocity face boundary:

$$\Pi_{yy} = \frac{\rho}{3} + \rho u_y^2, \qquad \Pi_{zz} = \frac{\rho}{3} + \rho u_z^2. \tag{3.66}$$

Setting the velocities at the face boundary to U_x, U_y and U_z, and using the selected moment expressions from (3.63), the system of equations takes the following form:

$$\begin{cases} f_1 + f_7 + f_{10} + f_{15} + f_{18} &= \rho U_x - f_2 + f_8 + f_9 + f_{16} + f_{17} \\ f_7 - f_{10} &= \rho U_y - f_3 + f_4 + f_8 - f_9 - f_{11} + f_{12} + f_{13} - f_{14}, \\ f_{15} - f_{18} &= \rho U_z - f_5 + f_6 - f_{11} + f_{12} - f_{13} + f_{14} + f_{16} - f_{17}, \\ f_7 + f_{10} &= \dfrac{\rho}{3} + \rho U_y^2 - f_3 - f_4 - f_8 - f_9 - f_{11} - f_{12} - f_{13} - f_{14}, \\ f_{15} + f_{18} &= \dfrac{\rho}{3} + \rho U_z^2 - f_5 - f_6 - f_{11} - f_{12} - f_{13} - f_{14} - f_{16} - f_{17}. \end{cases} \tag{3.67}$$

Solving the system (3.67) yields the unknown functions:

$$\begin{cases} f_1 &= \rho\left(U_x - U_y^2 - U_z^2 - \frac{2}{3}\right) + f_2 + f_3 + f_4 + f_5 + f_6 + \\ &\quad +2(f_8 + f_9 + f_{11} + f_{12} + f_{13} + f_{14} + f_{16} + f_{17}), \\ f_7 &= \frac{\rho}{2}\left(\frac{1}{3} + U_y(U_y + 1)\right) - f_3 - f_9 - f_{11} - f_{14}, \\ f_{10} &= \frac{\rho}{2}\left(\frac{1}{3} + U_y(U_y - 1)\right) - f_4 - f_8 - f_{12} - f_{13}, \\ f_{15} &= \frac{\rho}{2}\left(\frac{1}{3} + U_z(U_z + 1)\right) - f_5 - f_{11} - f_{13} - f_{17}, \\ f_{18} &= \frac{\rho}{2}\left(\frac{1}{3} + U_z(U_z - 1)\right) - f_6 - f_{12} - f_{14} - f_{16}. \end{cases} \tag{3.68}$$

The first equation from (3.68) can be simplified by using the consistency condition, which relates the density and momentum normal to the west face boundary,

$$\rho = f_0+f_3+f_4+f_5+f_6+f_{11}+f_{12}+f_{13}+f_{14}+2(f_2+f_8+f_9+f_{16}+f_{17})+\rho U_x, \tag{3.69}$$

and substituting it into the equation for f_1. The unknown functions at the west face velocity boundary can then be expressed in the following compact form:

$$\begin{cases} f_1 &= \rho\left(\frac{1}{3} - U_y^2 - U_z^2\right) - f_0 - f_2 + f_{11} + f_{12} + f_{13} + f_{14}, \\ f_7 &= \frac{\rho}{2}\left(\frac{1}{3} + U_y(U_y + 1)\right) - f_3 - f_9 - f_{11} - f_{14}, \\ f_{10} &= \frac{\rho}{2}\left(\frac{1}{3} + U_y(U_y - 1)\right) - f_4 - f_8 - f_{12} - f_{13}, \\ f_{15} &= \frac{\rho}{2}\left(\frac{1}{3} + U_z(U_z + 1)\right) - f_5 - f_{11} - f_{13} - f_{17}, \\ f_{18} &= \frac{\rho}{2}\left(\frac{1}{3} + U_z(U_z - 1)\right) - f_6 - f_{12} - f_{14} - f_{16}. \end{cases} \tag{3.70}$$

Face pressure

Pressure boundary requires the density to be specified at the west face, leaving out the normal momentum as it is now an unknown moment (see Table 3.4). So, the only change from the velocity type boundary is the selection of the density, ρ, in the first group. Other momentums, ρu_y and ρu_z, and momentum fluxes, Π_{yy} and Π_{zz}, remain unchanged.

Restricting the pressure inlet boundary to normal flow, the tangential velocities are set to zero. Due to the velocities being zero and their derivatives being zero, only the first terms in the momentum flux expressions remain from (3.65) giving

$$\rho u_y = 0, \qquad \rho u_z = 0, \qquad \Pi_{yy} = \frac{\rho}{3}, \qquad \Pi_{zz} = \frac{\rho}{3}. \tag{3.71}$$

Setting the pressure value at the boundary to $p = \rho_0 c_s^2 = \dfrac{\rho_0}{3}$, where ρ_0 is being imposed, and solving the system,

$$\begin{cases} f_1 + f_7 + f_{10} + f_{15} + f_{18} & = \rho_0 - f_0 - f_2 - f_3 - f_4 - f_5 - f_6 - f_8 + \\ & \quad - f_9 - f_{11} - f_{12} - f_{13} - f_{14} - f_{16} - f_{17}, \\ f_7 - f_{10} & = -f_3 + f_4 + f_8 - f_9 - f_{11} + f_{12} + f_{13} - f_{14}, \\ f_{15} - f_{18} & = -f_5 + f_6 - f_{11} + f_{12} - f_{13} + f_{14} + f_{16} - f_{17}, \\ f_7 + f_{10} & = \dfrac{\rho_0}{3} - f_3 - f_4 - f_8 - f_9 - f_{11} - f_{12} - f_{13} - f_{14}, \\ f_{15} + f_{18} & = \dfrac{\rho_0}{3} - f_5 - f_6 - f_{11} - f_{12} - f_{13} - f_{14} - f_{16} - f_{17}. \end{cases} \tag{3.72}$$

gives the following unknown functions for the west face boundary:

$$\begin{cases} f_1 &= \frac{\rho_0}{3} - f_0 - f_2 + f_{11} + f_{12} + f_{13} + f_{14}, \\ f_7 &= \frac{\rho_0}{6} - f_3 - f_9 - f_{11} - f_{14}, \\ f_{10} &= \frac{\rho_0}{6} - f_4 - f_8 - f_{12} - f_{13}, \\ f_{15} &= \frac{\rho_0}{6} - f_5 - f_{11} - f_{13} - f_{17}, \\ f_{18} &= \frac{\rho_0}{6} - f_6 - f_{12} - f_{14} - f_{16}. \end{cases} \tag{3.73}$$

The normal velocity, u_x, can be calculated as

$$u_x = 1 - \frac{1}{\rho_0}\Big(f_0 + f_3 + f_4 + f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + \\ + 2(f_2 + f_8 + f_9 + f_{16} + f_{17})\Big). \tag{3.74}$$

Edge velocity

For the edge boundary, the number of unknown PDFs is nine, and nine linearly independent combinations are required to solve for the unknowns.

For example, for the south-west edge boundary the unknown PDFs are shown in red in Figure 3.8. They are the same five from the west face plus five functions from the south face. Because one function overlaps, f_7 in this case, there end up being nine unknown function: f_1, f_3, f_7, f_9, f_{10}, f_{11}, f_{14}, f_{15} and f_{18}. The different combinations of the incoming distribution functions and the corresponding moments are listed in Table 3.5. Ideally, one would like to pick the first nine appropriate moments, however the combinations appearing to be different are not all linearly independent. One can easily check that by looking at the rows 4, 9 and 10 in Table 3.5, for example.

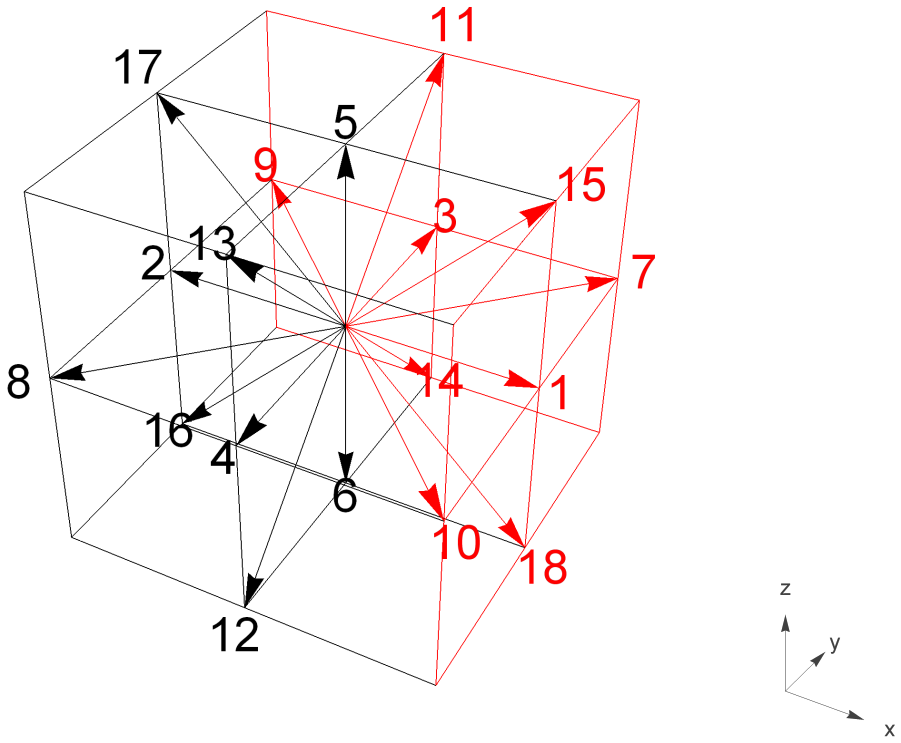


Figure 3.8: Unknown incoming distribution functions (red) at the south-west edge boundary.

The matrix composed from these expressions is a rank-two matrix meaning that only two of the involved different row moments can be selected to specify a boundary condition.

$$\text{rank}\left[\begin{array}{c|c|c} 4 & f_{11}-f_{14}+f_{15}-f_{18} & \rho u_z \\ 9 & f_{15}-f_{18} & \begin{matrix}\Pi_{xz}\\ Q_{xxz}\end{matrix} \\ 10 & f_{11}-f_{14} & \begin{matrix}\Pi_{yz}\\ Q_{yyz}\end{matrix} \end{array}\right] = 2 \tag{3.75}$$

Table 3.5: Unknown function combinations and the moments at the south-west edge boundary.

#	Moments	Unknown f combinations
1	ρ	$f_1 + f_3 + f_7 + f_9 + f_{10} + f_{11} + f_{14} + f_{15} + f_{18}$
2	ρu_x	$f_1 + f_7 - f_9 + f_{10} + f_{15} + f_{18}$
3	ρu_y	$f_3 + f_7 + f_9 - f_{10} + f_{11} + f_{14}$
4	ρu_z	$f_{11} - f_{14} + f_{15} - f_{18}$
5	Π_{xx}	$f_1 + f_7 + f_9 + f_{10} + f_{15} + f_{18}$
6	Π_{yy}	$f_3 + f_7 + f_9 + f_{10} + f_{11} + f_{14}$
7	Π_{zz}	$f_{11} + f_{14} + f_{15} + f_{18}$
8	Π_{xy}	$f_7 - f_9 - f_{10}$
9	Π_{xz}, Q_{xxz}	$f_{15} - f_{18}$
10	Π_{yz}, Q_{yyz}	$f_{11} - f_{14}$
11	Q_{xxy}	$f_7 + f_9 - f_{10}$
12	Q_{xyy}	$f_7 - f_9 + f_{10}$
13	Q_{xzz}, S_{xxzz}	$f_{15} + f_{18}$
14	Q_{yzz}, S_{yyzz}	$f_{11} + f_{14}$
15	S_{xxyy}	$f_7 + f_9 + f_{10}$

The same simply noticeable restriction applies to the rows 7, 13 and 14.

$$\operatorname{rank}\left[\begin{array}{c|c|c} 7 & f_{11} + f_{14} + f_{15} + f_{18} & \Pi_{zz} \\ 13 & f_{15} + f_{18} & \begin{matrix} Q_{xzz} \\ S_{xxzz} \end{matrix} \\ 14 & f_{11} + f_{14} & \begin{matrix} Q_{yzz} \\ S_{yyzz} \end{matrix} \end{array}\right] = 2 \tag{3.76}$$

So, in a situation where there are more unknown combinations than unknowns, the linearly independent rows have to be selected prioritising the physically interpretable ones. By looking at the rank of the matrix consisting of the unknown combinations for the south-west edge boundary, it turns out that the first seven rows from Table 3.5 are all linearly independent. Other dependencies are less obvious. The row 8 is a linear combination of the rows 1, 2 and 3.

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & f_1+f_3+f_7+f_9+f_{10}+f_{11}+f_{14}+f_{15}+f_{18} & \rho \\ 2 & f_1+f_7-f_9+f_{10}+f_{15}+f_{18} & \rho u_x \\ 3 & f_3+f_7+f_9-f_{10}+f_{11}+f_{14} & \rho u_y \\ 8 & f_7-f_9-f_{10} & \Pi_{xy} \end{array}\right] = 3 \quad (3.77)$$

The rows 9 and 10 have already been covered earlier. The rows 11 and 12 are a linear combination of the rows 1, 3, 5 and 1, 2, 6, respectively.

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & f_1+f_3+f_7+f_9+f_{10}+f_{11}+f_{14}+f_{15}+f_{18} & \rho \\ 3 & f_3+f_7+f_9-f_{10}+f_{11}+f_{14} & \rho u_y \\ 5 & f_1+f_7+f_9+f_{10}+f_{15}+f_{18} & \Pi_{xx} \\ 11 & f_7+f_9-f_{10} & Q_{xxy} \end{array}\right] = 3 \quad (3.78)$$

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & f_1+f_3+f_7+f_9+f_{10}+f_{11}+f_{14}+f_{15}+f_{18} & \rho \\ 2 & f_1+f_7-f_9+f_{10}+f_{15}+f_{18} & \rho u_x \\ 6 & f_3+f_7+f_9+f_{10}+f_{11}+f_{14} & \Pi_{yy} \\ 12 & f_7-f_9+f_{10} & Q_{xyy} \end{array}\right] = 3 \quad (3.79)$$

And finally, the row 15 is a linear combination of the rows 1, 5 and 6.

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & f_1+f_3+f_7+f_9+f_{10}+f_{11}+f_{14}+f_{15}+f_{18} & \rho \\ 5 & f_1+f_7+f_9+f_{10}+f_{15}+f_{18} & \Pi_{xx} \\ 6 & f_3+f_7+f_9+f_{10}+f_{11}+f_{14} & \Pi_{yy} \\ 15 & f_7+f_9+f_{10} & S_{xxyy} \end{array}\right] = 3 \quad (3.80)$$

Considering the available options from the analysis above, for the velocity boundary at the south-west edge, the first nine appropriate moments are the three momentums, $\rho u_x, \rho u_y, \rho u_z$, the momentum fluxes and shear stresses, Π_{xx}, Π_{yy}, Π_{zz}, Π_{xy}, Π_{xz} or Π_{yz}, and one higher order moment, Q_{xzz} or Q_{yzz}. There is

still some freedom in selecting the moments to complete the system, however no matter how the nine moments are chosen, having the higher order moment in the selection is inevitable. Basing the choice of the two moments on the symmetry of the components, meaning that either Π_{xz} and Q_{yzz} or Π_{yz} and Q_{xzz} are selected, the final system is written as

$$\text{rank}\left[\begin{array}{c|c|c} 2 & f_1+f_7-f_9+f_{10}+f_{15}+f_{18} & \rho u_x \\ 3 & f_3+f_7+f_9-f_{10}+f_{11}+f_{14} & \rho u_y \\ 4 & f_{11}-f_{14}+f_{15}-f_{18} & \rho u_z \\ 5 & f_1+f_7+f_9+f_{10}+f_{15}+f_{18} & \Pi_{xx} \\ 6 & f_3+f_7+f_9+f_{10}+f_{11}+f_{14} & \Pi_{yy} \\ 7 & f_{11}+f_{14}+f_{15}+f_{18} & \Pi_{zz} \\ 8 & f_7-f_9-f_{10} & \Pi_{xy} \\ 9 & f_{15}-f_{18} & \Pi_{xz} \\ 14 & f_{11}+f_{14} & Q_{yzz} \end{array}\right] = 9. \tag{3.81}$$

Using the truncated approximation (3.65) for the momentum fluxes, Π_{xx}, Π_{yy} and Π_{zz}, and shear stresses, Π_{xy} and Π_{xz}, the higher order moment Q_{yzz} is approximated using its equilibrium value (3.33), where the terms of order $O(u^3)$ are neglected. This can be justified by the fact that only the equilibrium value of $Q_{\alpha\beta\gamma}$ is used in the recovery of the Navier−Stokes equation up to the second order through the Chapman−Enskog analysis. The equilibrium approximation is written as

$$Q_{yzz} = \frac{\rho}{3}(u_y + u_z\delta_{yz} + u_z\delta_{yz}) = \frac{\rho}{3}u_y. \tag{3.82}$$

Setting the south-west edge boundary velocities to U_x, U_y and U_z, the unknown distribution function values are given as

$$
\begin{cases}
f_1 &= \rho\left(\frac{2}{3}(2U_y-1)+U_x-U_xU_y-U_y^2-U_z^2\right)+ \\
&+f_2+f_5+f_6+2(f_4+f_{16}+f_{17})+4(f_8+f_{12}+f_{13}), \\
f_3 &= \rho\left(\frac{1}{3}(2U_y-1)+U_x-U_xU_y-U_x^2\right)+ \\
&+f_4+2(f_2+f_{16}+f_{17})+4f_8, \\
f_7 &= \frac{\rho}{2}\left(\frac{2}{3}-U_x-U_y+(U_x+U_y)^2\right)+ \\
&-f_2-f_4-f_{12}-f_{13}-f_{16}-f_{17}-3f_8, \\
f_9 &= \frac{\rho}{2}\left(\frac{1}{3}-U_x+U_x^2\right)-f_2-f_8-f_{16}-f_{17}, \\
f_{10} &= \frac{\rho}{2}\left(\frac{1}{3}-U_y+U_y^2\right)-f_4-f_8-f_{12}-f_{13}, \\
f_{11} &= \frac{\rho}{2}\left(\frac{1}{3}U_y+U_z(1-U_x)\right)-\frac{1}{2}(f_5-f_6)+f_{12}+f_{16}-f_{17}, \\
f_{14} &= \frac{\rho}{2}\left(\frac{1}{3}U_y-U_z(1-U_x)\right)+\frac{1}{2}(f_5-f_6)+f_{13}-f_{16}+f_{17}, \\
f_{15} &= \frac{\rho}{2}\left(\frac{1}{3}(1-U_y)+U_z(U_z+U_x)\right)-\frac{1}{2}(f_5+f_6)-f_{12}-f_{13}-f_{16}, \\
f_{18} &= \frac{\rho}{2}\left(\frac{1}{3}(1-U_y)+U_z(U_z-U_x)\right)-\frac{1}{2}(f_5+f_6)-f_{12}-f_{13}-f_{17},
\end{cases}
\tag{3.83}
$$

The density in the above equations is given by the formula,

$$
\rho=\frac{f_0+f_5+f_6+2(f_2+f_4+f_{12}+f_{13}+f_{16}+f_{17}+2f_8)}{1-U_x-U_y+U_xU_y}, \tag{3.84}
$$

which is expressed in terms of the known distribution functions and the relevant moments at the south-west edge boundary.

Edge pressure

Specifying the pressure inlet at the edge is not straight forward. The conditions there have to agree with the ones at both adjacent faces, but that cannot be achieved due to the uncertainty of the velocity values. At the south-west edge, the normal pointing into the domain has components on x and y axis. The conditions for velocities on the west face read u_x = unknown, $u_y = 0$ and $u_z = 0$, and on the south face they are $u_x = 0$, u_y = unknown and $u_z = 0$. Problems arise when trying to merge these conditions. Tangential velocity is easy, $u_z = 0$, but how to know what value to set for the other velocities? Are they both unknown or both zero, or maybe one is unknown while the other is zero? This is not a common physical setup, in fact it is far from it. Rarely, if at all, two pressure inlets are encountered being perpendicular to each other. One possible setup is shown in Figure 3.9 where two ducts form the perpendicular pressure inlets. In this situation there is a solid wall separating the two openings meaning that all the velocities are zero at that point.

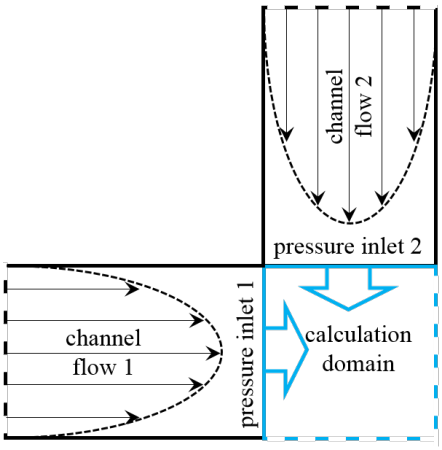


Figure 3.9: Two pressure inlets.

Adopting the idea for the physical conditions at the edge, the system of the

nine moments can now be formed. The only difference from the velocity edge boundary is that the pressure is known. Because of this and (3.77), the shear stress Π_{xy} is left out of the selection leading to

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & f_1+f_3+f_7+f_9+f_{10}+f_{11}+f_{14}+f_{15}+f_{18} & \rho \\ 2 & f_1+f_7-f_9+f_{10}+f_{15}+f_{18} & \rho u_x \\ 3 & f_3+f_7+f_9-f_{10}+f_{11}+f_{14} & \rho u_y \\ 4 & f_{11}-f_{14}+f_{15}-f_{18} & \rho u_z \\ 5 & f_1+f_7+f_9+f_{10}+f_{15}+f_{18} & \Pi_{xx} \\ 6 & f_3+f_7+f_9+f_{10}+f_{11}+f_{14} & \Pi_{yy} \\ 7 & f_{11}+f_{14}+f_{15}+f_{18} & \Pi_{zz} \\ 9 & f_{15}-f_{18} & \Pi_{xz} \\ 14 & f_{11}+f_{14} & Q_{yzz} \end{array}\right] = 9. \quad (3.85)$$

Setting the pressure value at the boundary to ρ_0 and solving the system formed from (3.85) gives the following unknown function values at the south-west edge

boundary:

$$\begin{cases} f_1 &= \frac{\rho_0}{3} - f_0 - f_2 + 2(f_{12} + f_{13}), \\ f_3 &= \frac{2\rho_0}{3} - f_0 - f_4 - f_5 - f_6 - 2(f_{12} + f_{13}), \\ f_7 &= -\frac{2\rho_0}{3} + f_0 + f_2 + f_4 + f_5 + f_6 + f_8 + f_{12} + f_{13} + f_{16} + f_{17}, \\ f_9 &= \frac{\rho_0}{6} - f_2 - f_8 - f_{16} - f_{17}, \\ f_{10} &= \frac{\rho_0}{6} - f_4 - f_8 - f_{12} - f_{13}, \\ f_{11} &= -\frac{1}{2}(f_5 - f_6) + f_{12} + f_{16} - f_{17}, \\ f_{14} &= \frac{1}{2}(f_5 - f_6) + f_{13} - f_{16} + f_{17}, \\ f_{15} &= \frac{\rho_0}{6} - \frac{1}{2}(f_5 + f_6) - f_{12} - f_{13} - f_{16}, \\ f_{18} &= \frac{\rho_0}{6} - \frac{1}{2}(f_5 + f_6) - f_{12} - f_{13} - f_{17}. \end{cases} \tag{3.86}$$

Edge pressure-velocity

In situations where the two adjacent faces have different boundary conditions imposed on them, namely velocity and pressure, the density together with the three momentums have to be specified simultaneously at the edge boundary. It means that ρ, ρu_x, ρu_y and ρu_z are definitely selected from Table 3.5. Momentum fluxes Π_{xx}, Π_{yy} and Π_{zz} also get included. Only one of the shear stresses, Π_{xz} or Π_{yz}, can be selected because of (3.75), and only one of the third order moments, Q_{xzz} or Q_{yzz}, is a viable option due to (3.76). Following the choice made earlier when talking about the velocity and pressure boundaries, the moments Π_{xz} and Q_{yzz} are selected to complete the system. This leads to the same selection of moments (3.85) as in the pressure-pressure edge case.

Setting the density to ρ_0 and velocities to U_x, U_y, U_y, and solving for the unknown distribution functions gives the following expressions for the pressure-velocity boundary at the south-west edge:

$$\begin{cases} f_1 &= \rho\left(\frac{1}{3}(1+U_y) - U_y^2 - U_z^2\right) - f_0 - f_2 + 2(f_{12}+f_{13}), \\ f_3 &= \rho\left(\frac{1}{3}(2-U_y) - U_x^2\right) - f_0 - f_4 - f_5 - f_6 - 2(f_{12}+f_{13}), \\ f_7 &= \frac{\rho}{2}\left(-\frac{4}{3} + U_x(1+U_x) + U_y(1+U_y)\right) + \\ &\quad + f_0 + f_2 + f_4 + f_5 + f_6 + f_8 + f_{12} + f_{13} + f_{16} + f_{17}, \\ f_9 &= \frac{\rho}{2}\left(\frac{1}{3} - U_x + U_x^2\right) - f_2 - f_8 - f_{16} - f_{17}, \\ f_{10} &= \frac{\rho}{2}\left(\frac{1}{3} - U_y + U_y^2\right) - f_4 - f_8 - f_{12} - f_{13}, \\ f_{11} &= \frac{\rho}{2}\left(\frac{1}{3}U_y + U_z(1-U_x)\right) - \frac{1}{2}(f_5 - f_6) + f_{12} + f_{16} - f_{17}, \\ f_{14} &= \frac{\rho}{2}\left(\frac{1}{3}U_y - U_z(1-U_x)\right) + \frac{1}{2}(f_5 - f_6) + f_{13} - f_{16} + f_{17}, \\ f_{15} &= \frac{\rho}{2}\left(\frac{1}{3}(1-U_y) + U_z(U_z+U_x)\right) - \frac{1}{2}(f_5 + f_6) - f_{12} - f_{13} - f_{16}, \\ f_{18} &= \frac{\rho}{2}\left(\frac{1}{3}(1-U_y) + U_z(U_z-U_x)\right) - \frac{1}{2}(f_5 + f_6) - f_{12} - f_{13} - f_{17}. \end{cases} \tag{3.87}$$

Corner velocity

For the corner boundary, the number of unknown PDFs and therefore the number of required linearly independent equations is twelve. The low-south-west corner node is considered here. It means that the twelve unknown functions are f_1, f_3, f_5, f_7, f_9, f_{10}, f_{11}, f_{13}, f_{14}, f_{15}, f_{17} and f_{18} (see Figure 3.10).

Listing all the unknown combinations for the low-south-west corner boundary gives a total of 19 different equations. They are shown in Table 3.6. Every

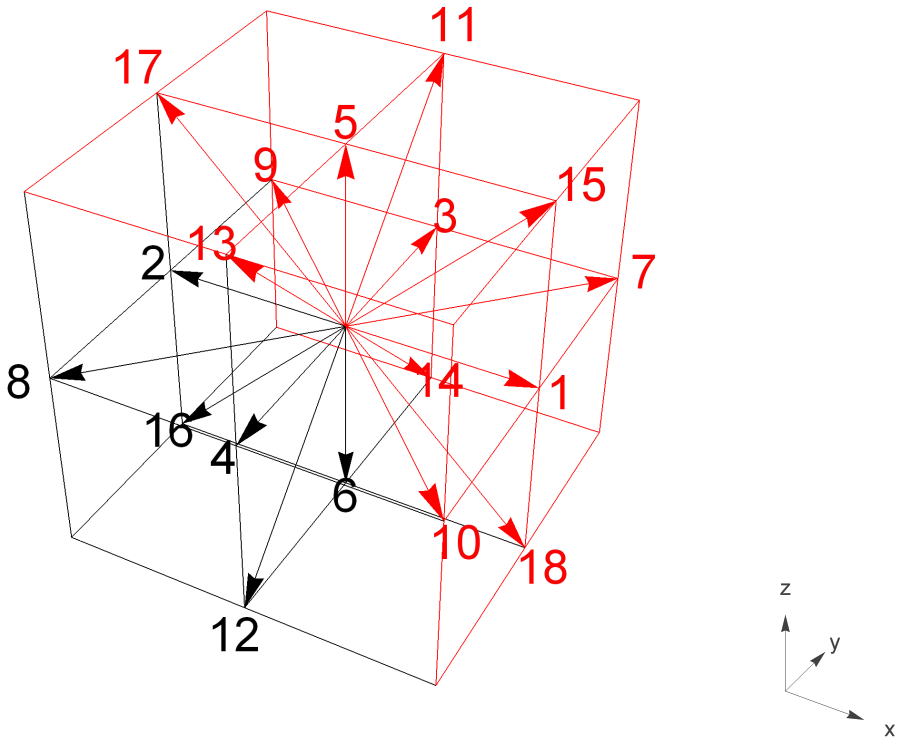


Figure 3.10: Unknown incoming distribution functions (red) at the low-south-west corner boundary.

moment has a unique combination of the unknown distribution functions so one is left with no choice but selecting the first twelve appropriate moments to impose the boundary conditions on them.

The first nine rows of the moment combinations in Table 3.6 are linearly independent. The row 10 turns out to be a linear combination of the rows 1, 2, 3, 4, 8 and 9.

Table 3.6: Unknown function combinations and the moments at the low-south-west edge boundary.

#	Moments	Unknown f combinations
1	ρ	$f_1 + f_3 + f_5 + f_7 + f_9 + f_{10} + f_{11} + f_{13} + f_{14} + f_{15} + f_{17} + f_{18}$
2	ρu_x	$f_1 + f_7 - f_9 + f_{10} + f_{15} - f_{17} + f_{18}$
3	ρu_y	$f_3 + f_7 + f_9 - f_{10} + f_{11} - f_{13} + f_{14}$
4	ρu_z	$f_5 + f_{11} + f_{13} - f_{14} + f_{15} + f_{17} - f_{18}$
5	Π_{xx}	$f_1 + f_7 + f_9 + f_{10} + f_{15} + f_{17} + f_{18}$
6	Π_{yy}	$f_3 + f_7 + f_9 + f_{10} + f_{11} + f_{13} + f_{14}$
7	Π_{zz}	$f_5 + f_{11} + f_{13} + f_{14} + f_{15} + f_{17} + f_{18}$
8	Π_{xy}	$f_7 - f_9 - f_{10}$
9	Π_{xz}	$f_{15} - f_{17} - f_{18}$
10	Π_{yz}	$f_{11} - f_{13} - f_{14}$
11	Q_{xxy}	$f_7 + f_9 - f_{10}$
12	Q_{xxz}	$f_{15} + f_{17} - f_{18}$
13	Q_{xyy}	$f_7 - f_9 + f_{10}$
14	Q_{xzz}	$f_{15} - f_{17} + f_{18}$
15	Q_{yyz}	$f_{11} + f_{13} - f_{14}$
16	Q_{yzz}	$f_{11} - f_{13} + f_{14}$
17	S_{xxyy}	$f_7 + f_9 + f_{10}$
18	S_{xxzz}	$f_{15} + f_{17} + f_{18}$
19	S_{yyzz}	$f_{11} + f_{13} + f_{14}$

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & \begin{array}{c} f_1 + f_3 + f_5 + f_7 + f_9 + f_{10} + \\ + f_{11} + f_{13} + f_{14} + f_{15} + f_{17} + f_{18} \end{array} & \rho \\ 2 & f_1 + f_7 - f_9 + f_{10} + f_{15} - f_{17} + f_{18} & \rho u_x \\ 3 & f_3 + f_7 + f_9 - f_{10} + f_{11} - f_{13} + f_{14} & \rho u_y \\ 4 & f_5 + f_{11} + f_{13} - f_{14} + f_{15} + f_{17} - f_{18} & \rho u_z \\ 8 & f_7 - f_9 - f_{10} & \Pi_{xy} \\ 9 & f_{15} - f_{17} - f_{18} & \Pi_{xz} \\ 10 & f_{11} - f_{13} - f_{14} & \Pi_{yz} \end{array}\right] = 6 \tag{3.88}$$

The row 12 is a linear combination of the rows 2, 5, 8, 9 and 11.

$$\operatorname{rank}\left[\begin{array}{c|c|c} 2 & f_1+f_7-f_9+f_{10}+f_{15}-f_{17}+f_{18} & \rho u_x \\ 5 & f_1+f_7+f_9+f_{10}+f_{15}+f_{17}+f_{18} & \Pi_{xx} \\ 8 & f_7-f_9-f_{10} & \Pi_{xy} \\ 9 & f_{15}-f_{17}-f_{18} & \Pi_{xz} \\ 11 & f_7+f_9-f_{10} & Q_{xxy} \\ 12 & f_{15}+f_{17}-f_{18} & Q_{xxz} \end{array}\right] = 5 \tag{3.89}$$

The rows 13 and 14 cannot be expressed as a linear combinations of the preceding rows so they both make the selection. However, the rows 15 and 16 can be expressed in terms of the rows 1, 2, 4, 6, 9, 13 and 1, 2, 3, 7, 8, 14, respectively.

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & \begin{array}{c} f_1+f_3+f_5+f_7+f_9+f_{10}+ \\ +f_{11}+f_{13}+f_{14}+f_{15}+f_{17}+f_{18} \end{array} & \rho \\ 2 & f_1+f_7-f_9+f_{10}+f_{15}-f_{17}+f_{18} & \rho u_x \\ 4 & f_5+f_{11}+f_{13}-f_{14}+f_{15}+f_{17}-f_{18} & \rho u_z \\ 6 & f_3+f_7+f_9+f_{10}+f_{11}+f_{13}+f_{14} & \Pi_{yy} \\ 9 & f_{15}-f_{17}-f_{18} & \Pi_{xz} \\ 13 & f_7-f_9+f_{10} & Q_{xyy} \\ 15 & f_{11}+f_{13}-f_{14} & Q_{yyz} \end{array}\right] = 6 \tag{3.90}$$

$$\text{rank}\left[\begin{array}{c|c|c} 1 & \begin{array}{c} f_1+f_3+f_5+f_7+f_9+f_{10}+ \\ +f_{11}+f_{13}+f_{14}+f_{15}+f_{17}+f_{18} \end{array} & \rho \\ 2 & f_1+f_7-f_9+f_{10}+f_{15}-f_{17}+f_{18} & \rho u_x \\ 3 & f_3+f_7+f_9-f_{10}+f_{11}-f_{13}+f_{14} & \rho u_y \\ 7 & f_5+f_{11}+f_{13}+f_{14}+f_{15}+f_{17}+f_{18} & \Pi_{zz} \\ 8 & f_7-f_9-f_{10} & \Pi_{xy} \\ 14 & f_{15}-f_{17}+f_{18} & Q_{xzz} \\ 16 & f_{11}-f_{13}+f_{14} & Q_{yzz} \end{array}\right] = 6 \tag{3.91}$$

For the velocity boundary, the main thing is to pick the three momentums, ρu_x, ρu_y and ρu_z, followed by the momentum fluxes and shear stresses, Π_{xx}, Π_{yy}, Π_{zz}, Π_{xy}, Π_{xz} and Π_{yz}. Then ideally choosing the third order moments before considering anything else. Therefore, the last three fourth order moments from Table 3.6, S_{xxyy}, S_{xxzz} and S_{yyzz}, are overlooked for now. There are enough moments to impose a boundary condition at the corner without them.

So, not counting the density, the next ten moment combinations are linearly independent. Together with the rows 13 and 14 they make the basic complete system of equations ready to be solved. Again, there is still some freedom left in choosing which moments will be included in the final system, but there is no clear reason why one would be chosen over the other. For instance, which is better out of the two in each case? Is it Q_{xxy} or Q_{xxz}, Q_{xyy} or Q_{yyz}, Q_{xzz} or Q_{yzz}? One could probably argue that there are two mathematically explainable options. From a symmetry point of view, either Q_{xxy}, Q_{yyz} and Q_{xzz} are selected or Q_{xxz}, Q_{xyy} and Q_{yzz}. This is as far as the mathematical reasoning can take. Any further choices are left to be made subjectively.

The final system of moment combinations includes the momentums, ρu_x, ρu_y and ρu_z, the momentum fluxes and shear stresses, Π_{xx}, Π_{yy}, Π_{zz}, Π_{xy}, Π_{xz} and Π_{yz}, and three higher order moments, Q_{xxy}, Q_{yyz} and Q_{xzz}. Alternatively, one can choose the other trio of the third order moments for the corner. Nevertheless, the system for solving the twelve unknowns at the low-south-east corner boundary is given below.

$$\operatorname{rank}\left[\begin{array}{c|c|c} 2 & f_1+f_7-f_9+f_{10}+f_{15}-f_{17}+f_{18} & \rho u_x \\ 3 & f_3+f_7+f_9-f_{10}+f_{11}-f_{13}+f_{14} & \rho u_y \\ 4 & f_5+f_{11}+f_{13}-f_{14}+f_{15}+f_{17}-f_{18} & \rho u_z \\ 5 & f_1+f_7+f_9+f_{10}+f_{15}+f_{17}+f_{18} & \Pi_{xx} \\ 6 & f_3+f_7+f_9+f_{10}+f_{11}+f_{13}+f_{14} & \Pi_{yy} \\ 7 & f_5+f_{11}+f_{13}+f_{14}+f_{15}+f_{17}+f_{18} & \Pi_{zz} \\ 8 & f_7-f_9-f_{10} & \Pi_{xy} \\ 9 & f_{15}-f_{17}-f_{18} & \Pi_{xz} \\ 10 & f_{11}-f_{13}-f_{14} & \Pi_{yz} \\ 11 & f_7+f_9-f_{10} & Q_{xxy} \\ 14 & f_{15}-f_{17}+f_{18} & Q_{xzz} \\ 15 & f_{11}+f_{13}-f_{14} & Q_{yyz} \end{array}\right] = 12. \tag{3.92}$$

Using the derived approximations for the second and third order moments at the boundaries, (3.65) and (3.82), and setting the low-south-west corner velocities

to U_x, U_y and U_z, the unknown incoming function values are given as

$$\begin{cases} f_1 &= \rho\left(\frac{1}{3}(2U_x + U_z - 1) + U_y(1 - U_x - U_y - U_z)\right) + f_2 + 2f_4 + 4f_8 + 4f_{12}, \\ f_3 &= \rho\left(\frac{1}{3}(2U_y + U_x - 1) + U_z(1 - U_x - U_y - U_z)\right) + f_4 + 2f_6 + 4f_{12} + 4f_{16}, \\ f_5 &= \rho\left(\frac{1}{3}(2U_z + U_y - 1) + U_x(1 - U_x - U_y - U_z)\right) + f_6 + 2f_2 + 4f_8 + 4f_{16}, \\ f_7 &= \frac{\rho}{2}\left(\frac{1}{3}(1 - U_z) + U_y\left(U_x + U_y + U_z - \frac{2}{3}\right)\right) - f_4 - f_8 - 2f_{12}, \\ f_9 &= \frac{\rho U_y}{2}\left(\frac{1}{3} - U_x\right) + f_8, \\ f_{10} &= \frac{\rho}{2}\left(\frac{1}{3}(1 - U_z) + U_y(U_y + U_z - 1)\right) - f_4 - f_8 - 2f_{12}, \\ f_{11} &= \frac{\rho}{2}\left(\frac{1}{3}(1 - U_x) + U_z\left(U_x + U_y + U_z - \frac{2}{3}\right)\right) - f_6 - f_{12} - 2f_{16}, \\ f_{13} &= \frac{\rho U_z}{2}\left(\frac{1}{3} - U_y\right) + f_{12}, \\ f_{14} &= \frac{\rho}{2}\left(\frac{1}{3}(1 - U_x) + U_z(U_x + U_z - 1)\right) - f_6 - f_{12} - 2f_{16}, \\ f_{15} &= \frac{\rho}{2}\left(\frac{1}{3}(1 - U_y) + U_x\left(U_x + U_y + U_z - \frac{2}{3}\right)\right) - f_2 - f_{16} - 2f_8, \\ f_{17} &= \frac{\rho}{2}\left(\frac{1}{3}(1 - U_y) + U_x(U_x + U_y - 1)\right) - f_2 - f_{16} - 2f_8, \\ f_{18} &= \frac{\rho U_x}{2}\left(\frac{1}{3} - U_z\right) + f_{16}. \end{cases} \tag{3.93}$$

This is a general form including all the velocity components. It simplifies significantly when specific cases are considered. For example, all the velocity terms disappear when the no-slip condition is imposed at the corner.

The density in the above equations (3.93) is calculated from the expression,

$$\rho - \rho U_x - \rho U_y - \rho U_z + \Pi_{xy} + \Pi_{xz} + \Pi_{yz} = f_0 + 2(f_2 + f_4 + f_6) + 4(f_8 + f_{12} + f_{16}), \tag{3.94}$$

which links the density with the known distribution functions. Rearranging (3.94) and substituting in the approximation values (3.66) for the moments gives the density at the low-south-west corner boundary:

$$\rho = \frac{f_0 + 2(f_2 + f_4 + f_6) + 4(f_8 + f_{12} + f_{16})}{1 - U_x(1 - U_y - U_z) - U_y(1 - U_z) - U_z}. \tag{3.95}$$

Corner pressure-velocity

As discussed earlier when considering a physical system with multiple pressure inlets, no-slip condition for velocity is applied at the point connecting the different pressure inlet boundaries. Similarly for a system with mixed type boundaries, both the density and velocity conditions have to be specified. If density is to be included into the linearly independent moment selection (3.92) then from (3.88) one of the moments must be left out. Velocities are set, which means that one of the shear stresses must be discarded. There is no clear preference for which is the odd one out as they form a closed symmetry group. A choice has to be made so Π_{yz} is discarded giving the following selection of moments:

$$\operatorname{rank}\left[\begin{array}{c|c|c} 1 & \begin{array}{c} f_1+f_3+f_5+f_7+f_9+f_{10}+ \\ +f_{11}+f_{13}+f_{14}+f_{15}+f_{17}+f_{18} \end{array} & \rho \\ 2 & f_1+f_7-f_9+f_{10}+f_{15}-f_{17}+f_{18} & \rho u_x \\ 3 & f_3+f_7+f_9-f_{10}+f_{11}-f_{13}+f_{14} & \rho u_y \\ 4 & f_5+f_{11}+f_{13}-f_{14}+f_{15}+f_{17}-f_{18} & \rho u_z \\ 5 & f_1+f_7+f_9+f_{10}+f_{15}+f_{17}+f_{18} & \Pi_{xx} \\ 6 & f_3+f_7+f_9+f_{10}+f_{11}+f_{13}+f_{14} & \Pi_{yy} \\ 7 & f_5+f_{11}+f_{13}+f_{14}+f_{15}+f_{17}+f_{18} & \Pi_{zz} \\ 8 & f_7-f_9-f_{10} & \Pi_{xy} \\ 9 & f_{15}-f_{17}-f_{18} & \Pi_{xz} \\ 11 & f_7+f_9-f_{10} & Q_{xxy} \\ 14 & f_{15}-f_{17}+f_{18} & Q_{xzz} \\ 15 & f_{11}+f_{13}-f_{14} & Q_{yyz} \end{array}\right] = 12. \tag{3.96}$$

Setting the density to ρ_0 and velocities to U_x, U_y and U_z, the unknown distribution functions for pressure-velocity at the low-south-west corner boundary are

given as

$$
\begin{cases}
f_1 &= \rho_0\left(\frac{1}{3}(2-2U_z-U_x)+U_xU_z-U_y^2\right)-f_0-f_2-2f_6-4f_{16},\\
f_3 &= \rho_0\left(\frac{1}{3}(2-2U_x-U_y)+U_xU_y-U_z^2\right)-f_0-f_4-2f_2-4f_8,\\
f_5 &= \rho_0\left(\frac{1}{3}(2U_z+U_y-1)+U_x(1-U_x-U_y-U_z)\right)+\\
&\quad +f_6+2f_2+4f_8+4f_{16},\\
f_7 &= \frac{\rho_0}{2}\left(\frac{1}{3}(2U_z+U_y-2)+U_x-U_xU_z+U_y^2\right)+\\
&\quad +\frac{1}{2}f_0+f_2+f_6+f_8+2f_{16},\\
f_9 &= \frac{\rho_0U_y}{2}\left(\frac{1}{3}-U_x\right)+f_8,\\
f_{10} &= \frac{\rho_0}{2}\left(\frac{1}{3}(2U_z-2)+U_x-U_xU_y-U_xU_z+U_y^2\right)+\\
&\quad +\frac{1}{2}f_0+f_2+f_6+f_8+2f_{16},\\
f_{11} &= \frac{\rho_0}{2}\left(\frac{1}{3}(2U_x+U_z-2)+U_y-U_xU_y+U_z^2\right)+\\
&\quad +\frac{1}{2}f_0+f_2+f_4+f_8+2f_{12},\\
f_{13} &= \frac{\rho_0}{2}\left(1-\frac{2}{3}U_z-U_y-U_x(1-U_y-U_z)\right)+\\
&\quad -\frac{1}{2}f_0-f_2-f_4-f_6-f_{12}-2f_8-2f_{16},\\
f_{14} &= \frac{\rho_0}{2}\left(\frac{1}{3}(1-U_x)+U_z(U_x+U_z-1)\right)-f_6-f_{12}-2f_{16},\\
f_{15} &= \frac{\rho_0}{2}\left(\frac{1}{3}(1-U_y)+U_x\left(U_x+U_y+U_z-\frac{2}{3}\right)\right)-f_2-f_{16}-2f_8,\\
f_{17} &= \frac{\rho_0}{2}\left(\frac{1}{3}(1-U_y)+U_x(U_x+U_y-1)\right)-f_2-f_{16}-2f_8,\\
f_{18} &= \frac{\rho_0U_x}{2}\left(\frac{1}{3}-U_z\right)+f_{16}.
\end{cases}
\tag{3.97}
$$

Corner pressure

To obtain the unknown distribution function values for the multiple pressure inlet at the low-south-west corner all the velocities in the above system of equations have to be set to zero to realise the no-slip boundary condition for velocity. The system (3.97) simplifies to

$$\begin{cases} f_1 &= \frac{2\rho_0}{3} - f_0 - f_2 - 2f_6 - 4f_{16}, \\ f_3 &= \frac{2\rho_0}{3} - f_0 - f_4 - 2f_2 - 4f_8, \\ f_5 &= -\frac{\rho_0}{3} + f_6 + 2f_2 + 4f_8 + 4f_{16}, \\ f_7 &= -\frac{\rho_0}{3} + \frac{1}{2}f_0 + f_2 + f_6 + f_8 + 2f_{16}, \\ f_9 &= f_8, \\ f_{10} &= -\frac{\rho_0}{3} + \frac{1}{2}f_0 + f_2 + f_6 + f_8 + 2f_{16}, \\ f_{11} &= -\frac{\rho_0}{3} + \frac{1}{2}f_0 + f_2 + f_4 + f_8 + 2f_{12}, \\ f_{13} &= \frac{\rho_0}{2} - \frac{1}{2}f_0 - f_2 - f_4 - f_6 - f_{12} - 2f_8 - 2f_{16}, \\ f_{14} &= \frac{\rho_0}{6} - f_6 - f_{12} - 2f_{16}, \\ f_{15} &= \frac{\rho_0}{6} - f_2 - f_{16} - 2f_8, \\ f_{17} &= \frac{\rho_0}{6} - f_2 - f_{16} - 2f_8, \\ f_{18} &= f_{16}. \end{cases} \tag{3.98}$$

3.7 Collision schemes

Lattice Boltzmann method comes with a range of collision schemes that vary in complexity, but also accuracy and stability.

3.7.1 Single-relaxation-time model

The simplest and most popular is the single relaxation time scheme or the lattice BGK as mentioned earlier in Section 3.2. It uses a single parameter to relax all the populations to their equilibria in the collision step,

$$f_i^* = -\frac{1}{\tau}(f_i - f_i^{eq}) + F_i \tag{3.99}$$

where f_i^* is the post collision distribution function and F_i is an external source term. The relaxation rate is directly related to the viscosity through (3.38), which means that the accuracy and stability of the method are affected by the choice of the fluid viscosity. For example, it has been reported that the spatial discretisation error for the BGK collision operator is proportional to $\frac{1}{4}\left(\tau - \frac{1}{2}\Delta t\right)^2$. Depending on the type of the problem, there are certain values of τ that allow for the removal of the spatial truncation error contributions to the solution [145], but generally if this is to be avoided, one has to use an improved collision scheme, such as TRT or MRT.

3.7.2 Two-relaxation-time model

As the name suggests, TRT uses two relaxation rates. One is directly linked to the viscosity of the physical system, while the other is a free parameter that can be fine-tuned for optimal accuracy and stability. Because the truncation errors depend on a certain combination of the two parameters, the viscosity is no longer directly connected to the accuracy of the system, as it is in the BGK model. This

combination is referred to as the magic parameter, and it is expressed as

$$\Lambda = \left(\frac{1}{\omega^+} - \frac{1}{2}\right)\left(\frac{1}{\omega^-} - \frac{1}{2}\right), \tag{3.100}$$

where ω^+ is the symmetric relaxation rate that is related to viscosity, and ω^- is the antisymmetric rate that can be freely adjusted for stability purposes. Note that the relaxation rate and time have an inverse proportionality, $\omega^\pm = \frac{1}{\tau^\pm}$. This separation of the symmetric and antisymmetric elements is the main idea of the TRT model. Because all the velocity sets are symmetric, distribution functions can be paired up in terms of their velocities as $c_i = -c_{\bar{i}}$ forming a so-called link [146]. Any link can be decomposed into its symmetric and antisymmetric parts as

$$\begin{aligned} f_i^+ &= \frac{1}{2}(f_i + f_{\bar{i}}), \qquad f_i^- = \frac{1}{2}(f_i - f_{\bar{i}}), \\ f_i^{eq+} &= \frac{1}{2}(f_i^{eq} + f_{\bar{i}}^{eq}), \quad f_i^{eq-} = \frac{1}{2}(f_i^{eq} - f_{\bar{i}}^{eq}). \end{aligned} \tag{3.101}$$

The rest population is its own opposite so it only has a symmetric part, $f_i^+ = f_i$ and $f_i^{eq+} = f_i^{eq}$, and a zero antisymmetric component, $f_i^- = 0$ and $f_i^{eq-} = 0$. Using the introduced components, the TRT post-collision distribution function can be written as

$$f_i^* = -\frac{1}{\tau^+}(f_i^+ - f_i^{eq+}) - \frac{1}{\tau^-}(f_i^- - f_i^{eq-}) + F_i. \tag{3.102}$$

The relaxation time related to the viscosity is the symmetric one:

$$\nu = c_s^2\left(\tau^+ - \frac{1}{2}\right). \tag{3.103}$$

As stated in [115], because the collision step is only performed for one half of the populations, TRT is computationally as efficient as BGK, but with an improved control over stability and accuracy. Moreover, TRT is preferred over BGK in simulations dealing with large boundary areas, like porous media flows. Another example of systems with large boundary areas is microstructure solidification where the liquid flows past the growing dendrites due to convection, as discussed in Section 2.2.

3.7.3 Multiple-relaxation-time model

Originally proposed by d'Humieres [147], MRT is the most general collision scheme considered in this thesis. It has as many relaxation parameters as there are distribution functions for a particular velocity set. In the MRT collision scheme, the relaxation is performed in the moment space rather than the kinetic space as it is in BGK and TRT. It means that the distribution functions need to be transformed into the moment space and then transformed back to the kinetic space to perform the streaming step. The transformation can be written down using matrix form as

$$\mathbf{m} = \mathbf{M}\mathbf{f}, \tag{3.104}$$

where $\mathbf{m}$ is the moment column vector of length q as in DdQq, $\mathbf{M}$ is the transformation matrix of size $q \times q$, and $\mathbf{f}$ is the distribution function column vector. $\mathbf{M}$ can be found using either Hermite polynomials or Gram−Schmidt orthogonalisation, the latter of which is a more common procedure. Sparing the derivation

process details here, the post-collision distribution function can be written as

$$\mathbf{f}^{*} = -\mathbf{M}^{-1}\mathbf{S}\mathbf{M}(\mathbf{f} - \mathbf{f}^{eq}) + \mathbf{M}^{-1}\left(\mathbf{I} - \frac{\mathbf{S}}{2}\right)\mathbf{M}\mathbf{F}, \tag{3.105}$$

where $\mathbf{M^{-1}}$ is the inverse of the transformation matrix, and $\mathbf{S}$ is the relaxation matrix. If the Gram−Schmidt procedure is used, the relaxation matrix is a diagonal one,

$$\mathbf{S} = \mathrm{diag}(s_0, s_1, \ldots, s_{q-1}), \tag{3.106}$$

implying that every moment $\mathbf{m}_i$ relaxes to its equilibrium at a corresponding rate s_i. There is only a handful of physically meaningful parameters. Conserved quantities have a zero relaxation rate. That applies to mass density and momentum density when there are no source terms present. Relaxation rate for the shear stress components is the one linked with the shear viscosity, and another one linked to the bulk viscosity. All the rest can be tuned freely to improve stability. $\mathbf{I}$ is the identity matrix, and $\mathbf{F}$ is the external force term column vector. Forcing schemes will be discussed in Section 3.8.

3.7.4 Central-moments-based LBM

The cascaded or central-moments-based LBM (CLBM) was originally proposed by Geier *et al.* [148] to overcome the shortcomings of the standard BGK and MRT collision schemes and achieving better stability and higher degree of Galilean invariance. The underlying idea of the CLBM is to perform the collision step in a reference frame that has been shifted by the local hydrodynamic fluid velocity. Similarly to the MRT sceme, the post-collision distribution function can be

written as [149]

$$\mathbf{f}^{*} = \mathbf{M}^{-1}\mathbf{N}^{-1}\left((\mathbf{I}-\mathbf{S})\boldsymbol{\Gamma} + \mathbf{S}\boldsymbol{\Gamma}^{eq} + \left(\mathbf{I}-\frac{\mathbf{S}}{2}\right)\boldsymbol{\Xi}\right), \tag{3.107}$$

where $\mathbf{N}$ is the shift matrix, $\boldsymbol{\Gamma}$ is the distribution function in central-moment space and $\boldsymbol{\Xi}$ is the forcing source term in central-moment space. The factor $(\mathbf{I}-\mathbf{S}/2)$ in front of the forcing source term is due to removing implicitness after the trapezoidal integration of the collision part of the LBE. The transformation matrix $\mathbf{M}$ maps the distribution functions $\mathbf{f}$ into the raw moment space where they are shifted by the matrix $\mathbf{N}$ to give central moments as $\boldsymbol{\Gamma} = \mathbf{NMf}$. The forcing term follows the same path, $\boldsymbol{\Xi} = \mathbf{NMF}$. Fei and Luo [150] examined the different forcing schemes suggested by other authors and proposed a more consistent one with improved accuracy and isotropy. Furthermore, the CLBM has been successfully used to simulate multiphase [151], shallow water [152] and thermal flows [149] highlighting the stability feature at low viscosities.

3.7.5 Overview

Several lattice Boltzmann collision schemes have been described above highlighting their advantages and drawbacks to choose a potential candidate to be used in large-scale simulations of dendritic solidification. Although BGK is the simplest collision scheme with only a single relaxation time, it fails to offer a numerical stability control not connected to the physical system. MRT and CLBM overcomes this drawback by offering individual relaxation rates for each hydrodynamic moment and several parameters for stability control. Moreover, the CLBM offers better numerical stability and Galilean invariance than MRT [148]. However, the

added stability control comes at a cost of an increased computational time [153]. A compromise of the methods described is the TRT collision scheme. It can be as efficient as BGK, while at the same time offering a stability and accuracy control that is a simplified version of the MRT one.

3.8 Forcing schemes

Force implementation is not straightforward in lattice Boltzmann. The continuous Boltzmann equation is written as

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla_x f + \mathbf{F} \cdot \nabla_\xi f = \Omega(f), \tag{3.108}$$

where $f = f(\mathbf{x}, \boldsymbol{\xi}, t)$ and $\mathbf{F}$ is a body force. Discretisation of (3.108) can be complicated with the force term present, however one can simply add a source term to the right hand side of the lattice Boltzmann equation as

$$f_i(\mathbf{x} + \mathbf{c}_i \delta t, t + \delta t) - f_i(\mathbf{x}, t) = \Omega_i(f(\mathbf{x}, t)) + F_i \delta t, \tag{3.109}$$

where the expression for F_i depends on the discretisation order and derivation process of a particular scheme. Several forcing schemes have been proposed over the years. The simplest one that satisfies mass and momentum conservation given by,

$$\sum_i F_i = 0, \qquad \sum_i \mathbf{c}_i F_i = \mathbf{F}, \tag{3.110}$$

is the one proposed by Luo [154]:

$$F_i = w_i \frac{\mathbf{c}_i \cdot \mathbf{F}}{c_s^2}. \tag{3.111}$$

Unfortunately, this simple forcing scheme has an unwanted residual of

$$\Delta_{res} = \left(\tau - \frac{1}{2}\right) \nabla \cdot (\mathbf{uF} + \mathbf{Fu}). \tag{3.112}$$

This is to do with the force contribution to the momentum flux [155; 156]. To correct this inaccuracy, the forcing term must meet the following criteria,

$$\begin{aligned} \sum_i F_i &= 0, \\ \sum_i \mathbf{c}_i F_i &= \mathbf{F}, \\ \sum_i \mathbf{c}_i \mathbf{c}_i F_i &= \mathbf{uF} + \mathbf{Fu}, \end{aligned} \tag{3.113}$$

where the first criterion accounts for a mass source, the second one denotes the momentum source (external force), and the third one prevents the spurious term (3.112) from appearing in the momentum expression when recovering the Navier−Stokes equation.

The forcing schemes that do meet the criteria in (3.113) have been proposed by He, Shan and Doolen [157], and Luo [158], for example.

The HSD scheme [157] (named after the authors) has the source term expressed as

$$F_i = \left(1 - \frac{1}{2\tau}\right) \frac{(\mathbf{c}_i - \mathbf{u}) \cdot \mathbf{F}}{\rho c_s^2} f_i^{eq}. \tag{3.114}$$

It can be shown that it satisfies the constraints if the term of order $O(u^3)$ are neglected [101]. The velocity is calculated from

$$\rho \mathbf{u} = \sum_i \mathbf{c}_i f_i + \frac{\delta_t}{2} \mathbf{F}, \tag{3.115}$$

which is also the velocity for the equilibrium $f_i^{eq}(\rho, \mathbf{u})$.

Luo's forcing term [158] can be written as

$$F_i = w_i \left(\frac{\mathbf{c}_i - \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})\mathbf{c}_i}{c_s^4} \right) \cdot \mathbf{F}. \tag{3.116}$$

The fluid velocity is calculated without using the half contribution of the force,

$$\rho \mathbf{u} = \sum_i \mathbf{c}_i f_i, \tag{3.117}$$

and the same velocity is used to calculate the equilibrium distribution function $f_i^{eq}(\rho, \mathbf{u})$.

3.9 Summary

In this chapter, the lattice Boltzmann method has been described. Originating from the LGCA, it still embodies several of its great properties: the LBM has a linear advection term, inherent parallelism, no Poisson solver for pressure, easy handling of complex geometries. It can be shown to lead to the weakly compressible NSE through the Chapman−Enskog multisacle expansion. Furthermore, by neglecting the small density variations, the incompressible NSE can be recovered. All these properties make it a serious contender for an alternative numerical method.

In 3D, there are three common choices of the velocity sets, D3Q15, D3Q19 and D3Q27. The fewer the velocities, the more computationally efficient the model. On the other hand, the more discrete velocities, the more isotropic the lattice and the more stable the simulations. The D3Q19 lattice is somewhere in the middle between the two extremes so it is chosen to be used in the calculations in this

thesis. Moreover, it is a direct extension of the popular D2Q9 model. Because of that, the results can be easily compared between the two models.

Stability and accuracy of the method have been described seeking the optimal stability regimes of the flow and recognising the errors affecting the solution. The stability of the BGK collision scheme strictly depends on the relaxation time τ, which is directly linked to the fluid viscosity. To avoid this unphysical setup, the TRT scheme is considered. It has two separate relaxation times, one for the physical viscosity and the other for controlling the stability. Although, the MRT scheme offers more thorough control, it comes at a cost of being computationally heavier than the BGK. Due to the TRT model offering almost the same stability control as the MRT scheme and being as computationally efficient as the BGK scheme, it is the optimal choice of the collision schemes considered here.

Forcing schemes have to meet certain criteria to avoid producing non-zero residuals in the momentum equation. Both schemes described here, the HSD and Luo's, satisfy the conditions given in (3.113) so it is down to the user's preference to choose one over the other.

It is not a coincidence that the biggest part of the section is dedicated to the boundary methods. First of all, different lattice Boltzmann boundary schemes are described and their merits and faults are listed. The kinetic schemes are simple and good for complex geometries both in 2D and 3D, but they are normally not on-site and are τ-dependent. The non-equilibrium bounce-back scheme is a simple hydrodynamic scheme in 2D, but it gets quite complicated in 3D with all the momentum corrections and the rest particle distribution function recalculations. In addition, the moment analysis has revealed that the boundary methods using some variation of the bounce-back rule are imposing constraints on the third

order moments. For the standard bounce-back and Zou and He scheme in 2D it has been shown in [2], and for the 3D non-equilibrium bounce-back it is shown in Section 3.6.3. The Moment Method is a hydrodynamic scheme, in which the constraints are set directly on the velocity moments. There is no connection to the kinetic theory, which means that the macroscopic boundary conditions are respected and treated properly.

Second of all, by examining the faults of the current 3D boundary schemes, a new 3D boundary method for the D3Q19 model is proposed that is an extension of the 2D Moment Method and that overcomes these issues. The derivation process is fully covered including the description of the face, edge and corner boundary conditions for both velocity and pressure.

Chapter 4

THE NUMERICAL METHOD

4.1 Introduction

In this chapter, the numerical algorithm will be described and the parallelisation aspect of it will be covered. To assure compatibility with the in-house parallel software TESA for simulating microstructure solidification, which is written in Fortran language, the LB code is also written in Fortran. MPI libraries are used for parallelisation on CPUs and CUDA environment is used for the code acceleration on GPUs.

4.2 Structure of the LB algorithm

The LB algorithm can be sequentially written out using subroutines or processes. As the flow diagram shows in Figure 4.1, the LB algorithm consists of preparatory processes like unit scaling and initialisation, main time loop that comprises calculations of collisions, streaming, boundary conditions and macroscopic variables, and an I/O process that periodically outputs data to a file. These subroutines are

discussed in more detail in the next sections providing snippets of pseudocodes where appropriate.

4.2.1 LB unit scaling

Lattice Boltzmann uses non-dimensional units, commonly called the lattice units, where both time and space are discretized by unit steps. Density is also assumed to be unity, and a unit mass follows from the latter.

By knowing the physical entities such as length, velocity, viscosity, temperature, etc., one can work out the correct values for parameters and variables in LB units. As many values in LB can be chosen freely as there are variable dimensions being considered, that is, metre, second, kilogram, kelvin and so on. The rest can be derived accordingly. It is normally desired to have an adjustable domain size so node count N in LB is usually one of the chosen entities. Another one that contains the time dimension is picked depending on the physical problem or the occasion, whether tunable velocity or viscosity is of interest. Further reasoning might be the constraints imposed on these two problem defining properties. For velocity, it is the stability of the method itself, which is limited to low Mach numbers. For viscosity, it is the positivity constraint, but also the fact that it should be kept small to reduce any error arising from the potential use of the bounce-back rule or simply to achieve higher Reynolds numbers on smaller grids. That is why one might find it necessary to freely manipulate this momentum diffusion parameter. The main idea here is to get time scaling because the physical grid step size has already been worked out.

$$\Delta x = \frac{L}{N_{LB}}, \qquad \Delta t = \frac{U_{LB}}{U} \cdot \Delta x, \qquad \Delta t = \frac{\nu_{LB}}{\nu} \cdot \Delta x^2. \tag{4.1}$$

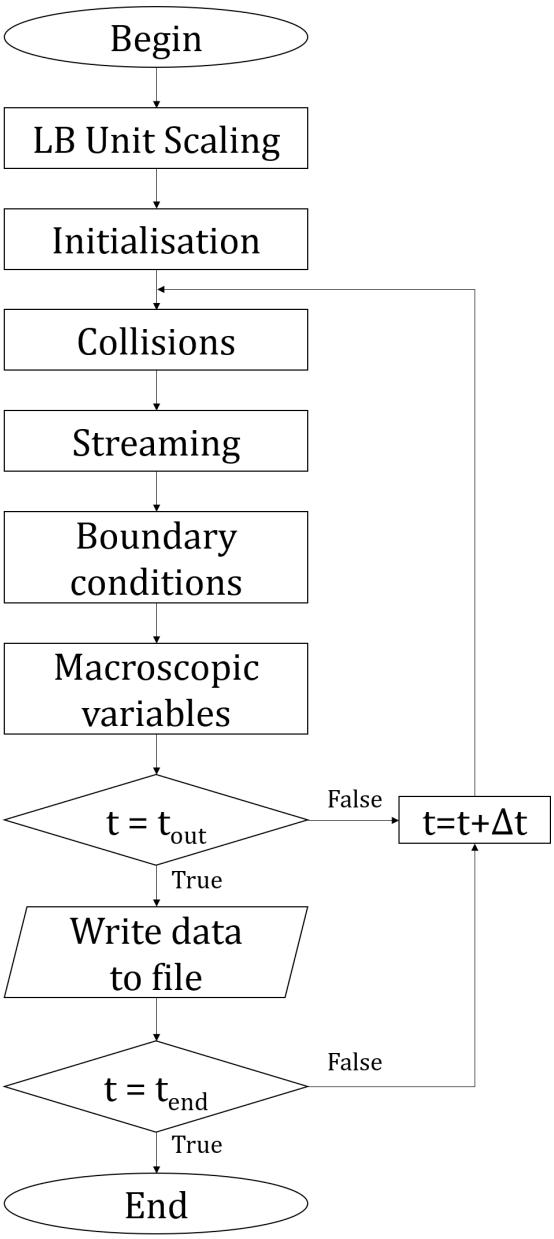


Figure 4.1: Flow diagram of the LB algorithm.

Table 4.1: Unit conversion.

Physical parameters	LB parameters		
$L = 10^{-2}$ m	$N_{LB} = 10^2$	$\Delta x = \frac{L}{N_{LB}}$	$\Delta x = 10^{-4}$ m
$\nu = 10^{-6}$ m^2 s^{-1}	$\nu_{LB} = 10^{-1}$	$\Delta t = \frac{\nu_{LB}}{\nu} \cdot \Delta x^2$	$\Delta t = 10^{-3}$ s
$U = 10^{-2}$ m s^{-1}	$U_{LB} = ?$	$U_{LB} = U \cdot \frac{\Delta t}{\Delta x}$	$U_{LB} = 10^{-1}$
$\mathrm{Re} = \frac{UL}{\nu} = 100$	$\stackrel{?}{=}$	$\mathrm{Re_{LB}} = \frac{U_{LB} N_{LB}}{\nu_{LB}} = 100$	

An easy way to make sure that the scaling is done correctly is to compare a problem specific dimensionless characteristic such as the flow Reynolds number. It should give the same value in physical units and in lattice units. A problem with set physical parameters is given in Table 4.1. Because only kinematic quantities, namely, time and length, are considered, the number of grid points and the LB viscosity are chosen as appropriate giving the time step Δt and spatial step size Δx. Using these discretisation parameters, the velocity value can then be derived. The same Reynolds number is obtained in both continuum and the LB approach, which serves as an indicator suggesting that the scaling is done correctly.

The scaling can be written in code as shown in pseudocode in Listing 4.1. The length scale `dx` is calculated on line 1 assuming the longest dimension of the uniform grid (`dx=dy=dz`) is `L` metres long. The diffusive scaling is being used to obtain the time scale `dt` from the chosen LB viscosity `nu` and the physical kinematic viscosity `knu` on line 2. Velocity and force are rescaled to lattice units on lines 3-8, where the unit density `r0=1` and the physical density `rho` are used for the force rescaling. Note that repeated multiplication is used throughout the code instead of whole powers because it is computationally less expensive.

LISTINGS 4.1: Unit scaling

```
1 dx = L/(max(xm,ym,zm)+1) !length scale
2 dt = (nu/knu)*dx*dx     !time scale
3 u = u*dt/dx           !x velocity
4 v = v*dt/dx           !y velocity
5 w = w*dt/dx           !z velocity
6 fx = fx*r0/(rho*dx)*dt*dt !x force
7 fy = fy*r0/(rho*dx)*dt*dt !y force
8 fz = fz*r0/(rho*dx)*dt*dt !z force
```

4.2.2 Initialisation

The equilibrium initialisation of the distribution function is used in the LB algorithm. It takes into account the previous velocity and pressure fields and calculates the initial distribution functions in the domain. Listing 4.2 shows the typed out pseudocode version of (4.2) in 3D where the fluid and lattice velocities are broken down into the components $\mathbf{u} = \{u, v, w\}$ and $\mathbf{c}_i = \{c_{ix}, c_{iy}, c_{iz}\}$.

$$f_i^{eq} = w_i \rho \left(1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{u^2}{2c_s^2}\right), \tag{4.2}$$

The index i represents the discretisation in the velocity space whose range depends on the dimension and the lattice being used, here $i = 0 - 18$. To indicate that `(x,y,z)` indices have been dropped inside some of the `do` loops, the bold font is used. The density `r` on line 2 is included to recover the pressure field from the previous calculations, otherwise the pressure is reset and the obstacle boundaries might not be respected. The flag `stokes` on line 3 allows to neglect the nonlinear

terms in the equilibrium distribution function when simulating flows in Stokes regime.

LISTINGS 4.2: Initialisation

```
do loop through x,y,z,i
f(i) = wt(i)*(r*(1&
 +3*(cx(i)*u+cy(i)*V+cz(i)*W)+(1-stokes)*(0.5*&
((3*(cx(i)*u+cy(i)*V+cz(i)*W))&
*(3*(cx(i)*u+cy(i)*V+cz(i)*W)))&
-1.5*(u*u+V*V+W*W)))
end do
```

4.2.3 Collision and streaming

Having originated from the LGCA, the lattice Boltzmann equation is already given in a discretised form and can simply be written as

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t) = f_i(\mathbf{x}, t) - \frac{1}{\tau}\left(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right) + F_i. \tag{4.3}$$

Time and space are both normally discretised by using unit steps so $\Delta t = 1$ and $\Delta x = c\Delta t = 1$. It must be noted that the Δt and Δx used within the main loop are the non-dimensional time and length scales of the LBM. To avoid confusion, (4.3) can be rewritten as

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = f_i(\mathbf{x}, t) - \frac{1}{\tau}\left(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right) + F_i. \tag{4.4}$$

The left side describes the streaming process and the right side describes the collision process. If we introduce a post-collision distribution function f_i^* then

(4.4) can be split into two parts and rewritten as

$$\begin{aligned} &\text{collision:} && f_i^*(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{1}{\tau}\left(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)\right) + F_i, \\ &\text{streaming:} && f_i(\mathbf{x}+\mathbf{c}_i, t+1) = f_i^*(\mathbf{x},t). \end{aligned} \tag{4.5}$$

The collision process in (4.5) only uses a single relaxation time, however two or more relaxation times are possible. Moreover, a forcing scheme might be considered. Listings 4.4 and 4.5 show how the collision process changes in complexity when more advanced schemes are considered. The collision subroutines have a built-in support for the Stokes regime flows through the flag `stokes`. Some of the predefined variables used in the code are shown in Listing 4.3 including the relaxation parameter `omega` that gets calculated from the viscosity `nu`.

LISTINGS 4.3: Predefined variables

```
1 omega = 1/(3*nu+0.5)
2 tau = 1/omega-0.5
3 tauJ = 1/(4*tau)!Magic parameter lambda=tau*tauJ=1/4
4 cu = cx*u+cy*v+cz*w
5 cu2 = cu*cu
6 vel = u*u+v*v+w*w
7 uf = u*fx+v*fy+w*fz
8 cf = cx*fx+cy*fy+cz*fz
```

In Listing 4.4, the BGK collision scheme is written down in code including two options for the forcing term – the HSD [157] or the Luo's forcing scheme [158].

LISTINGS 4.4: BGK collision

```
1 do loop through x,y,z,i
2 f(i) = (1-omega)*f(i)+omega
     *wt(i)*r*(1+3*cu(i)+(1-stokes)*(4.5*cu(i)*cu(i)-1.5*vel))&
3 !HSD 1998 Force (Comment out if not used)
4 +(1-omega/2)*3*(cf(i)-uf)
     *wt(i)*r*(1+3*cu(i)+(1-stokes)*(4.5*cu(i)*cu(i)-1.5*vel))
5 !Luo 1998 Force (Comment out if not used)
6 +(1-omega/2)*wt(i)*r*3*(cf(i)-uf+3*cu(i)*cf(i))
7 end do
```

Listing 4.5 shows the TRT collision scheme, see Section 3.7.2. Because it is dealing with the symmetric and antisymmetric parts of the distribution function (line 4), it has more than double the lines compared to the BGK collision subroutine. The interesting thing about TRT is that the calculations are only required for half of the distribution functions, which makes TRT as computationally efficient as BGK. The chosen numbering of the lattice velocities allows for addressing every link in a simple way (line 3). Lines 7-11 show the collision calculation using the HSD forcing term. Luo's forcing scheme is on lines 13-17. Lines 18-22 is where the post collision distribution functions are derived. Listing 4.6 shows the streaming process. The propagation of information is written in the reference frame of the post-streaming distribution functions. The loop does not contain `f(x,y,z,0)` because the rest particle velocity does not get streamed. All the `if` statements are in place to stay within the computer memory bounds and not get a memory access violation. If it were not for the memory issues, the streaming subroutine could be as simple as shown in Listing 4.7.

LISTINGS 4.5: TRT collision

```
1  do loop through x,y,z
2  fp(0) = f(0)
3  do i = 1, 18, 2
4  fp(i) = (f(i)+f(opp(i)))/2; fm(i) = (f(i)-f(opp(i)))/2
5  enddo
6  !HSD 1998 Force (Comment out if not used)
7  fpe(0) = -tau*r
       *((1+(1-stokes)*(-1.5*vel))*uf)+r/3*(1+(1-stokes)*(-1.5*vel))
8  do i = 1, 18, 2
9  fpe(i) = -tau*3*wt(i)*r
       *((1+(1-stokes)*(4.5*cu2(i)-1.5*vel))*uf-3*cu(i)*cf(i))
       +wt(i)*r*(1+(1-stokes)*(4.5*cu2(i)-1.5*vel))
10 fme(i) = tauJ*3*wt(i)*r
       *((1+(1-stokes)*(4.5*cu2(i)-1.5*vel))*cf(i)-3*cu(i)*uf)
       +wt(i)*r*3*cu(i)
11 enddo!HSD end
12 !Luo 1998 Force (Comment out if not used)
13 fpe(0) = -tau*r*(uf)+r/3*(1+(1-stokes)*(-1.5*vel))
14 do i = 1, 18, 2
15 fpe(i) = -tau*3*wt(i)*r*(uf-3*cu(i)*cf(i))
       +wt(i)*r*(1+(1-stokes)*(4.5*cu2(i)-1.5*vel))
16 fme(i) = tauJ*3*wt(i)*r*(cf(i))+wt(i)*r*3*cu(i)
17 enddo!Luo end
18 f(0) = f(0)-(fp(0)-fpe(0))/(tau+0.5)
19 do i = 1, 18, 2
20 f(i) = f(i)-(fp(i)-fpe(i))/(tau+0.5)-(fm(i)-fme(i))/(tauJ+0.5)
21 f(i+1) = f(i+1)-(fp(i)-fpe(i))/(tau+0.5)-(fm(i)-fme(i))/(tauJ+0.5)
22 enddo
23 end do
```

LISTINGS 4.6: Streaming

```
do loop through x,y,z
if(x.gt.0)     f(x,y,z,1) = f(x-1,y,z,1)
if(x.lt.xm+1)  f(x,y,z,2) = f(x+1,y,z,2)
if(y.gt.0)     f(x,y,z,3) = f(x,y-1,z,3)
if(y.lt.ym+1)  f(x,y,z,4) = f(x,y+1,z,4)
if(z.gt.0)     f(x,y,z,5) = f(x,y,z-1,5)
if(z.lt.zm+1)  f(x,y,z,6) = f(x,y,z+1,6)
if(x.gt.0.and.y.gt.0)        f(x,y,z, 7) = f(x-1,y-1,z,7)
if(x.lt.xm+1.and.y.lt.ym+1) f(x,y,z, 8) = f(x+1,y+1,z,8)
if(x.lt.xm+1.and.y.gt.0)     f(x,y,z, 9) = f(x+1,y-1,z,9)
if(x.gt.0.and.y.lt.ym+1)     f(x,y,z,10) = f(x-1,y+1,z,10)
if(y.gt.0.and.z.gt.0)        f(x,y,z,11) = f(x,y-1,z-1,11)
if(y.lt.ym+1.and.z.lt.zm+1) f(x,y,z,12) = f(x,y+1,z+1,12)
if(y.lt.ym+1.and.z.gt.0)     f(x,y,z,13) = f(x,y+1,z-1,13)
if(y.gt.0.and.z.lt.zm+1)     f(x,y,z,14) = f(x,y-1,z+1,14)
if(x.gt.0.and.z.gt.0)        f(x,y,z,15) = f(x-1,y,z-1,15)
if(x.lt.xm+1.and.z.lt.zm+1) f(x,y,z,16) = f(x+1,y,z+1,16)
if(x.lt.xm+1.and.z.gt.0)     f(x,y,z,17) = f(x+1,y,z-1,17)
if(x.gt.0.and.z.lt.zm+1)     f(x,y,z,18) = f(x-1,y,z+1,18)
end do
```

LISTINGS 4.7: Simple streaming

```
do loop through x,y,z,i
f(x,y,z,i) = f(x-cx(i),y-cy(i),z-cz(i),i)
end do
```

4.2.4 Boundary conditions

Depending on what type of the problem the solver is going to be used for, calls to certain subroutines that handle boundary conditions can be made.

The simplest boundary condition is the bounce-back scheme which is still very simple and straightforward to type in code, see Listing 4.8. Because the collisions are still carried out on the boundary nodes, the scheme used in some simulations is actually the modified bounce-back rule.

LISTINGS 4.8: Bounce-back scheme

```
1 do loop through x,y,z,i
2 if (obstacle) then
3 f(x,y,z,i) = fp(x,y,z,opp(i))
4 endif
5 end do
```

The half-way bounce-back rule is very similar to the streaming process. The only differences are copying the information to the opposite velocity direction of the current node (lines 3-20) and acting on the solid boundaries (the use of the flag `obstacle` on line 2), as shown in Listing 4.9. Despite being more complicated than the standard bounce-back scheme, it is oftentimes preferred because of its second-order accuracy.

LISTINGS 4.9: Half-way bounce-back

```
1  do loop through x,y,z
2  if (obstacle) then
3  if(x.gt.0)     f(x,y,z,opp(1)) = f(x-1,y,z,1)
4  if(x.lt.xm+1)  f(x,y,z,opp(2)) = f(x+1,y,z,2)
5  if(y.gt.0)     f(x,y,z,opp(3)) = f(x,y-1,z,3)
6  if(y.lt.ym+1)  f(x,y,z,opp(4)) = f(x,y+1,z,4)
7  if(z.gt.0)     f(x,y,z,opp(5)) = f(x,y,z-1,5)
8  if(z.lt.zm+1)  f(x,y,z,opp(6)) = f(x,y,z+1,6)
9  if(x.gt.0.and.y.gt.0)        f(x,y,z,opp( 7)) = f(x-1,y-1,z,7)
10 if(x.lt.xm+1.and.y.lt.ym+1) f(x,y,z,opp( 8)) = f(x+1,y+1,z,8)
11 if(x.lt.xm+1.and.y.gt.0)     f(x,y,z,opp( 9)) = f(x+1,y-1,z,9)
12 if(x.gt.0.and.y.lt.ym+1)     f(x,y,z,opp(10)) = f(x-1,y+1,z,10)
13 if(y.gt.0.and.z.gt.0)        f(x,y,z,opp(11)) = f(x,y-1,z-1,11)
14 if(y.lt.ym+1.and.z.lt.zm+1) f(x,y,z,opp(12)) = f(x,y+1,z+1,12)
15 if(y.lt.ym+1.and.z.gt.0)     f(x,y,z,opp(13)) = f(x,y+1,z-1,13)
16 if(y.gt.0.and.z.lt.zm+1)     f(x,y,z,opp(14)) = f(x,y-1,z+1,14)
17 if(x.gt.0.and.z.gt.0)        f(x,y,z,opp(15)) = f(x-1,y,z-1,15)
18 if(x.lt.xm+1.and.z.lt.zm+1) f(x,y,z,opp(16)) = f(x+1,y,z+1,16)
19 if(x.lt.xm+1.and.z.gt.0)     f(x,y,z,opp(17)) = f(x+1,y,z-1,17)
20 if(x.gt.0.and.z.lt.zm+1)     f(x,y,z,opp(18)) = f(x-1,y,z+1,18)
21 endif
22 end do
```

The newly derived moment-based boundary conditions in 3D are split into three subroutines separating calls to face, edge and corner calculations. Listing 4.10 shows an example of the general face boundary calculation subroutine for velocity and pressure type boundaries. The same expressions can be used for the

other faces by exploiting the symmetry property of the lattice and transforming the velocities and distribution functions accordingly (lines 2 and 19). The `mappingback` call on line 19 only has one argument `f` because velocities will be recalculated using the obtained distribution function values.

LISTINGS 4.10: Moment-based face boundary conditions

```
1  do loop through x,y,z of the boundary face
2  call mapping(f,u,v,w)
3  if(face_pressure_BC) u =
       1-(f(0)+f(3)+f(4)+f(5)+f(6)+f(11)+f(12)+f(13)+f(14)
       +2*(f(2)+f(8)+f(9)+f(16)+f(17)))/r
4  if(face_velocity_BC) r =
       (f(0)+f(3)+f(4)+f(5)+f(6)+f(11)+f(12)+f(13)+f(14)
       +2*(f(2)+f(8)+f(9)+f(16)+f(17)))/(1-u)
5  f(1) = r *(1/3-v *v-w*w)-f(0)-f(2)+f(11)+f(12)+f(13)+f(14)
6  f(7) = r /2*(1/3+v*(v+1))-f(3)-f(9)-f(11)-f(14)
7  f(10) = r /2*(1/3+v*(v-1))-f(4)-f(8)-f(12)-f(13)
8  f(15) = r /2*(1/3+w*(w+1))-f(5)-f(11)-f(13)-f(17)
9  f(18) = r /2*(1/3+w*(w-1))-f(6)-f(12)-f(14)-f(16)
10 call mappingback(f)
11 end do
```

One can work out the mapping for the variable components using the ZXZ Euler rotation. Choosing the west face, south-west edge and low-south-west corner as the default boundaries with Euler angles $(0, 0, 0)$ or simply no rotation, the correct sets of variable components can be derived. Figure 4.2 shows the default rotation or the unknowns at the west face boundary, whereas Figure 4.3 shows the unknowns at the east face boundary in the same positions by rotating the lattice

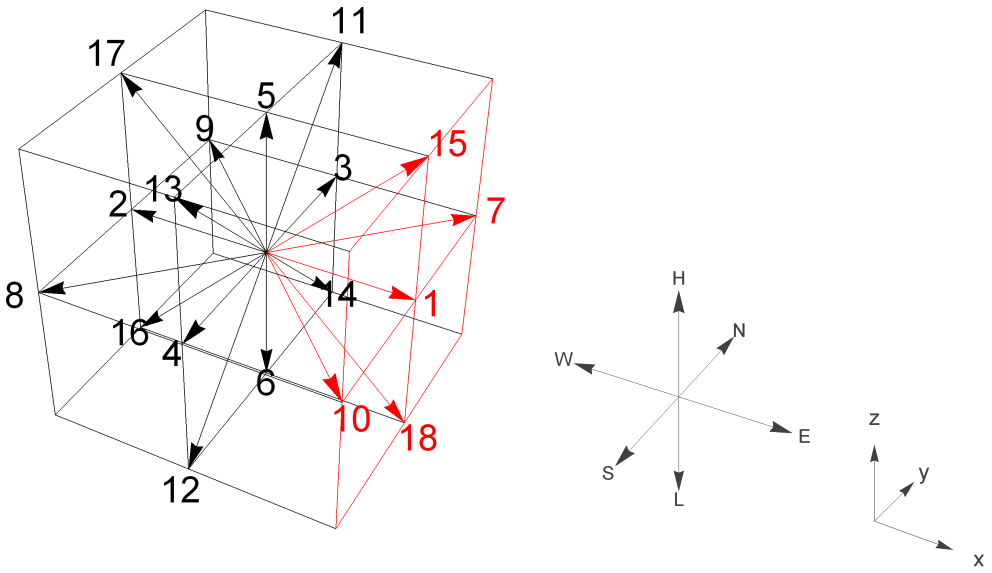


Figure 4.2: Unknowns (shown in red) at the west face boundary, no rotation $(0, 0, 0)$.

by π radians around Z axis. Other sets of Euler angles for the remaining faces, edges and corners are shown in Table 4.2 using the symbols W, E, S, N, L and H to abbreviate the directions west, east, south, north, low and high, respectively.

From Figure 4.3 we see that the unknowns 2, 8, 9, 16, 17 correspond to 1, 7, 10, 15, 18 of the default lattice configuration. These correspondences or mappings can be written down for all the faces, edges and corners. The mappings of the velocities are shown in Table 4.3. Similarly the distribution function mappings can be derived, see Table 4.4. For example, if the unknown distributions functions at the north face boundary need to be calculated, then all the incoming distribution functions `f(4), f(8), f(10), f(12), f(13)` would be mapped onto the default set of the unknowns `f(1), f(7), f(10), f(15), f(18)` using the mappings from Table 4.4 and the rest of the populations `f(i)` would follow the same path. Velocities would go through the same procedure where the x-velocity component `u`

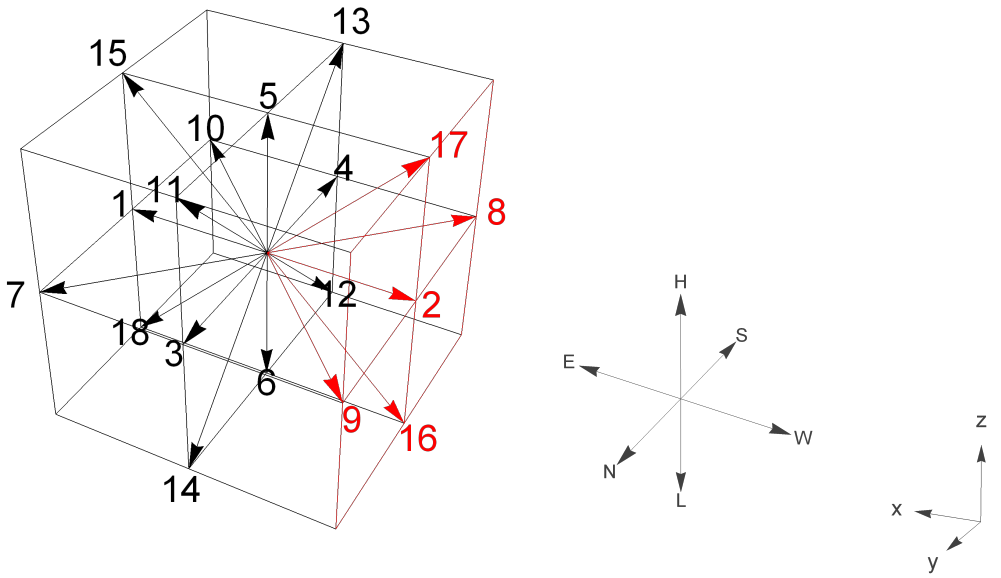


Figure 4.3: Unknowns (shown in red) at the east face boundary, Euler ZXZ rotation $(\pi, 0, 0)$.

Table 4.2: ZXZ Euler angles for boundaries.

Faces

W	0	0	0
E	π	0	0
S	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
N	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$
L	$\frac{\pi}{2}$	$\frac{\pi}{2}$	0
H	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	0

Edges

SW	0	0	0
SE	$-\frac{\pi}{2}$	0	0
NW	$\frac{\pi}{2}$	0	0
NE	π	0	0
LW	$\frac{\pi}{2}$	$\frac{\pi}{2}$	0
LE	π	$\frac{\pi}{2}$	0
HW	0	$\frac{\pi}{2}$	0
HE	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	0
LS	π	$\frac{\pi}{2}$	$\frac{\pi}{2}$
LN	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
HS	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
HN	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$

Corners

LSW	0	0	0
LSE	$-\frac{\pi}{2}$	0	0
LNW	$\frac{\pi}{2}$	0	0
LNE	π	0	0
HSW	$\frac{\pi}{2}$	π	0
HSE	π	π	0
HNW	0	π	0
HNE	$-\frac{\pi}{2}$	π	0

Table 4.3: Mappings for velocities $u = 1, v = 2, w = 3$.

Faces

W*	1	2	3
E	-1	-2	3
S	2	3	1
N	-2	-3	1
L	3	1	2
H	-3	-1	2

Edges

SW*	1	2	3
SE	2	-1	3
NW	-2	1	3
NE	-1	-2	3
LW	3	1	2
LE	-1	3	2
HW	1	-3	2
HE	-3	-1	2
LS	2	3	1
LN	3	-2	1
HS	-3	2	1
HN	-2	-3	1

Corners

LSW*	1	2	3
LSE	2	-1	3
LNW	-2	1	3
LNE	-1	-2	3
HSW	2	1	-3
HSE	-1	2	-3
HNW	1	-2	-3
HNE	-2	-1	-3

* Default rotation.

gets transformed into `w`, `v` into `-u` and `w` into `-v`. This substitution is meant to minimise the redundancy of the code by exploiting the rotational symmetry of the cubic lattice.

The subroutines for the edge and corner boundary conditions are similar to the face one shown in Listing 4.10. Because different moments are being selected for velocity and pressure-velocity type boundaries, see Section 3.6.5, the incoming distribution functions can have different expressions, and therefore need to be separated. For example, distribution functions `f(1), f(3), f(7)` have different expressions for edge velocity (lines 5-7) and pressure-velocity (lines 9-11) type boundaries in Listing 4.11. For the corner boundaries, the list of unique expressions is longer – `f(1), f(3), f(7), f(10), f(11), f(13)` on lines 5-10 and 12-17 for velocity and pressure-velocity type boundaries respectively, see Listing 4.12.

Table 4.4: Mappings for distribution functions f_i.

Faces

W*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
E	2	1	4	3	5	6	8	7	10	9	13	14	11	12	17	18	15	16
S	3	4	5	6	1	2	11	12	13	14	15	16	18	17	7	8	10	9
N	4	3	6	5	1	2	12	11	14	13	18	17	15	16	10	9	7	8
L	5	6	1	2	3	4	15	16	18	17	7	8	9	10	11	12	14	13
H	6	5	2	1	3	4	16	15	17	18	9	10	7	8	14	13	11	12

Edges

SW*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SE	3	4	2	1	5	6	9	10	8	7	17	18	15	16	11	12	13	14
NW	4	3	1	2	5	6	10	9	7	8	15	16	17	18	13	14	11	12
NE	2	1	4	3	5	6	8	7	10	9	13	14	11	12	17	18	15	16
LW	5	6	1	2	3	4	15	16	18	17	7	8	9	10	11	12	14	13
LE	2	1	5	6	3	4	17	18	15	16	11	12	14	13	9	10	7	8
HW	1	2	6	5	3	4	18	17	16	15	14	13	11	12	7	8	9	10
HE	6	5	2	1	3	4	16	15	17	18	9	10	7	8	14	13	11	12
LS	3	4	5	6	1	2	11	12	13	14	15	16	18	17	7	8	10	9
LN	5	6	4	3	1	2	13	14	12	11	10	9	7	8	15	16	18	17
HS	6	5	3	4	1	2	14	13	11	12	7	8	10	9	18	17	15	16
HN	4	3	6	5	1	2	12	11	14	13	18	17	15	16	10	9	7	8

Corners

LSW*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
LSE	3	4	2	1	5	6	9	10	8	7	17	18	15	16	11	12	13	14
LNW	4	3	1	2	5	6	10	9	7	8	15	16	17	18	13	14	11	12
LNE	2	1	4	3	5	6	8	7	10	9	13	14	11	12	17	18	15	16
HSW	3	4	1	2	6	5	7	8	10	9	18	17	16	15	14	13	12	11
HSE	2	1	3	4	6	5	9	10	7	8	14	13	12	11	16	15	18	17
HNW	1	2	4	3	6	5	10	9	8	7	12	11	14	13	18	17	16	15
HNE	4	3	2	1	6	5	8	7	9	10	16	15	18	17	12	11	14	13

* Default rotation.

LISTINGS 4.11: Moment-based edge boundary conditions

```
do loop through x,y,z of the boundary edge
call mapping(f,u,v,w)
if(edge_velocity_BC) then
r = (f(0)+f(5)+f(6)+2*(f(2)+f(4)+f(12)+f(13)+f(16)+f(17)+2*f(8)))
   /(1-(u+v-u*v))
f(1) = r*(-1/3+u*2/3+v*(1-u-v))+f(2)+2*(f(4)+2*f(8)+f(12)+f(13))
f(3) = r*(-1/3+v*2/3+u*(1-u-v))+f(4)+2*(f(2)+2*f(8)+f(16)+f(17))
f(7) =
   r/2*(2/3-(u+v)*(1-u-v))-f(2)-f(4)-3*f(8)-f(12)-f(13)-f(16)-f(17)
elseif(edge_pressure-velocity_BC) then
f(1) = r*(2/3-u/3-v*v)-f(0)-f(2)-f(5)-f(6)-2*(f(16)+f(17))
f(3) = r*(2/3-v/3-u*u)-f(0)-f(4)-f(5)-f(6)-2*(f(12)+f(13))
f(7) = r/2*(-4/3+u*(u+1)+v*(v+1))
   +f(0)+f(2)+f(4)+f(5)+f(6)+f(8)+f(12)+f(13)+f(16)+f(17)
endif
f(11) = f(13)+r/2*v*(w+1/3)
f(14) = f(12)-r/2*v*(w-1/3)
f(15) = f(17)+r/2*u*(w+1/3)
f(18) = f(16)-r/2*u*(w-1/3)
f(9) = r/2*(1/3+u*(u-1))-f(2)-f(8)-f(16)-f(17)
f(10) = r/2*(1/3+v*(v-1))-f(4)-f(8)-f(12)-f(13)
call mappingback(f)
end do
```

LISTINGS 4.12: Moment-based corner boundary conditions

```
do loop through x,y,z of the boundary corner
call mapping(f,u,v,w)
if(corner_velocity_BC) then
r=(f(0)+2*(f(2)+f(4)+f(6))+4*(f(8)+f(12)+f(16)))/(1-u*(1-v-w)-v*(1-w)-w)
f(1) = f(2)+r*u/3
f(3) = f(4)+r*v/3
f(7) = f(8)+r*(u+v)/6
f(10)=r*(1/3+v*(v-1)+u*(u-1/3)-w*(w-1/3))/4-(f(2)+f(4)-f(6))/2-f(8)
f(11) = f(12)+r*(v+w)/6
f(13)=r*(1/3+v*(v-1)+w*(w-1/3)-u*(u-1/3))/4-(f(2)+f(4)-f(6))/2-f(12)
elseif(corner_pressure-velocity_BC) then
f(1) = r*((2-2*w-u)/3+u*w-v*v)-f(0)-f(2)-2*f(6)-4*f(16)
f(3) = r*((2-2*u-v)/3+u*v-w*w)-f(0)-f(4)-2*f(2)-4*f(8)
f(7) = r/2*((2*w+v-2)/3+u-u*w+v*v)+f(0)/2+f(2)+f(6)+f(8)+2*f(16)
f(10) = r/2*((2*w-2)/3+u-u*v-u*w+v*v)+f(0)/2+f(2)+f(6)+f(8)+2*f(16)
f(11) = r/2*((2*u+w-2)/3+v-u*v+w*w)+f(0)/2+f(2)+f(4)+f(8)+2*f(12)
f(13) =
   r/2*(1-2/3*w-v-u*(1-v-w))-f(0)/2-f(2)-f(4)-f(6)-f(12)-2*f(8)-2*f(16)
endif
f(5) = f(6)+r*w/3
f(9)=r*(1/3+u*(u-1)+v*(v-1/3)-w*(w-1/3))/4-(f(2)+f(4)-f(6))/2-f(8)
f(14)=r*(1/3+w*(w-1)+v*(v-1/3)-u*(u-1/3))/4-(f(2)+f(4)-f(6))/2-f(12)
f(15) = f(16)+r*(u+w)/6
f(17)=r*(1/3+u*(u-1)+w*(w-1/3)-v*(v-1/3))/4-(f(2)+f(4)-f(6))/2-f(16)
f(18)=r*(1/3+w*(w-1)+u*(u-1/3)-v*(v-1/3))/4-(f(2)+f(4)-f(6))/2-f(16)
call mappingback(f)
end do
```

4.2.5 Macroscopic variables

The fluid variables get calculated after the boundary conditions have been applied and the unknown distribution functions have been obtained. The calculation is as simple as the formulas listed in (3.48). Because the interest is in the low-order hydrodynamic moments, only the fluid density and velocity are calculated here. The expressions for the velocities generally include a half contribution of the applied force, lines 3-5 in Listing 4.13.

LISTINGS 4.13: Macroscopic variables

```
1 do loop through x,y,z
2 r(x,y,z) = sum(f(x,y,z,:))
3 u(x,y,z) = sum(cx(:)*f(x,y,z,:))/r(x,y,z)+fx(x,y,z)/2
4 v(x,y,z) = sum(cy(:)*f(x,y,z,:))/r(x,y,z)+fy(x,y,z)/2
5 w(x,y,z) = sum(cz(:)*f(x,y,z,:))/r(x,y,z)+fz(x,y,z)/2
6 end do
```

4.2.6 Output

The solution can be periodically written to a file by calling the write-to-file subroutine shown in Listing 4.14.

LISTINGS 4.14: Writing to file

```
1 open(datafile='output'//timestep//'.dat')
2 do loop through x,y,z
3 write(datafile) x,y,z,r,u,v,w,fx,fy,fz,obstacle
4 end do
5 close(datafile)
```

All the necessary variables get passed in and one by one they get recorded into a file for an intermediate analysis or just a later post-processing. For convenience, the velocity and force magnitudes can be included on line 3 by adding two extra variables, namely, `sqrt(u*u+v*v+w*w)` for the fluid velocity and `sqrt(fx*fx+fy*fy+fz*fz)` for the force magnitude. Note that the variables in the written file will be in lattice units unless they get converted to SI units before outputting.

4.3 Parallelisation

With the ever increasing sizes of the computational domains and the availability of large computing resources the parallelisation is inevitable and more so because a personal computer is limited by the completion time of the simulation and the available RAM. There are different ways of parallelising the code, whether it is CPU-only or CPU-GPU hybrid, single workstation or a cluster, etc.

The LBM code given in parts in the previous section can be made parallel by decomposing the calculation domain into cuboids of equal size and using the MPI-based libraries developed by Kao in the framework of TESA for transferring data between the interprocessor boundaries. MPI is a message-passing library interface specification that allows the data communication between multiple concurrent processes. The message-passing standard is portable and easy to use because it is packed as a set of functions and subroutines, and is language independent [159].

Running the LBM algorithm on multiple processors requires a constant information exchange between the processing units. If N is the length of the calculation domain, then the size of the communicated data arrays is proportional to

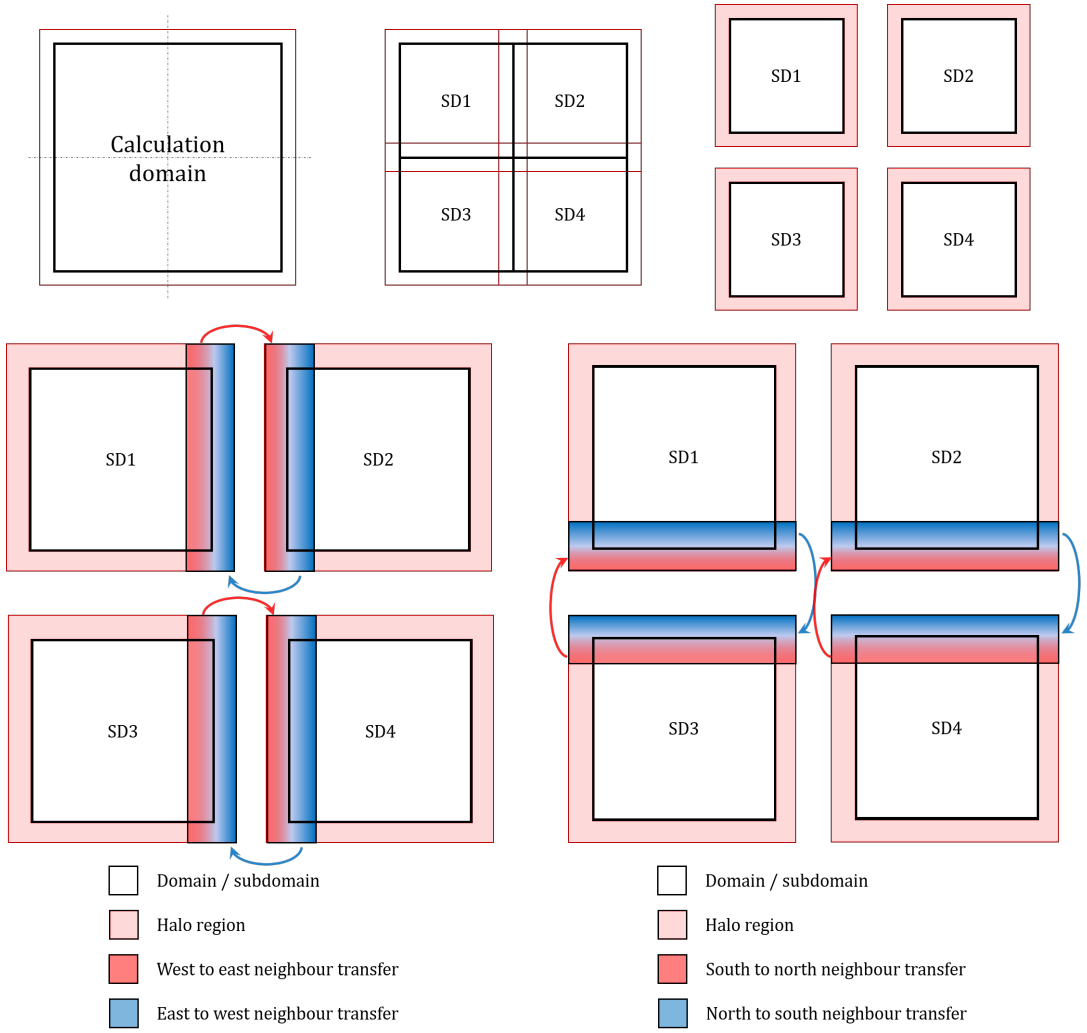


Figure 4.4: Domain decomposition and MPI data transfers to halo regions.

N^2 because only the surface nodes need to be passed on. Currently, the whole variable array gets passed to the MPI boundary updating subroutine where the surface arrays get extracted and communicated to the neighbouring subdomains. The outer layer of each cuboid is called the *halo region*. It is the overlapping region where the data is sent to and received at. Figure 4.4 shows a 2D schematic drawing of the data transfer between the subdomains.

In the LBM code the MPI boundary update for velocity and pressure is called

every time step before the LB boundary condition calculation, and the MPI update for the distribution functions is called after the LB boundary conditions are handled. Passing the whole variable array to the MPI subroutine might become an issue when the code gets parallelised on GPUs using CUDA. It would mean that the whole variable array needs to be copied over from the device to host every time step, and it would seriously affect the calculation time. A possible solution is discussed in Section 6.2.

The LB algorithm can also be parallelised on GPUs in several different ways. Depending on the amount of the programming effort that the user is willing or is able to put in the implementation of the GPU parallelisation, one can choose either to use a directive-based model to accelerate the code or modify the code using OpenCL [160; 161], CUDA programming language extension. Although OpenCL is a heterogeneous computing environment that utilises CPUs and GPUs, it has been demonstrated that the CUDA implementation on NVIDIA devices slightly but outperforms the OpenCL approach [162–164]. Originally OpenCL is written in C, but it can be accessed from Fortran applications through the CLFORTRAN interface [165]. Using parallel compiler directives, such as OpenACC [166] or OpenMP [167] requires the least amount of programming, offers relatively reasonable speed-up and allows to access either CPU or GPU resources. However, to get the most out of the Fortran code the use of CUDA Fortran is necessary [168]. Moreover, considering the available resources and the previous knowledge within the research group, the GPU parallelisation of the code is executed using the CUDA platform.

GPUs are different from CPUs in that they have several hundred or even thousands of cores that can simultaneously execute tasks, see Figure 4.5. A GPU

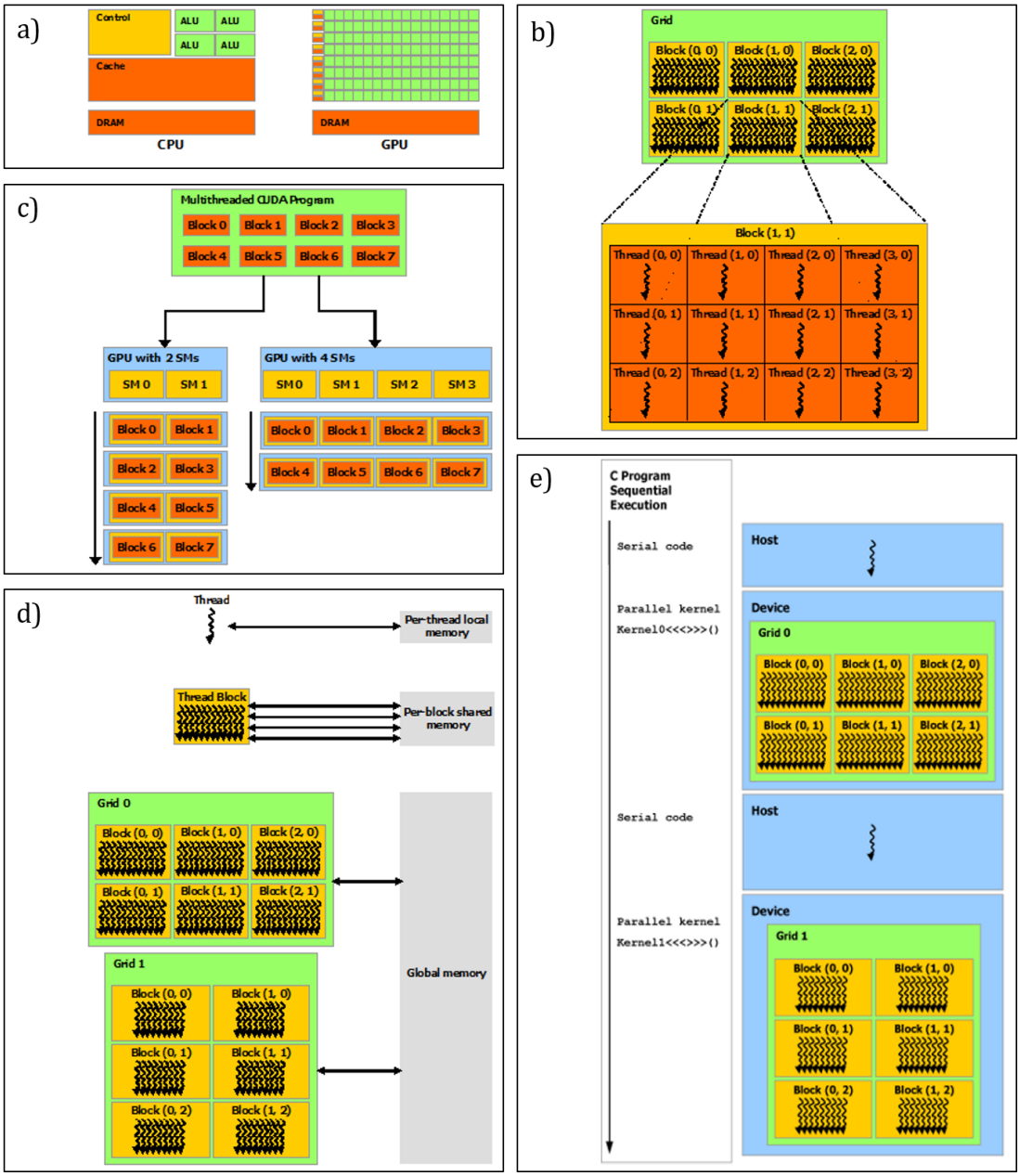


Figure 4.5: GPU architecture on different levels: a) comparison between the CPU and GPU build, b) a grid of thread blocks, c) CUDA program invoking kernel grids on GPUs, d) memory hierarchy, e) an example of heterogeneous programming [169].

consists of an array of streaming multiprocessors (SMs), and each SM hosts a grid of thread blocks. A CUDA application invokes a kernel grid which allocates available blocks for concurrent execution of tasks. There are different types of memory spaces available on a CUDA device limited by the size, scope, lifetime and access speed that need to be considered when optimising the code for performance. The algorithm resides on the host where all the preparatory and I/O subroutines are executed. Calls to the device are also made from the host. The variable arrays are transferred between host and device using host-to-device and device-to-host memory copy operations. In Fortran, these operations can be as simple as `a_h = a_d` and `a_d = a_h`. To speed up the copying process, all the frequently used variable arrays are allocated in the pinned memory on the host.

LISTINGS 4.15: CUDA Fortran code example

```
1  x = (blockIdx%x-1) * blockDim%x + threadIdx%x
2  y = (blockIdx%y-1) * blockDim%y + threadIdx%y
3  z = (blockIdx%z-1) * blockDim%z + threadIdx%z
4  if (x.le.xm+1 .and. y.le.ym+1 .and. z.le.zm+1) then
5  do i = 0, 18
6  f(x,y,z,i) = (1-omega)*f(x,y,z,i)+omega
       *wt(i)*r(x,y,z)*(1+3*cu(i)+(1-stokes)*(4.5*cu(i)*cu(i)-1.5*vel))&
7  !HSD 1998 Force (Comment out if not used)
8  +(1-omega/2)*3*(cf(i)-uf)
       *wt(i)*r(x,y,z)*(1+3*cu(i)+(1-stokes)*(4.5*cu(i)*cu(i)-1.5*vel))
9  !Luo 1998 Force (Comment out if not used)
10 +(1-omega/2)*wt(i)*r(x,y,z)*3*(cf(i)-uf+3*cu(i)*cf(i))
11 end do
12 endif
```

Listing 4.15 shows the BGK collision subroutine written using CUDA Fortran syntax. The spatial `do` loops as seen in Listing 4.4 are replaced by the absolute thread addresses on the device, lines 1-3. The `if` condition on line 4 is used to stay within the bounds of memory when the domain size is not a multiple of thread block size. The rest of the subroutine remains the same.

In the main loop, the device subroutines are called using a special triple-chevron syntax `<<<grid,tblock>>>` where the grid dimensions and thread block dimensions are specified, lines 2-6 in Listing 4.16. Because the I/O operations are performed on the host, the calculated variable arrays need to be copied over from the device to the host before calling the write-to-file subroutine, lines 7-8.

LISTINGS 4.16: CUDA Fortran main loop

```
1 do loop through t
2 call
    collision<<<grid,tblock>>>(f_d,r_d,u_d,v_d,w_d,fx_d,fy_d,fz_d,omega)
3 call streaming<<<grid,tblock>>>(f_d,fp_d)
4 call bc_moment<<<grid,tblock>>>(f_d,r_d,u_d,v_d,w_d)
5 call bc_bbrule<<<grid,tblock>>>(f_d,fp_d,obstacle_d)
6 call macrovars<<<grid,tblock>>>(f_d,r_d,u_d,v_d,w_d,fx_d,fy_d,fz_d)
7 if (t=t_out) r=r_d; u=u_d; v=v_d; w=w_d
8 if (t=t_out) call writetofile(datafile,x,y,z,r,u,v,w,fx,fy,fz,obstacle)
9 end do
```

Both `grid` and `tblock` are three-dimensional parameters. Depending on the GPU architecture and compute capability these 3D parameters have certain limits. This can be checked by running a short device query program that comes with the CUDA Fortran compiler and detects the available CUDA devices and dis-

plays some basic information including the compute capability, available memory and other details, see Listing 4.17. Lines 10-11 show the maximum dimensions of the grid and thread block. The maximum number of threads per block is fixed at 1024 (line 12), which restricts the `tblock` dimensions to be `tblock = dim3(tbx,tby,tbz)` where `tbx*tby*tbz` $\leq$ 1024. An example of the dimension definitions is given in Listing 4.18. The `ceiling` function on lines 3-5 is used in case the thread block dimensions are not factors of the calculation domain size.

LISTINGS 4.17: Example of a device query program console output

```
1  One CUDA device found
2  Device Number: 0
3   GetDeviceProperties for device 0: Passed
4   Device Name: GeForce GTX 1060 6GB
5   Compute Capability: 6.1
6   Number of Multiprocessors: 10
7   Max Threads per Multiprocessor: 2048
8   Global Memory (GB):   6.000
9   Execution Configuration Limits
10    Max Grid Dims: 2147483647 x 65535 x 65535
11    Max Block Dims: 1024 x 1024 x 64
12    Max Threads per Block: 1024
```

LISTINGS 4.18: CUDA Fortran kernel grid dimensions

```
1  integer, parameter :: gridx = 128, gridy = 128, gridz = 128
2  type(dim3), parameter :: tblock = dim3(16,16,4)
3  type(dim3), parameter :: grid = dim3(ceiling(real(gridx)/tblock%x),&
4                                       ceiling(real(gridy)/tblock%y),
5                                       ceiling(real(gridz)/tblock%z))
```

4.4 Coupling between LB and other solvers

4.4.1 CA-LB coupling

The LB code described in the previous sections can be coupled to other solvers to model various multi-physics problems where flow is present. One such example is dendritic solidification with convection. It involves solving for heat, mass and momentum transport and even electromagnetics if the electromagnetic damping or the thermoelectric effect is being considered. The in-house software TESA is capable of solving the microstructure solidification. By coupling the LB code to TESA, the fluid flow can be resolved during the solidification process. The coupling of both algorithms is rather simple. The LB flow solver is interested in the previous velocities, solid fraction and the external force. The LB subroutine gets called with these arguments. So, for example, the effect of the thermal field calculated in TESA gets passed to the LB solver by the means of a thermal buoyancy force. Similarly, the effect of the concentration variations gets passed on in the form of a solutal buoyancy force. Inside the LB subroutine the values get scaled to lattice units, then they are used to update the velocity field via collisions and streaming and finally the calculated new velocities get converted back before they are returned to the external code. The external code is a black box as far as the LB code is concerned. Only the velocity, force and solid fraction arrays get passed to the LB flow solver which then returns the updated velocity values, see Figure 4.6. The flow solver gets called either every time step together with the solidification solver or every n-th step if the solid phase change is negligible. The parallelisation side and coupling between the CA and LB methods to simulate large-scale dendritic solidification has been described in [170] by Kao *et al.* In

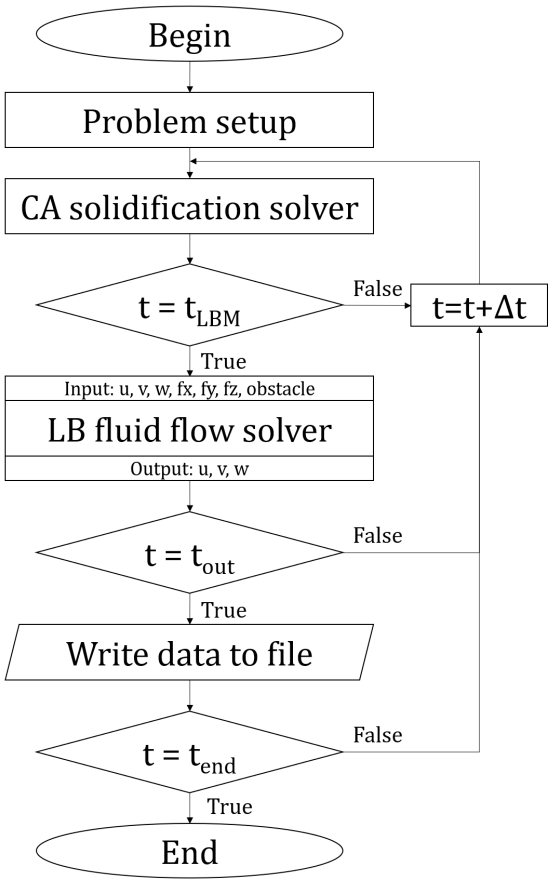


Figure 4.6: Flow diagram of the coupled CA-LB algorithm.

addition to the variables mentioned above, other arguments, such as the boundary types and values, processor number and topology, problem specific flags, as well as the current time step and time interval are also passed to the LB flow solver.

4.4.2 LB-enthalpy method coupling

For the simple 2D benchmark cases involving crystal growth, the LBM has been coupled to an enthalpy-based method analogous to the CA-LB coupling discussed in the previous section. In the enthalpy method described by Voller [171], the energy conservation equation is being solved, specifically the enthalpy, which

consists of the sum of sensible and latent heats given by

$$H = c_p T + f_{\text{liq}} \Delta H. \tag{4.6}$$

Transport of heat is defined as

$$\frac{\partial H}{\partial t} = \alpha \nabla^2 (c_p T) - \mathbf{u}(\nabla H), \tag{4.7}$$

where α is the thermal diffusivity and is assumed to be the same for both liquid and solid phases. The enthalpy H is expressed as a function of the heat capacity c_p, temperature T and the latent ΔH in the liquid phase, which is described by f_{liq}. For a pure material depending on the fusion temperature T_f, curvature κ and the surface tension anisotropy $\gamma(\theta)$ and neglecting kinetic effects, the solid-liquid interface temperature T^i can be written as

$$T^i = T_f - \frac{\gamma(\theta) T_f}{\Delta H} \kappa. \tag{4.8}$$

The local interface curvature can be captured from the liquid fraction gradients as

$$\kappa = \nabla \cdot \left(\frac{\nabla f}{|\nabla f|} \right) = \left(f_y^2 f_{xx} - 2 f_x f_y f_{xy} + f_x^2 f_{yy} \right) \cdot \left(f_x^2 + f_y^2 \right)^{-3/2}, \tag{4.9}$$

where f_x and f_{xx} represent, respectively, the first and the second derivative of f with respect to the coordinate x. The interface orientation θ, which is the angle between the interface normal and the x-axis, can be calculated by

$$\theta = \operatorname{atan} \left(\frac{f_x}{f_y} \right). \tag{4.10}$$

By introducing variable scaling, dimensionless form denoted by superscript (*) can be obtained as

$$\begin{gathered} T^* = \frac{T - T_f}{T_0}, \qquad T_0 = \frac{\Delta H}{c_p}, \qquad \alpha^* = \frac{\alpha}{\alpha_0}, \qquad \rho^* = \frac{\rho}{\rho_0}, \\ t^* = \frac{t}{t_0}, \qquad t_0 = \frac{\alpha}{\Delta H}, \qquad x^* = \frac{x}{x_0}, \qquad x_0 = \sqrt{\alpha t_0}, \qquad \kappa^* = \frac{\kappa}{\kappa_0}. \end{gathered} \tag{4.11}$$

The fusion temperature T_f for aluminium is approximately 933 K. For convenience α^* and ρ^* are both set equal to 1. With the superscripts dropped, this gives the following dimensionless equations:

$$\frac{\partial T}{\partial t} = \nabla^2(T) - \mathbf{u}(\nabla T), \tag{4.12}$$

$$H = T + f_{\text{liq}}, \tag{4.13}$$

$$T^i = -\kappa\left(1 - 15\epsilon\cos(4\theta)\right). \tag{4.14}$$

In the latter one (4.14), the interfacial temperature is expressed as a function of the dimensionless curvature and Gibbs–Thomson coefficient with ϵ being the anisotropy parameter.

4.5 Performance analysis

To quantify the efficiency of the LBM and the improvement that the parallelisation offers in terms of the simulation time, a performance analysis has to be carried out. First of all, the LBM is analysed by comparing the timings of each subroutine as a percentage of the total LBM runtime. The LBM algorithm can also be adapted for Stokes flows by activating the flag `stokes` which neglects the

nonlinear terms $O(u^2)$ in the code for low velocity flows. The benefit of it is the reduction in the calculation time. It makes the collision scheme run 20 % faster which makes the LBM algorithm perform 10 % faster. Figure 4.7 shows the timings of the subroutines as a percentage of the total time for the BGK and TRT collision schemes in normal and Stokes regime. The graphs are scaled proportionally to the total runtime of the respective scheme. In all cases, the main bottleneck is the collision scheme accounting for a half of the computational time on average. The present implementation of TRT is computationally more expensive and hence more time consuming than that of BGK. Contrarily to the reports in literature [115], the TRT scheme is observed to be 43 % more time consuming than BGK. In the present setup, that means the increase in simulation runtime by 17 %, which is not a bad trade-off for an improved accuracy and stability. The runtime of the moment method subroutine scales as L^2, where L is the length of the domain, because it only acts on the domain boundaries. As the moment method only handles Dirichlet type boundaries, the runtime of the subroutine varies for different problem setups. The calculation time percentage of the bounce-back subroutine runtime depends on the amount of solid nodes in the domain. Collision, streaming and macroscopic variable calculation subroutines are all bulk operations performed on every node, hence they scale as L^3.

Figure 4.8 shows the individual timings for handling face, edge and corner boundaries inside the moment method subroutine. Theoretically the ratio between the face, edge and corner boundary calculations time wise should be around $L^2 : 2L : 2$. For a cube with $L = 32$, the ratio is measured to be approximately $512 : 39 : 2$ which is relatively close to the theoretical ratio $512 : 32 : 1$, suggesting that the numerical implementation of the boundary method is correct.

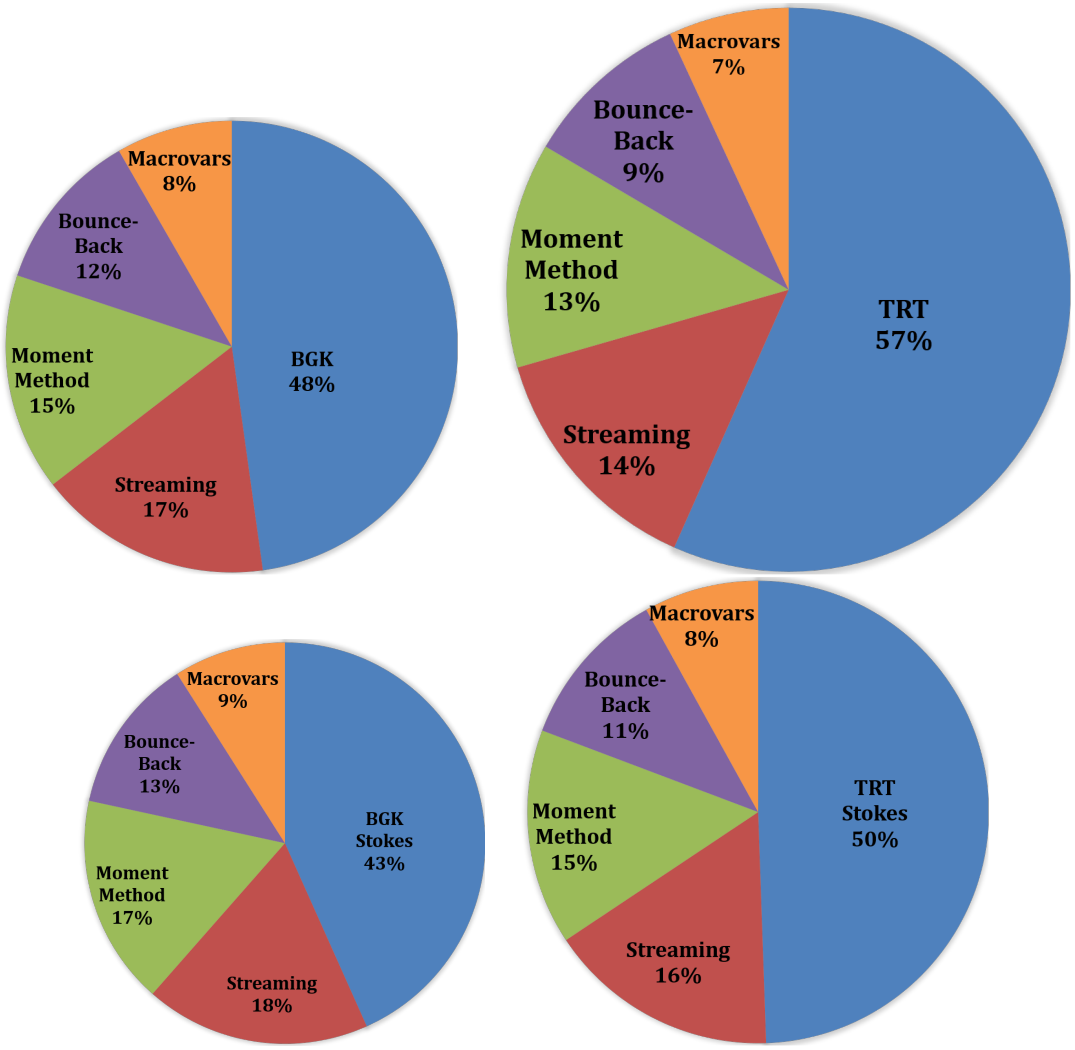


Figure 4.7: Subroutine runtime as a percentage of the total LBM runtime for BGK (left) and TRT (right) in normal (top) and stokes (bottom) regime. The graphs are scaled with respect to the runtime.

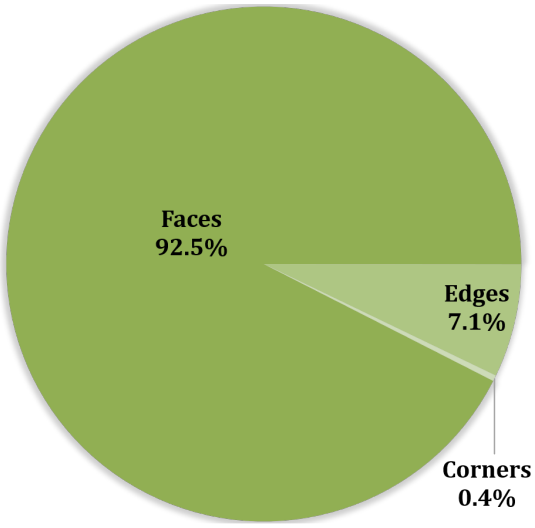


Figure 4.8: Calculation time of each type of boundary as a percentage of the total runtime of the moment-based boundary subroutine.

4.5.1 Strong and weak scaling

The strong and weak scaling of the developed LB code has been tested on a small cluster by running series of tests of various sizes at different processor setups. The idea of the strong scaling is to fix the problem size and employ more and more computer resources. Basically, distributing the work load across many processors. Ideally one would expect a perfect scaling where the speed-up gained by the program is equal to the assigned number of processing units. In reality an ideal scenario like this is very unlikely due to inter-node communications but mainly because of Amdahl's law, which predicts the potential parallel speed-up of the code based on the amount of the serial parts. It is, however, possible to achieve and maintain very high efficiencies that are considered desirable and still close to ideal.

The weak scaling tests how the performance changes when the problem size and the work load increases along with the number of processors. If the code

was *embarrassingly parallel*, that is, if the processes were independent of each other, the calculation time should be the same no matter how many processors are running the program. However, because the processes are not independent, Gustafson's law predicts the increase in timing the bigger the serial fraction of the code is. Furthermore, the information needs to be transferred between the processors, which accounts for the communication time and adds to the total runtime of the problem. When done optimally the communication process should only take a set amount of time while the processors exchange information.

The problem chosen to test the scaling is a 3D square-duct flow with an applied external force. The cross-section area of the square duct is varied between 16^2, 32^2, 64^2, 128^2 and 256^2 while the length of the duct is linked to the processor configuration either giving the same problem size for the strong scaling or changing proportionally with the number of processors assigned in the case of the weak scaling. The maximum length for the strong and weak scalings are chosen with the time and memory constraints in mind so 18720 and 30720 are used in the scaling calculations.

The results of the strong scaling are shown in Figure 4.9 where the speed-ups are scaled by node. Although the nodes used in the cluster comprise 16, 20 and 24 cores, the speed-up by node is scaled using particularly the 16-core node. The efficiency results of the test runs are gathered in Table 4.5. The efficiency is just how close the obtained speed-up is to ideal. In this case, the high efficiency can be attributed to the chosen configuration of the test problem. The domain is only decomposed in one direction which is favourable to the MPI communications.

The effect of the feature called *turbo boost*, which increases the performance of the processor when it is not fully utilised, is visible at low number of cores

Table 4.5: Efficiency of the test runs of different grid sizes scaled by node.

Cores	Ideal	16	32	64	128
1	1	1.45	1.53	-	-
2	1	1.41	1.47	-	-
4	1	1.21	1.28	1.31	-
8	1	0.90	0.93	0.97	-
16	1	1	1	1	1
32	1	1.00	1.01	0.98	0.95
52	1	0.98	1.02	0.96	0.91
72	1	0.96	1.01	0.99	0.91
96	1	0.96	1.00	1.02	0.89
120	1	0.98	0.98	0.98	0.88

in Table 4.5. It is also apparent in the weak scaling, where it is responsible for the relatively faster simulation runtimes when utilising sparsely populated cluster nodes, see Figure 4.10.

The data points have been collected and averaged over several runs to minimise the influence of the outside factors, such as other programs running in the background and using the CPU and memory resources. Due to the time limitation, the larger cases have only been run once, which might explain some deviations from the norm.

4.5.2 Single and double precision

Although using double precision in calculations theoretically gives a more accurate solution, the difference between the results is often indistinguishable. Plus using double precision is more time and memory consuming. For example, running the 3D developed duct flow on a 33^3 grid in double precision for 50000 time steps takes $\sim 45\%$ longer than using single precision, and the output file size is twice as big. Comparing the results of both precisions reveals that there is no

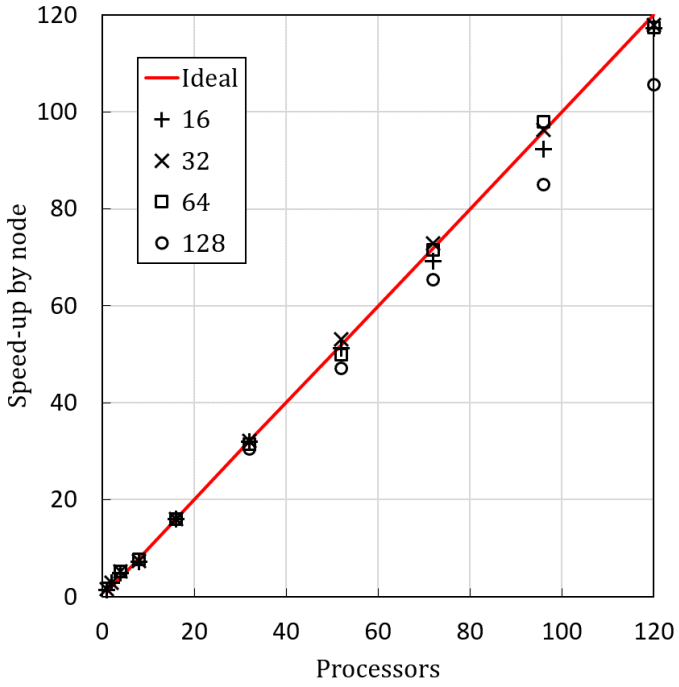


Figure 4.9: Strong scaling by node for different grid sizes.

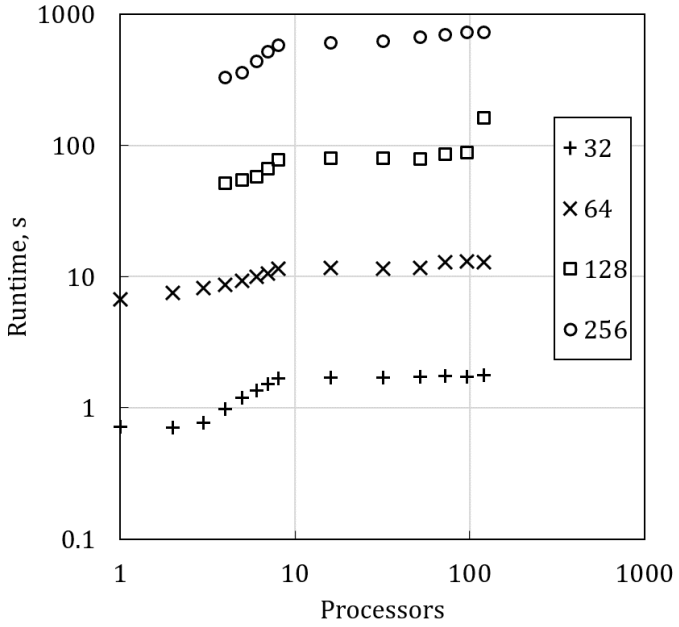


Figure 4.10: Weak scaling for different grid sizes.

significant difference in this case. The relative error norm for single and double precision runs is respectively 8.24E-4 and 8.21E-4. Considering these findings, the single precision is chosen over double precision due to the advantages it offers.

4.5.3 Serial vs. parallel LBM CUDA

In general, the use of CUDA Fortran should considerably speed up the calculations of large arrays. However, the amount of speed-up depends on how suitable for parallelisation the implementation of the subroutine is. For example, a well-structured, purely bulk operation has the potential of high speed-ups. On the other hand, a rather complex-structured code will yield no gain or even be slower then the serial version. This can be seen in Table 4.6 and Figure 4.11. For a benchmark case of a 128^3 grid and 1000 time steps, the parallel CUDA version shows on average a 7.5-time speed-up. While most subroutines show obvious speed-up, the moment-based boundary subroutine experiences hardly any speed-up due to unoptimised CUDA implementation. Because the subroutine acts on the domain boundaries, faces, edges and vertices, it requires a special kernel call that is different from that of the bulk operation kernel calls. However, optimisation of the subroutine would require restructuring it, which is out of scope of this thesis, as the main objective here is to show the potential of the use of GPUs.

The breakdown of the LBM CUDA subroutine runtimes is shown in Figure 4.12. The excessive runtime of the moment-based boundary subroutine overshadows the fast execution of the remaining LBM CUDA algorithm. The time spent in each of the boundary subroutine calls as a percentage of the total call time is 22.4 %, 48.4 % and 29.2 % for the faces, edges and corners, respectively. The distribution is somewhat arbitrary and far off the theoretical ratio of $L^2 : 2L : 2$,

Table 4.6: Timings and speed-ups of CPU and CUDA subroutines obtained on a 128^3 grid after 1000 time steps.

Subroutine	CPU*, s	CUDA**, s	Speed-up
BGK	442	9.97	44
TRT	637	15.0	42
Streaming	72.9	6.86	11
Moment Method	76.8	71.6	1.1
Bounce-Back	17.8	1.53	12
Macrovars	48.3	7.98	5.9
Total***:	755	101	7.5

*CPU: Intel Core i7-3820 3.60GHz
**CUDA: NVIDIA GTX680 4GB
***Using an average collision value

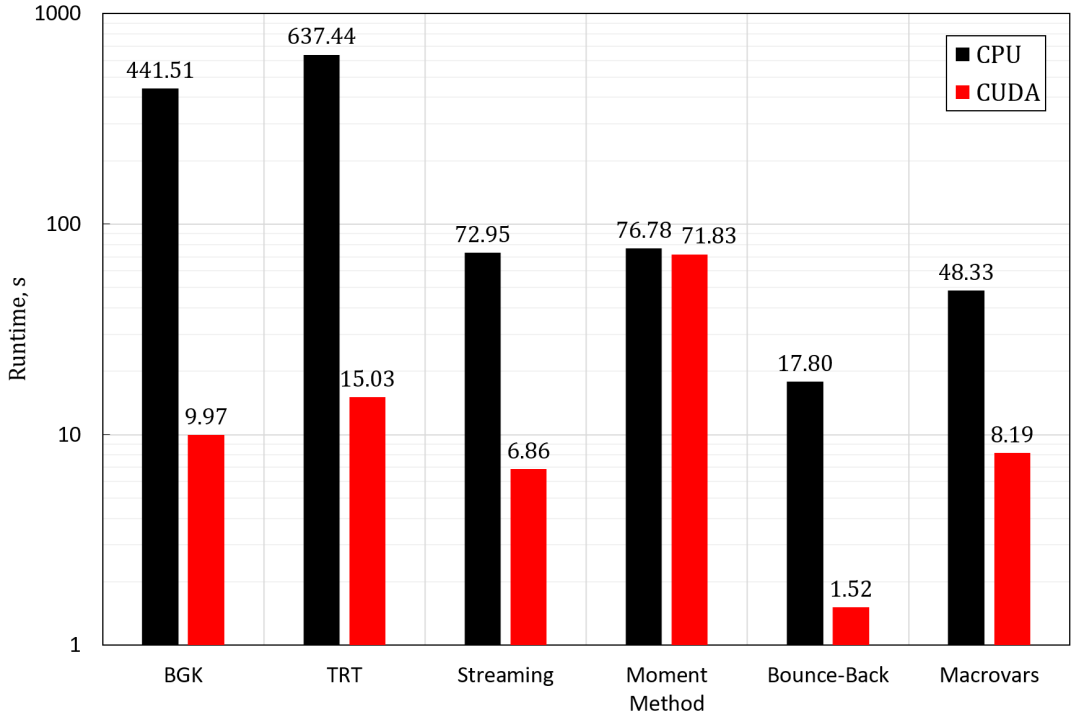


Figure 4.11: CPU and CUDA runtime comparison for the LBM subroutines.

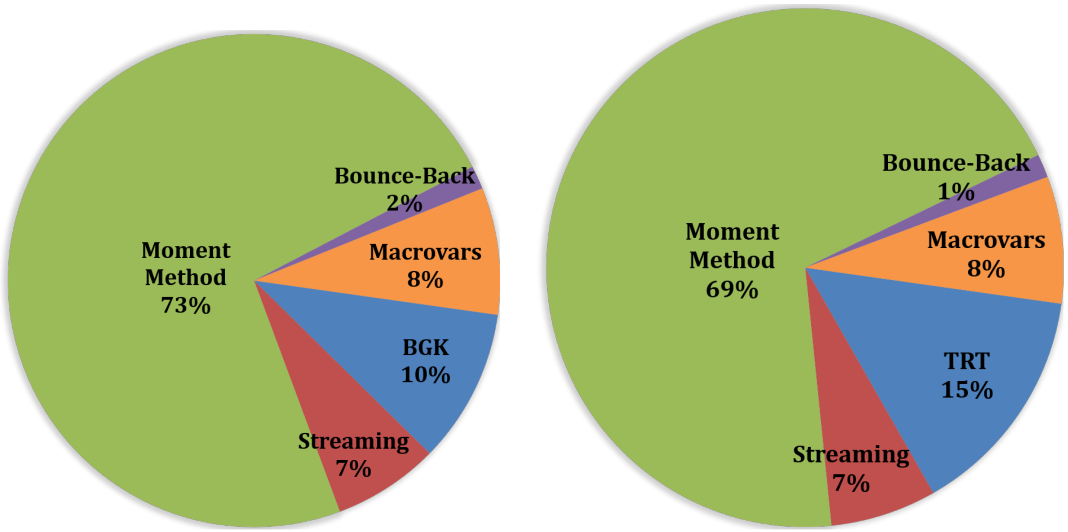


Figure 4.12: CUDA subroutine runtime as a percentage of the total LBM runtime for BGK (left) and TRT (right).

see Figure 4.8, which means that there is a high overhead that has nothing to do with the direct calculation of the boundary conditions themselves.

4.5.4 Lattice Boltzmann vs. discretised Navier-Stokes

One of the LBM merits is its efficiency. It can outperform conventional Navier–Stokes solvers that use FV, FE or FD discretisation schemes when simulating time-dependent hydrodynamics [70; 71]. In this section, the efficiency of the LBM is tested and compared to that of several other solvers. General-purpose commercial softwares like COMSOL [172], ANSYS Fluent [173] and PHOENICS [174], as well as the fluid flow solver from the in-house software TESA are all tested to see how they perform against the LBM. COMSOL is representing an explicit FE flow solver. ANSYS Fluent and PHOENICS are both using FVM, and TESA uses an implicit FDM to solve the flow. It is safe to say that explicit schemes are faster than implicit schemes when talking about time-dependent

problems with fixed time stepping. The question is how much faster? Also, FEM is not always the first choice when it comes to solving fluid flow so one might expect slower performance from COMSOL compared to other solvers.

The transient benchmark test case is a simple quasi-3D lid-driven cavity flow. The size of the square cavity is $L = 0.01$ m, the top wall is moving at a constant speed $u_{\text{lid}} = 0.01$ m/s, the kinematic viscosity is set as $\nu = 1 \cdot 10^{-6}$ m^2/s giving Re = 100. Neumann zero condition is applied to the near and far boundaries. The grid size varies from $8 \times 8 \times 1$ to $2048 \times 2048 \times 1$ going through powers of 2. For the LBM and TESA solvers, the depth is set to 3 nodes where the two outer ones represent the symmetry boundaries and the middle one is the interior. The simulation is run for 1000 time steps of a fixed step size of $\delta t = 3.125 \cdot 10^{-4}$ s.

The results are listed in Table 4.7 and plotted in Figure 4.13. The line of slope 2 shown in Figure 4.13 represents the theoretical trend of doubling the grid length of a 2D domain and as a result the computational time increasing 4 times, which corresponds to the change in the area of the computational domain. So the trend is basically a square function. All the solvers show similar behaviour in the linear region where the initialisation overhead is negligible. The rates range from 2 for the LBM to 2.3 for TESA. The deviation from the theoretical value might be attributed to the other computer processes running in the background when conducting the tests. The timing data at smaller grid sizes does not follow the linear theoretical trend due to the initialisation and memory allocation tasks that add a certain amount of time which is proportionally larger for smaller problems.

It is not surprising to see that COMSOL, which uses the FEM, is the most time-consuming solver followed by ANSYS Fluent. Fluent representing a commercially popular FV solver is more than 15 times faster than COMSOL when run

Table 4.7: Timings for different methods used to solve a lid-driven cavity flow.

1000 outputs / Grid size	Times, s (1000 time steps)				
	LBM	FEM COMSOL	FVM Fluent	FVM PHOENICS	FDM TESA
8	0.5	33	63	3.0	**0.1**
16	0.7	58	63	3.0	**0.4**
32	**0.7**	201	68	4.0	1.3
64	**2.0**	872	90	8.0	5.3
128	**7.2**	3667	210	23	22
256	**33**	15608	838	88	118
512	**137**	80101	5850	764	694
1024	**559**	-	16678*	4140	3517
2048	**2323**	-	-	18300	17069

*Grid size of 715 was used

explicitly. Because the global time stepping cannot be used to compute unsteady incompressible flows, the non-iterative regime of the PISO (*Pressure-Implicit with Splitting of Operators*) algorithm is chosen to mimic an explicit scheme.

PHOENICS is another example of a FV solver dating back to early 1980s when the available computational power and memory were very limited. Not being able to select a fully explicit time stepping, the implicit algorithm's iteration count during one time step is set to 1 and the variable values are forced to relax straight to those of the current time step. The results show that PHOENICS is on average 10 times faster than Fluent, closing in on the LBM, but still trailing by a factor of 4. In contrast to Fluent, PHOENICS and TESA are structured codes, which explains their speed advantage.

The iteration count in TESA flow solver at every time step is set to 1. Although the solution might not have converged at each time step, the FDM solver runs as fast as possible. Similarly to PHOENICS, the TESA flow solver is on average 4 times slower than the LBM, when comparing values at the larger grids.

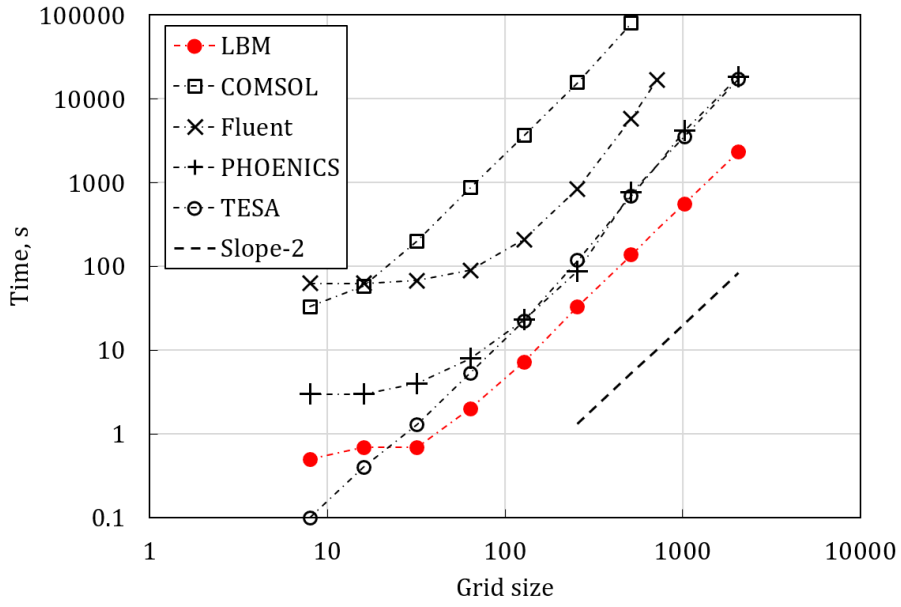


Figure 4.13: Timings for different methods used to solve a lid-driven cavity flow.

Knowing the speed-up of the LBM compared to the FDM flow solver, S_{LBM}, and that $80-90$ % of the total TESA simulation time is normally spent solving flow, $t_{\%}$, the total speed-up of TESA, S, gained by replacing the flow solver can be calculated as

$$S = \left[1 + \frac{t_{\%}}{100\ \%}\left(\frac{1}{S_{\mathrm{LBM}}} - 1\right)\right]^{-1}. \tag{4.15}$$

The possible outcomes are listed in Table 4.8. In the current case, using the values $t_{\%} = 80$ % and $S_{\mathrm{LBM}} = 4$, the total speed-up is calculated to be $S \approx 2.5$. This speed-up value is just the lower estimate due to the reduced iteration count in the implicit FDM flow solver and the assumption that it consumes only 80 % of the total simulation time. However, this is just the lower estimate because it normally takes the implicit solver more than 1 iteration for the solution to converge at each time step, which means that the LBM speed-up can be even higher. Also, the assumption that the flow solver consumes only 80 % and not

Table 4.8: Theoretical speed-ups of the coupled code with respect to the calculation time percentage of the flow part, $10 - 90$ %, and the LBM speed-up, $1 - \infty$.

Speed-up / t%	1	2	3	4	5	6	7	8	9	∞
10	1.0	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
20	1.0	1.1	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
30	1.0	1.2	1.3	1.3	1.3	1.3	1.3	1.4	1.4	1.4
40	1.0	1.3	1.4	1.4	1.5	1.5	1.5	1.5	1.6	1.7
50	1.0	1.3	1.5	1.6	1.7	1.7	1.8	1.8	1.8	2.0
60	1.0	1.4	1.7	1.8	1.9	2.0	2.1	2.1	2.1	2.5
70	1.0	1.5	1.9	2.1	2.3	2.4	2.5	2.6	2.6	3.3
80	1.0	1.7	2.1	2.5	2.8	3.0	3.2	3.3	3.5	5.0
90	1.0	1.8	2.5	3.1	3.6	4.0	4.4	4.7	5.0	10

90 % of the total calculation time is used. So, possibly the total speed-up offered by the LBM is even higher.

In addition, the LBM CUDA code is currently found to be approximately 7.5 times faster than the serial LBM, see Section 4.5.3. This means that the GPU version of the LBM is around 30 times faster than the serial FDM flow solver. So, the LBM CUDA code offers a potential speed-up in the range of 5.

The LB solver demonstrates a massive 500-time increase in speed compared to COMSOL, a 30-time increase compared to Fluent and a 4-time increase compared to both PHOENICS and the TESA flow solver proving it to be the most efficient transient flow solver along with other merits of the LBM.

4.6 Summary

In this chapter, the algorithm side of the purpose-built LBM has been discussed. The method can be broken down into simple subroutines each handling a separate

task. Because the LBM is written in a dimensionless form, the variables need to be scaled when passed in and out. The main building blocks of the method are the collision and streaming subroutines from which the collision is the most time-consuming operation. The newly proposed 3D moment-based boundary method can handle both velocity and pressure Dirichlet type boundaries. They can be written generally for each face, edge and corner boundary with the help of the Euler angles by exploiting the symmetry properties of the lattice. That reduces the chance of making typing errors when maintaining the code.

The parallelisation aspect has also been covered. The LB code has been parallelised using MPI for the CPU use. The LB code has been modified to include the calls to the MPI libraries developed by Kao that pass the variable information from one processor to another. The GPU version of the LB code has also been developed. The performance of the CUDA LB code has been tested against the serial version revealing a 7.5-time speed-up for a 128^3 problem. CUDA platform has been chosen over directive-based models because it offers a higher yield performance-wise with a little more effort in programming. However, currently only the MPI-enabled version of the LB code has been successfully coupled to TESA, the next step being a complete coupling of the GPU LB code and TESA, see Section 6.2.

The performed studies of the strong and weak scaling have revealed that the developed LB code scales very well in both cases. It demonstrates high efficiencies around 90 % of the ideal strong scaling value for all the grid sizes. The calculation times in weak scaling plateau out consistently for all grid sizes at higher number of processors used showing the expected behaviour.

Performance analysis has been conducted for different methods to see how

efficient the developed LBM is compared to other numerical schemes. The results clearly show why the LBM is the best choice for solving transient large-scale convection and diffusion problems. The most important number from the analysis is of course the time ratio of the current flow solver used in TESA and the developed LB code. It has been shown that the LBM outperforms the FDM flow solver by a factor of 4, which means that by replacing the flow solvers the total simulation time would decrease at least 2.5 times.

Chapter 5

MODEL VALIDATION AND RESULTS

5.1 Introduction

In this section, the newly derived 3D Moment Method is validated on various benchmark cases in 2D and 3D testing the conditions at the faces, edges and corners of the domain. The 2D simulations are used to verify the developed method, while the 3D simulations are used for the assessment of the method's accuracy and stability.

One of the tests in 2D is the relaxation time independence study performed using the 2D Poiseuille flow. Moreover, it is linked to the exact recovery of the no-slip condition for the velocity on the wall. Because the solution of the developed 2D Poiseuille flow is essentially 1D, a relatively small numerical grid of $33 \times 3 \times 3$ can be used in the calculations to test for the τ dependence and the no-slip recovery. A grid size of $129 \times 129 \times 3$ is selected for the 2D lid-driven cavity flow to match the the meshes employed by other authors [175; 176]. Velocity profiles

along the centerlines as well as the extreme values of the stream functions are compared in order to validate the method.

For the 3D convergence studies, the grid size is being varied proportionally with the relaxation time while fixing the Reynolds number. A changing grid of $N_x \times N_y \times N_z$ up to $513^2 \times 3$ is used for the 3D duct flow while fixing the velocity at 0.1. Force is applied to the fluid domain in the z direction in both 2D and 3D calculations. A zero Neumann boundary condition for the flow variables is applied to the redundant dimensions. A varying grid of $N_x \times N_y \times N_z$ up to 257^3 is used for the 3D lid-driven cavity flow while fixing the Reynolds number at Re = 1000.

5.2 2D validation of the Moment Method

The relaxation time dependence study is performed to see if the Moment Method is τ-independent, and the results are shown in Figure 5.1. The Moment Method with both collision schemes shows a whole range of the relaxation time values for which the solution does not change in general, apart from the small region where τ approaches its asymptotic lower limit, $\tau_{min} = 1/2$. The lower limit, τ_{min}, is obtained from (3.38) and simply restricts the fluid viscosity from becoming negative.

The carried out relaxation time dependence study confirms that the Moment Method has no slip at the walls, and it is τ-independent, see Figure 5.2.

The velocity is kept constant during the study, Reynolds number is moderately high for small τ, but decreases linearly as τ increases. The relative slip velocity for the modified bounce-back increases monotonically with τ irrespective of the

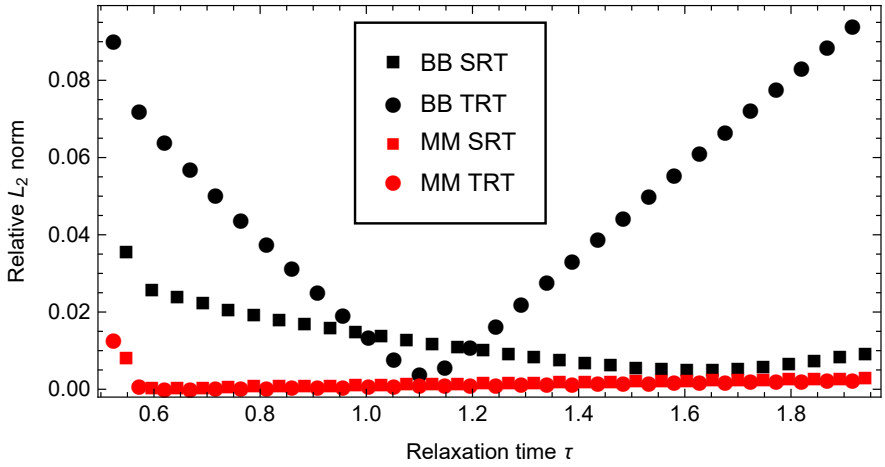


Figure 5.1: τ independence study for the the Moment Method and the modified bounce-back rule using SRT and TRT collision schemes in a 3D developed duct flow case.

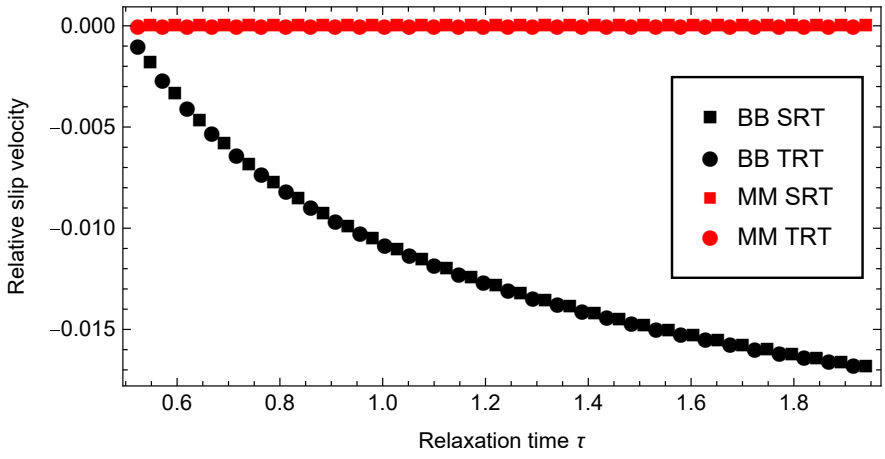


Figure 5.2: Relative slip velocity dependence on the relaxation time in a 2D Poiseuille flow case at the wall.

collision scheme used here.

The 2D Poiseuille flow is selected to test if the no-slip velocity condition on the wall is recovered exactly, which is an issue with the bounce-back rule. To highlight the difference between slip and no-slip, the calculations are also performed using the modified bounce-back rule. The results are shown in Figure 5.3. The Moment Method velocity profiles show very good agreement with the analytical solution compared to the bounce-back rule results. In fact, the LBM together with the Moment Method can recover the simplest 2D Poiseuille flow velocity profile exactly, see Figure 5.4. The inset in Figure 5.3 shows a zoomed in area of the flow next to the wall. The artificial slip at the wall node is apparent for the bounce-back rule while the no-slip condition is recovered exactly by the Moment Method.

5.2.1 Oscillatory flow around a cylinder

To test the transient nature of the LBM algorithm and the Moment Method flow past a cylinder is investigated. In this benchmark case at a critical Reynolds number the flow detaches from the obstacle forming vortices in the wake zone. The vortex shedding can be characterised by the Strouhal number, $\mathrm{St} = \dfrac{L\omega}{u_\infty}$, where ω is the shedding frequency and u_∞ is the free stream velocity. One way of comparing the obtained periodic solution to the results found in the literature is by plotting the relation between the Reynolds and Strouhal numbers. Figure 5.5 shows a comparison between the present model and experimental results by Williamson [177] and numerical results by Posdziech and Grundmann [178] at different Reynolds numbers. The grid size for this 2D transient benchmark case is $L \times H = 1200 \times 600$. Numerical results are obtained at five different Re ranging

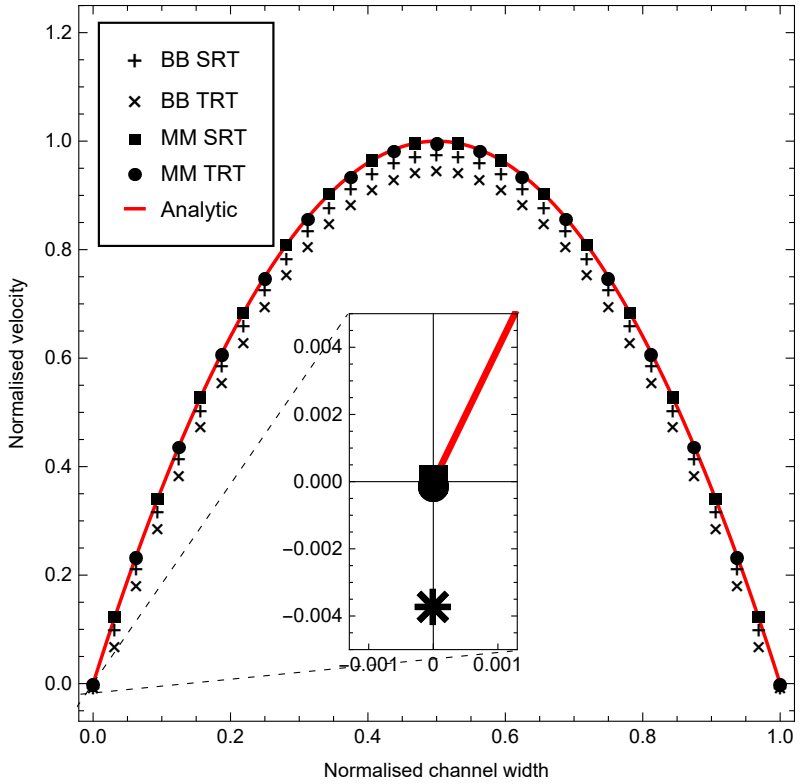


Figure 5.3: 2D Poiseuille flow velocity profile showing the exact recovery of the no-slip condition for the Moment Method and the artificial slip for the modified bounce-back rule.

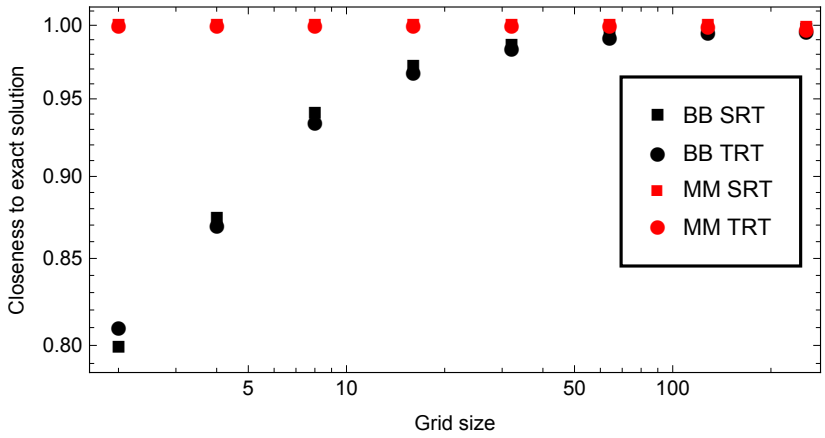


Figure 5.4: Grid convergence study for the the Moment Method and the modified bounce-back rule using SRT and TRT collision schemes in a 2D Poiseuille flow case.

from 50 to 250 which is the laminar regime of the flow. The grid size and the free stream velocity are fixed while the dimensionless viscosity varies for different Re. Free stream velocity, $u_\infty = 0.1$, is applied to the west, north and south boundaries, and a pressure outlet is used at the east boundary. A cylinder of diameter $D = H/15$ is placed at $(H/2, H/2)$ with the wake zone of length $22D$. The results show a good agreement between the data. The small differences might be attributed to the chosen mesh size and possible boundary effects. Also, the circular cylinder is not perfectly circular because no interpolation is used to describe the obstacle boundaries.

5.2.2 The 2D lid-driven cavity flow

The 2D lid-driven cavity flow is selected as one of the benchmark cases to test the ability of the Moment Method to describe shear flows, see Figure 5.6. Velocity profiles along the centerlines as well as the extreme values of the stream functions have been compared to the data from the literature [175; 176] at Reynolds numbers Re = 100 and Re = 1000. The results shown in Figure 5.7 and Table 5.1 are in very good agreement with the results obtained by Ghia *et al.* [175], who were the first ones to do a comprehensive study on the lid-driven cavity flow, but more so with the results from Botella and Peyret [176], who used a spectral method in their work.

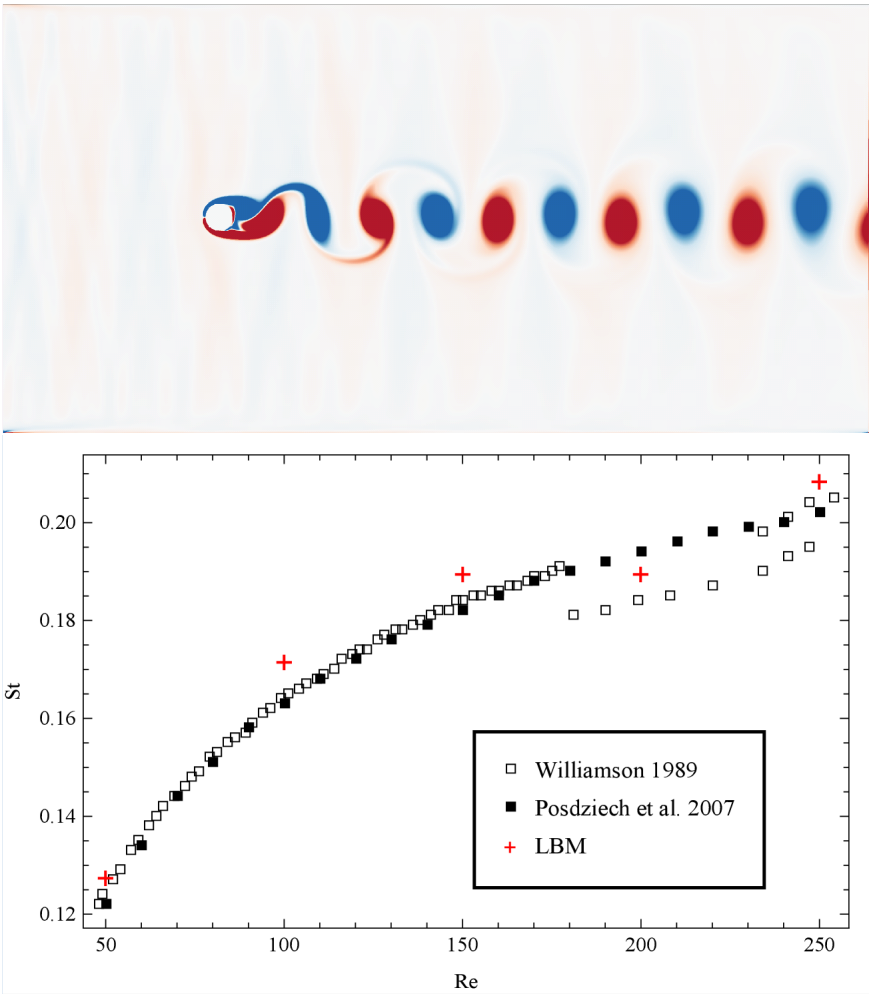


Figure 5.5: Vortex street (top) and comparison between the present results and data from the literature (bottom). Experimental results are from Williamson [177] and numerical results are from Posdziech and Grundmann [178]. [170]

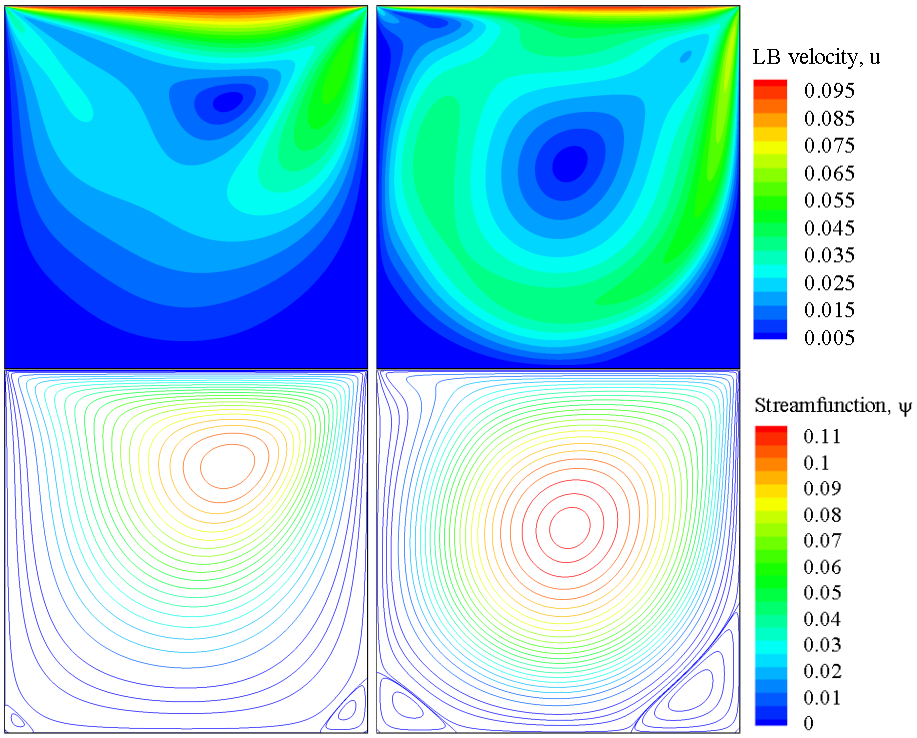


Figure 5.6: Lid-driven cavity flow. Velocity field (top) and streamlines (bottom) showing the nature of the flow in the square cavity at Reynolds numbers Re = 100 (left) and Re = 1000 (right).

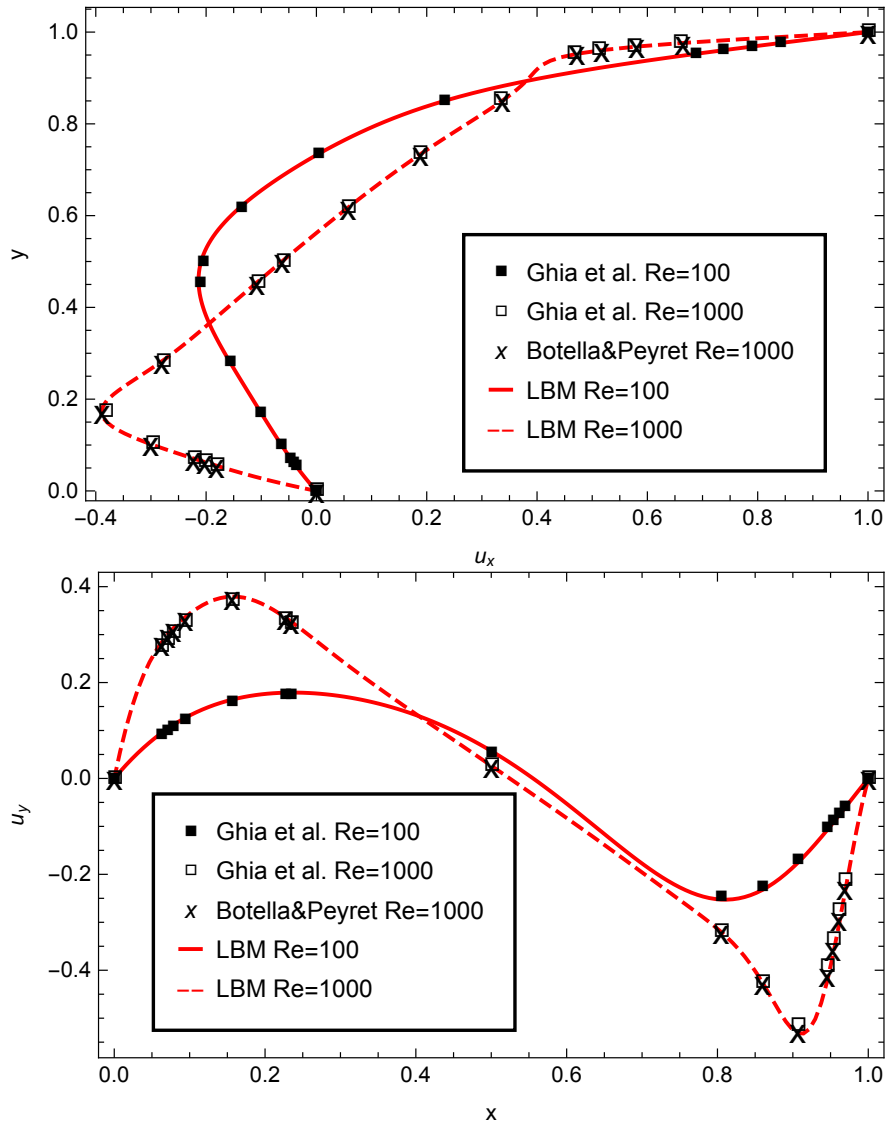


Figure 5.7: A comparison of horizontal (top) and vertical (bottom) velocity along the centerline at Reynolds numbers Re = 100 and Re = 1000 on a 129^2 grid.

Table 5.1: Comparison of the extreme values of the stream function at Reynolds numbers Re=100 and Re=1000 on a 129^2 grid.

	Primary vortex		Secondary vortex (BL)		Secondary vortex (BR)	
Reference	Re=100	Re=1000	Re=100	Re=1000	Re=100	Re=1000
Present, ψ	-0.103402	-0.119244	$1.51760 \cdot 10^{-6}$	$2.30158 \cdot 10^{-4}$	$1.21731 \cdot 10^{-5}$	$1.72769 \cdot 10^{-3}$
x, y	0.6172, 0.7344	0.5313, 0.5625	0.0313, 0.0391	0.0859, 0.0781	0.9453, 0.0625	0.8594, 0.1094
Ghia *et al.* ψ	-0.103423	-0.117929	$1.74877 \cdot 10^{-6}$	$2.31129 \cdot 10^{-4}$	$1.25374 \cdot 10^{-5}$	$1.75102 \cdot 10^{-3}$
[175] x, y	0.6172, 0.7344	0.5313, 0.5625	0.0313, 0.0391	0.0859, 0.0781	0.9453, 0.0625	0.8594, 0.1094
Botella & ψ	-	-0.118937	-	$2.33453 \cdot 10^{-4}$	-	$1.72972 \cdot 10^{-3}$
Peyret[176] x, y	-	0.5308, 0.5652	-	0.0833, 0.0781	-	0.864, 0.1118

BL - bottom left, BR - bottom right

5.3 3D validation of the Moment Method

The grid convergence studies in 3D are carried out to check if the boundary method is not degrading the second order accuracy of the LBM. 3D cases are chosen as the best representatives of the method's accuracy due to all the boundaries being included.

Because the LBM together with the Moment Method can recover the simplest 2D Poiseuille flow velocity profile exactly, see Figure 5.4, unlike the bounce-back rule, a 3D duct flow is chosen to test the grid convergence.

The analytical formula for the velocity in a developed 3D square duct flow has been adopted from the theory of elasticity when talking about the deflection surface of a membrane [179]. It has the following form:

$$u_z = \frac{4L^2 F_z}{\pi^3 \rho\nu} \sum_{i=1,3,5,\ldots}^{\infty} \frac{(-1)^{\frac{i-1}{2}}}{i^3} \cos\left(i\pi \frac{x}{L}\right) \left[1 - \frac{\cosh\left(i\pi \frac{y}{L}\right)}{\cosh\left(i\pi \frac{1}{2}\right)}\right]. \tag{5.1}$$

The expression for the velocity includes the information of the geometry of the square duct or the width L, fluid properties or the dynamic viscosity $\rho\nu$ and applied conditions or the driving force F_z. The infinite sum is used to account for the rectangular shape of the duct. The coordinates in the velocity solution (5.1) range from $-\frac{L}{2}$ to $\frac{L}{2}$ so that the origin $(0, 0)$ is placed in the middle of the duct. If the zeros are substituted into (5.1) then the formula for the maximum velocity can be obtained:

$$u_{\max} = \frac{4L^2 F_z}{\pi^3 \rho\nu} \sum_{i=1,3,5,\ldots}^{\infty} \frac{(-1)^{\frac{i-1}{2}}}{i^3} \left[1 - \frac{1}{\cosh\left(i\pi \frac{1}{2}\right)}\right]. \tag{5.2}$$

Analytical velocity values of the 3D duct flow are recovered beyond the ma-

chine precision error accuracy and are not affecting the grid convergence test study. Truncating the infinite series in (5.1) at $i = 450$ leads to a relative error of less than 10^{-8} which is beyond the machine single precision, 10^{-7}, used in this work. To evaluate the deviation from the exact solution, the L_2 relative error norm is calculated for the velocity field,

$$L_2 = \sqrt{\frac{\sum_i (u_i - u_i^*)^2}{\sum_i u_i^{*2}}}, \tag{5.3}$$

where u_i^* is the exact solution at the calculation domain node i.

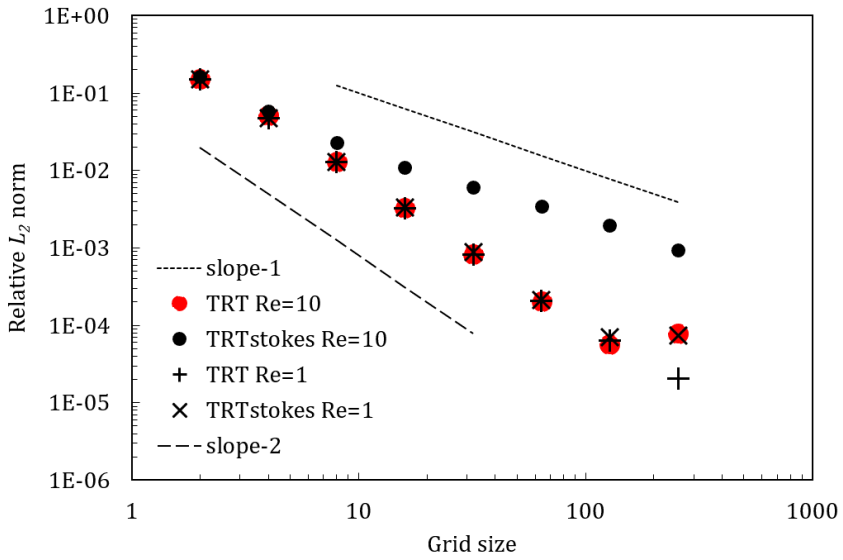


Figure 5.8: Grid convergence study for the the Moment Method using the TRT and TRT-Stokes collision schemes in a 3D developed duct flow case.

The results of the grid convergence study are shown in Figure 5.8. The Moment Method is at least second order accurate in the region where the grid size error is dominant. The error data points follow the line of slope 2 shown as a long-dashed line. Moreover, neglecting the nonlinear velocity terms in low-Re

case, Re = 1, yields the same accuracy as using the full expressions. However, the accuracy of the Stokes approach quickly deteriorates beyond the first order as the Reynolds number increases, see the dashed line in Figure 5.8.

To put the obtained Moment Method results into perspective, they are compared with the results of the half-way bounce-back scheme and Hecht and Harting [134], see Figure 5.9. Despite all the boundary methods reportedly being second order accurate, the developed Moment Method overall shows better accuracy than the half-way bounce-back scheme and the method proposed by Hecht and Harting. It should be noted, however, that the chosen range of the viscosity values in this convergence study is influencing the accuracy of the half-way bounce-back scheme. Here it shows only first order accuracy. Although the half-way bounce-back scheme is simpler and computationally less expensive than the Moment Method, its optimal viscosity region is not as wide as that of the Moment Method meaning that the calculation time to achieve the same accuracy is much longer. Choosing a 10-times smaller viscosity and hence 10-times smaller time step restores the second order accuracy of the half-way bounce-back scheme, see the 'pluses' in Figure 5.9. Another comparison of the Moment Method and the half-way bounce-back scheme can be found in a study conducted by Mohammed *et al.* [142], where they find that the results of the Moment Method are in a closer agreement to the spectral method's than those obtained with the half-way bounce-back scheme.

In addition to the 3D duct flow case, the grid convergence study is also performed using the 3D lid-driven cavity flow. Reynolds number is fixed at Re = 1000 and the lid velocity is kept constant at $u_{\text{lid}} = 0.1$ so that only the viscosity changes proportionally with the grid size. Because the lid driven cavity flow does not have

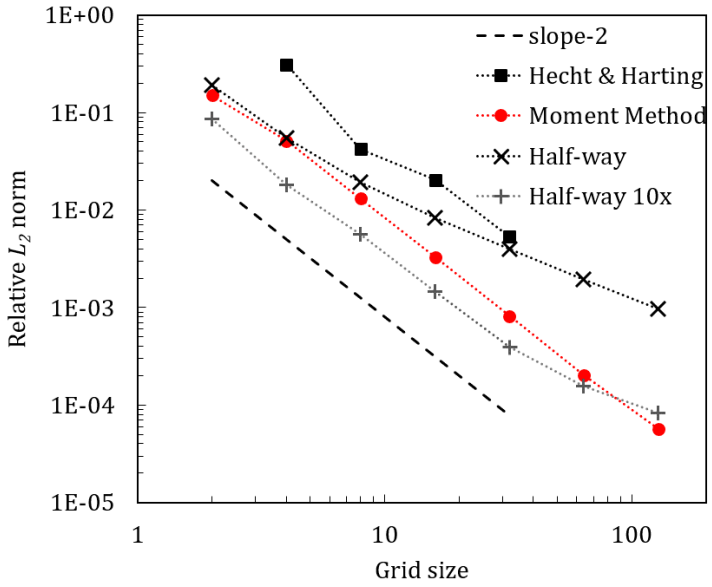


Figure 5.9: Comparison of the grid convergence for the 3D developed duct flow between the present Moment Method, half-way bounce-back scheme and the method proposed by Hecht and Harting [134].

an analytical solution, the variable field values obtained on the finest grid of 257^3 are used as a reference. Figure 5.10 shows the convergence study results gathered from comparing the velocity field values at different grid sizes. For this purpose, (5.3) is expanded to a 3D vector field calculation as,

$$L_2 = \sqrt{\frac{\sum_i \left((u_{xi} - u^*_{xi})^2 + (u_{yi} - u^*_{yi})^2 + (u_{zi} - u^*_{zi})^2 \right)}{\sum_i (u^{*2}_{xi} + u^{*2}_{yi} + u^{*2}_{zi})}}, \tag{5.4}$$

where $u^*_{\alpha i}$ is the velocity field value on the finest grid, here 257^3. The error data points in Figure 5.10 show the same trend as the line of slope 2, meaning that the Moment Method is second order accurate in 3D.

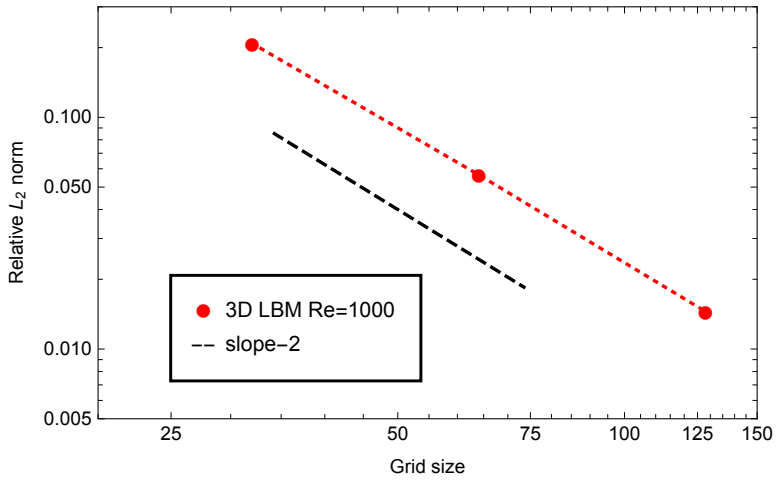


Figure 5.10: Grid convergence study for the the Moment Method using TRT collision scheme in a 3D lid-driven cavity flow case.

5.4 Differentially heated cavity flow

A 2D LB model has been coupled to a simple finite-difference heat transfer code to simulate differentially heated cavity (DHC) flow. Two cases are considered – constant temperature of $T_L = 0$ and $T_H = 1$ on opposing walls vertically and horizontally. The first case models the Rayleigh–Benard convection (RBC), where the periodic boundary condition is imposed on the side walls. The second scenario is an enclosed cavity with a moving lid. Both setups are shown in Figure 5.11. Temperature difference is imposed on the opposing walls, and the remaining (if any) walls are insulated. No-slip boundary conditions for velocity are imposed on all walls.

For the RBC, the main characteristic of the physical problem is the Rayleigh number, which is expressed as,

$$\mathrm{Ra} = \frac{g\beta}{\nu\alpha}\left(T_s - T_{\mathrm{ref}}\right) L^3, \tag{5.5}$$

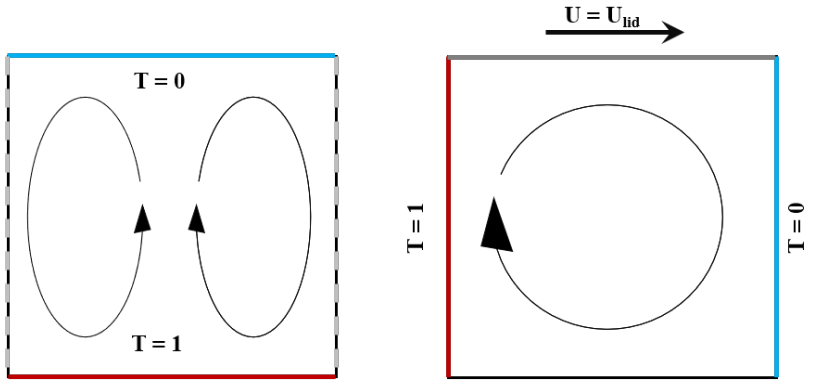


Figure 5.11: Schematic drawing of the differentially heated cavity flow. Rayleigh–Benard convection (left) and moving lid cavity (right).

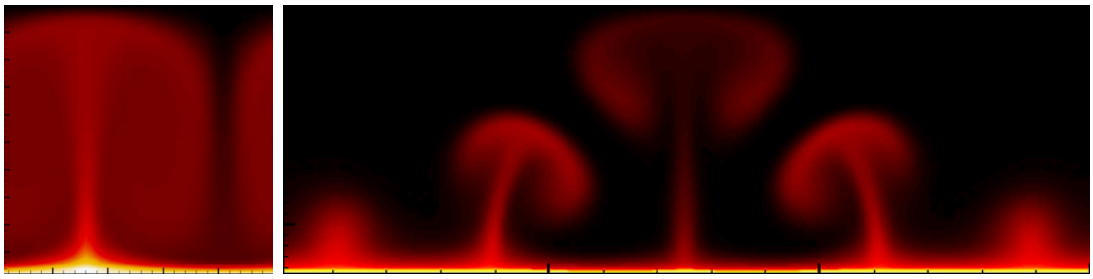


Figure 5.12: Rayleigh–Benard convection in a periodic domain. Single plume in a low-Ra case (left), multiple plumes in a mid-Ra case (right).

where g, β, ν and α are gravitational acceleration, thermal expansion coefficient, kinematic viscosity and thermal diffusivity, respectively. T_s is the surface temperature and T_{ref} is the reference or ambient temperature. L is the characteristic length. The thermal expansion coefficient is usually taken to be inversely proportional to the temperature $\beta = \frac{1}{T}$. For relatively small values of Ra, the system is stable as seen in Figure 5.12 on the left. For medium to high Ra, the system becomes unstable and does not converge to a steady-state solution, see Figure 5.12 on the right.

Next, the moving lid differentially heated cavity flow is described. In the stationary case, when the lid velocity $U_{\text{lid}} = 0$, the main acting phenomenon is

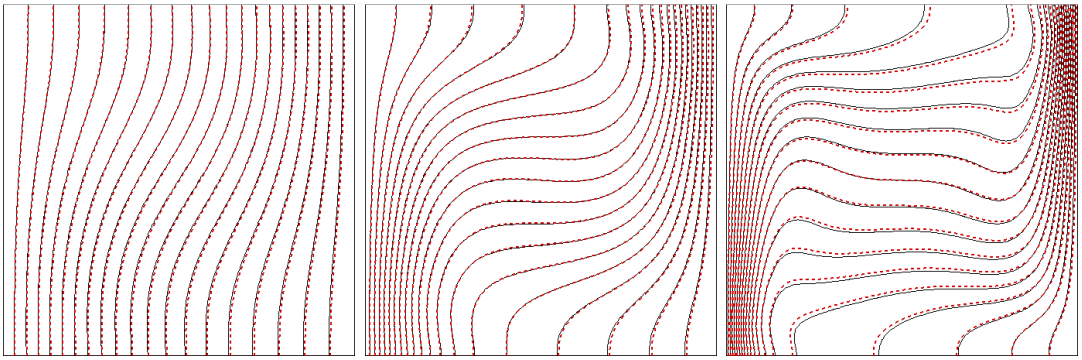


Figure 5.13: Comparison of the steady-state temperature distribution in the differentially heated cavity between the LBM (dashed red lines) and COMSOL (solid black lines). Ra = 10^3 (left), Ra = 10^4 (middle), Ra = 10^5 (right).

natural convection. Because of the symmetric setup of the system, the solution of the DHC problem also possesses central symmetry. Temperature field in the cavity at different Ra is shown in Figure 5.13. It can be seen that the distribution of isotherms changes from vertical to horizontal in the middle of the cavity as Ra increases, with heat transfer changing from conduction to convection dominated. The vertical wall boundary layers become progressively thinner. The results have been compared with the fine-mesh steady-state solution obtained with COMSOL and they show a very good agreement. Additionally, the maximum velocity values and their locations on the vertical and horizontal mid-lines have been listed in Table 5.2 and compared to the benchmark solutions provided by de Vahl Davis [180] and Markatos and Pericleous [181] also showing a very good agreement. The velocities are normalised using a factor of L/a, where L is the cavity width and a is the thermal diffusivity.

However, for non-zero lid velocities depending on the magnitude of the lid velocity the forced convection is dominant and the temperature distribution in the cavity loses its symmetry as shown in Figure 5.14.

Table 5.2: Comparison of the DHC solutions at different Rayleigh numbers.

	Ra $= 10^3$		Ra $= 10^4$		Ra $= 10^5$	
Reference	u_{max}	v_{max}	u_{max}	v_{max}	u_{max}	v_{max}
Present, x, y	3.500 0.5, 0.815	3.500 0.180, 0.5	16.10 0.5, 0.815	20.29 0.130, 0.5	34.30 0.5, 0.845	71.40 0.072, 0.5
COMSOL, x, y	3.656 0.5, 0.815	3.689 0.180, 0.5	16.24 0.5, 0.815	19.79 0.125, 0.5	35.71 0.5, 0.840	73.60 0.075, 0.5
Markatos & Pericleous[181], x, y	3.544 0.5, 0.832	3.593 0.168, 0.5	16.18 0.5, 0.832	19.44 0.113, 0.5	35.73 0.5, 0.857	69.08 0.067, 0.5
de Vahl Davis [180], x, y	3.649 0.5, 0.813	3.697 0.178, 0.5	16.18 0.5, 0.823	19.62 0.119, 0.5	34.73 0.5, 0.855	68.59 0.066, 0.5

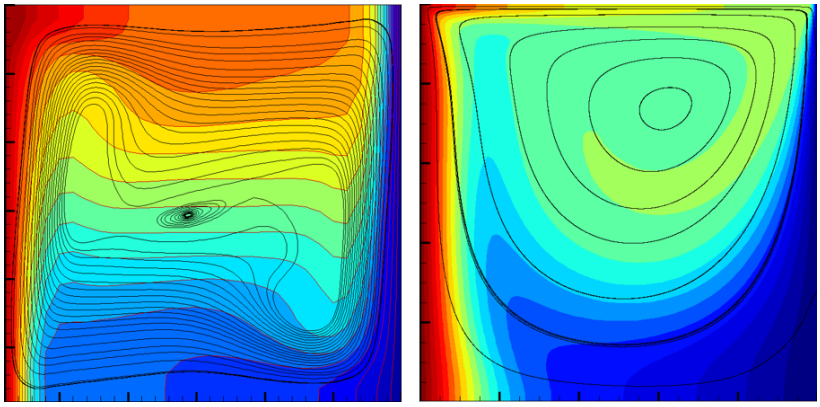


Figure 5.14: Steady-state temperature distribution and velocity field streamlines in the moving lid differentially heated cavity at Ra = 50000. Stationary lid (left) and moving lid (right).

5.5 Solidification in a DHC

The DHC model is slightly modified to include a simple solidification process (5.6) in the heat transfer solver. It states that the liquid fraction decreases linearly with the temperature when the temperature is in a certain region, $T_L \leq T \leq T_H$, and reaches zero when the temperature falls below the set threshold, $T < T_L$. The liquid turns solid at $f_{liq} = 0.5$ so that the bounce-back scheme can be employed. The buoyancy force, $\mathbf{F}_b = \rho \mathbf{g} \beta (T - T_{\text{ref}}) \cdot (1 - f_{\text{sol}})$, acts only in the non-solidified region.

$$f_{\text{liq}} = \max\left(0, \min\left(1, \frac{T - T_L}{T_H - T_L}\right)\right). \tag{5.6}$$

The results are compared to those of Voller and Prakash [182] showing a good agreement, see Figure 5.15. Ra = 10^4 and the Pradtl number that expresses the balance between the energy transport through momentum diffusivity and thermal diffusivity, Pr = $\frac{\nu}{\alpha}$, is set to Pr = 10^3. The slight mismatch can be explained by the different approaches taken to describe the flow in the mushy zone. In the present model, the viscous boundary layer of the solidified region slows down the

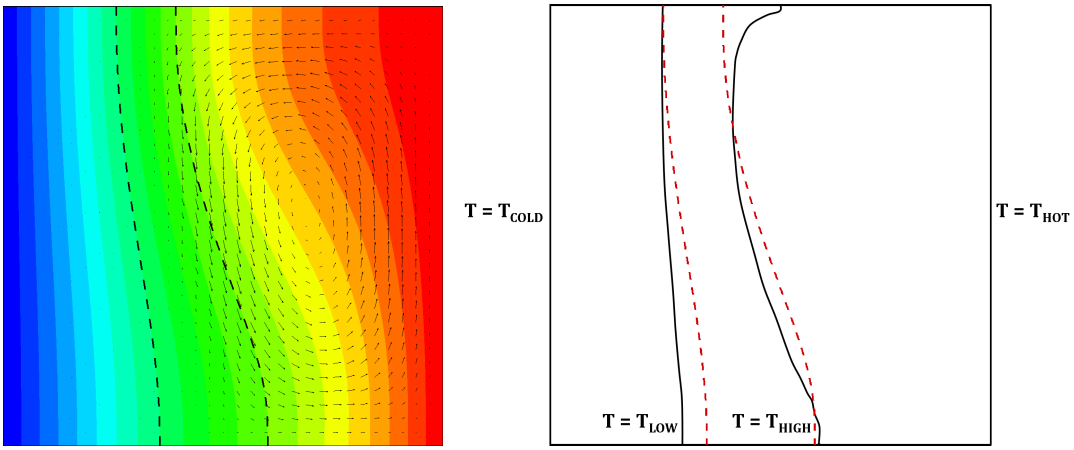


Figure 5.15: Solidification in the DHC. Left: temperature field showing the mushy region and velocity vectors showing convection. Right: comparison of the mushy region between the LBM (dashed red lines) and Voller [182] (solid black lines).

flow, while Voller and Prakash use a resistive force $\mathbf{F}_{\text{res}}$ to suppress the flow,

$$\mathbf{F}_{\text{res}} = -C_1 \rho \mathbf{U} \frac{(1 - f_{\text{liq}})^2}{f_{\text{liq}}^3 + q_1}, \tag{5.7}$$

where $C_1 = 1.6 \cdot 10^5$ and $q_1 = 10^{-1}$ are scaling and stability constants.

5.6 Undercooled crystal growth

The LBM and the enthalpy method described in Section 4.4.2 are fully coupled together to simulate 2D crystal growth in an undercooled melt. Physical properties of liquid aluminium-like material, see Table 5.3, are chosen as input parameters for the model. The growth takes place in a low Peclet number regime, $\text{Pe} = \dfrac{UL}{\alpha}$, where the diffusive transport is comparable or dominant over the convective transport. A square mesh of 1024×1024 is used for both solvers. At the beginning of the calculation, the entire domain is in a metastable liquid state and at a constant undercooled temperature with a constant bulk flow velocity. Solidi-

Table 5.3: Physical properties of liquid aluminium-like material.

Liquid density, ρ (kg/m^3)	2500
Viscosity, μ (Pa $\cdot$ s)	0.00285
Specific heat capacity, c_p (J/kg $\cdot$ K)	1200
Latent heat, ΔH (J/kg)	400000
Thermal conductivity, k (W/m $\cdot$ K)	90
Fusion/melting temperature, T_f (K)	933

fication begins with the nucleation of a single-cell seed at solidifying temperature in the middle of the domain. An inlet of homogeneous velocity $U = (U_{\text{in}}, 0)$ and a pressure outlet are defined on the west and east side of the domain, respectively. The north and south walls have the free-slip condition for the velocity. For the domain boundaries, a hydrodynamic scheme proposed by Zou and He [131] is applied. For the growing crystal interface a half-way bounce-back heuristic scheme is used due to the complex geometry shape. For stagnant flow, zero heat flux is applied to all the domain boundaries. In the forced convection case, the bulk undercooled temperature is applied at the inlet. The enthalpy method is written using a finite differencing scheme.

5.6.1 Single crystal growth in stagnant melt

From the numerical simulation of the crystal growth in a static melt, the normal tip velocities are obtained, which are later used to scale the tip velocities in forced convection. It can be seen from Figure 5.16 that there is no preferential growth direction. The crystal grows equally in all four directions because of the four-fold symmetry, see the coefficient in front of the interface orientation angle θ in (4.14). Furthermore, the growth is suppressed in regions where the temperature is higher, and that is at the base of the dendrite arms due to the negative curvature,

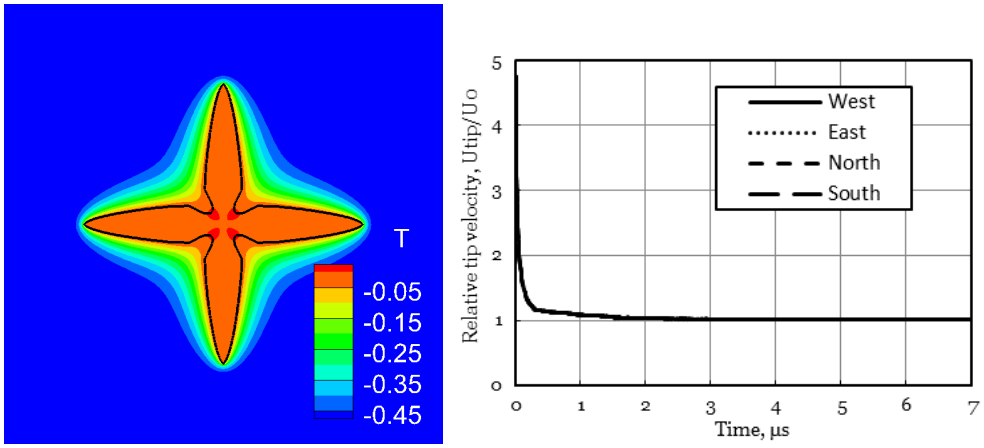


Figure 5.16: Thermal field of the growing crystal (left) and time histories of the relative tip velocities in a static melt (right) with undercooling of $T_{\mathrm{uc}} = -0.5$, time $t \approx 7$ µs.

see (4.14). The seed grows faster at the beginning because of the beneficial low surrounding temperature. As the crystal gets bigger and the thermal boundary layer develops, the growth speed decreases until converging to the steady-state growth predicted by microscopic solvability theory. The theory states that the analytic solution for the undercooled dendritic growth can be expressed by the means of the dendrite tip velocity U_{tip} and its radius R_{tip} as

$$\begin{aligned} U_{\mathrm{tip}} R_{\mathrm{tip}} &= C, \\ U_{\mathrm{tip}} R_{\mathrm{tip}}^2 &= C, \end{aligned} \tag{5.8}$$

where C is a constant. Figure 5.16 also shows the time evolution of the tip velocities scaled by the steady-state growth speed U_0.

5.6.2 Forced convection crystal growth

When flow is applied, the morphology of the growing crystal changes compared to the case of a static melt. The incoming flow introduces a preferential growth

direction by decreasing the thermal boundary layer upstream and increasing it downstream. This, in turn, results in a greater tip velocity for the upstream arm and stunting of the downstream arm. The applied convection also has a negative effect on the perpendicular arm growth as for they decrease.

Figure 5.17 shows thermal fields with different magnitudes of forced convection and how the morphology of the growing crystal is affected. The greater the inlet velocity, the greater the difference between tip velocities and furthermore the arm lengths. Figure 5.18 shows the time histories of the relative tip velocities, and the aggregate results. It can clearly be seen that the upstream or west tip velocity increases with increasing inlet velocity. Whereas, the east or downstream tip velocity approaches zero, with greater forced convection. Both north and south tip velocities experience the same effect. The results for the 2D undercooled crystal growth are in a good qualitative agreement with the literature data [24–28; 55].

5.7 Large-scale results

Having validated the LB code using various benchmark cases, the developed LB code can be applied to model large-scale multi-physics problems. Furthermore, the simulated microstructure results can be compared to the experimental data obtained on a macroscale, verifying the developed model even further and entering the final stage of the numerical modelling which involves using the model to predict and investigate various physical phenomena in certain multi-physics systems.

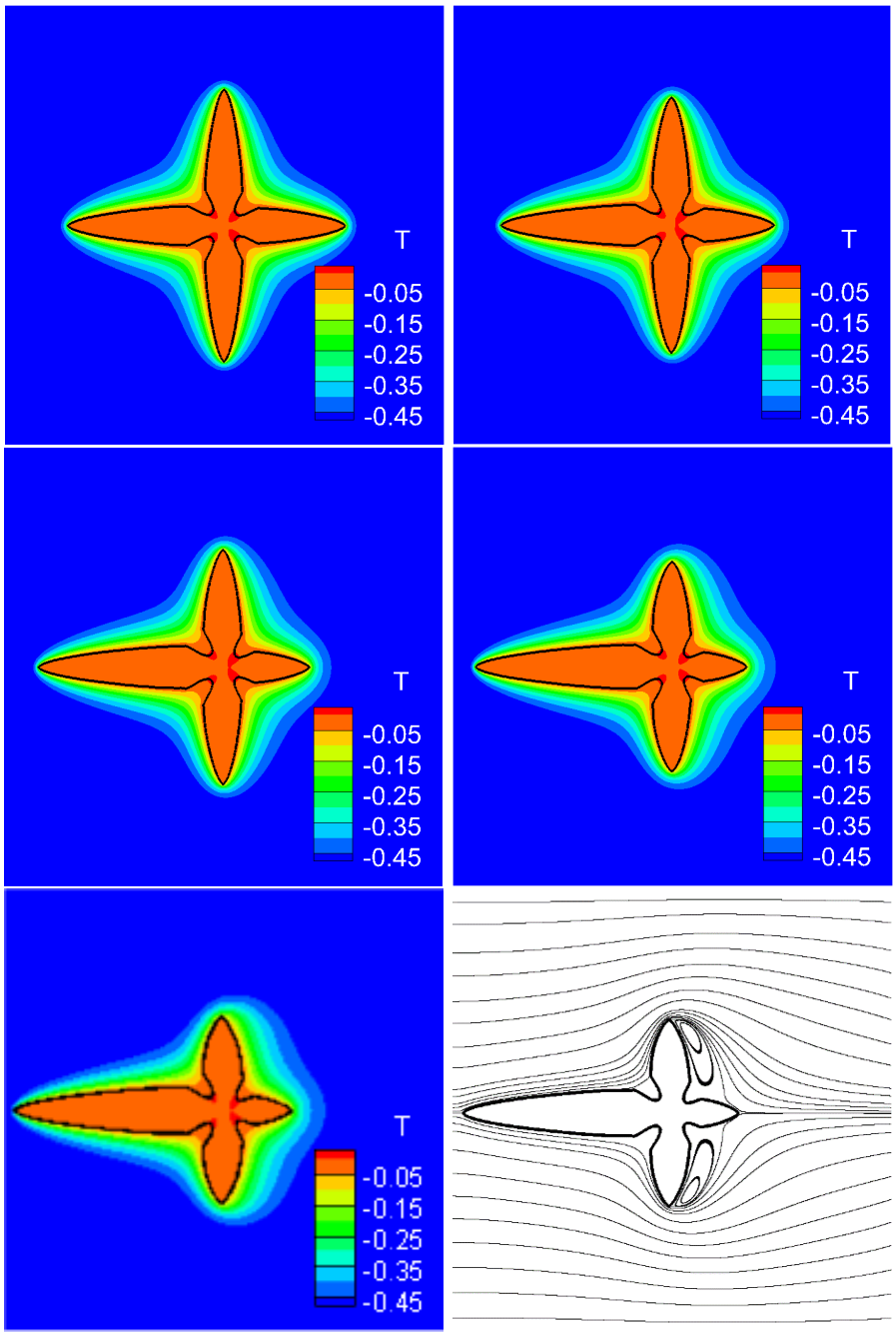


Figure 5.17: Thermal field of the growing crystal in a convectional melt with undercooling of $T_{\mathrm{uc}} = -0.5$ at different inlet velocities. Top left: $U_{\mathrm{in}} = 0.0025$, top right: $U_{\mathrm{in}} = 0.005$, middle left: $U_{\mathrm{in}} = 0.0075$, middle right: $U_{\mathrm{in}} = 0.01$, bottom left: $U_{\mathrm{in}} = 0.0125$, bottom right: velocity streamlines, all at time $t \approx 7$ µs.

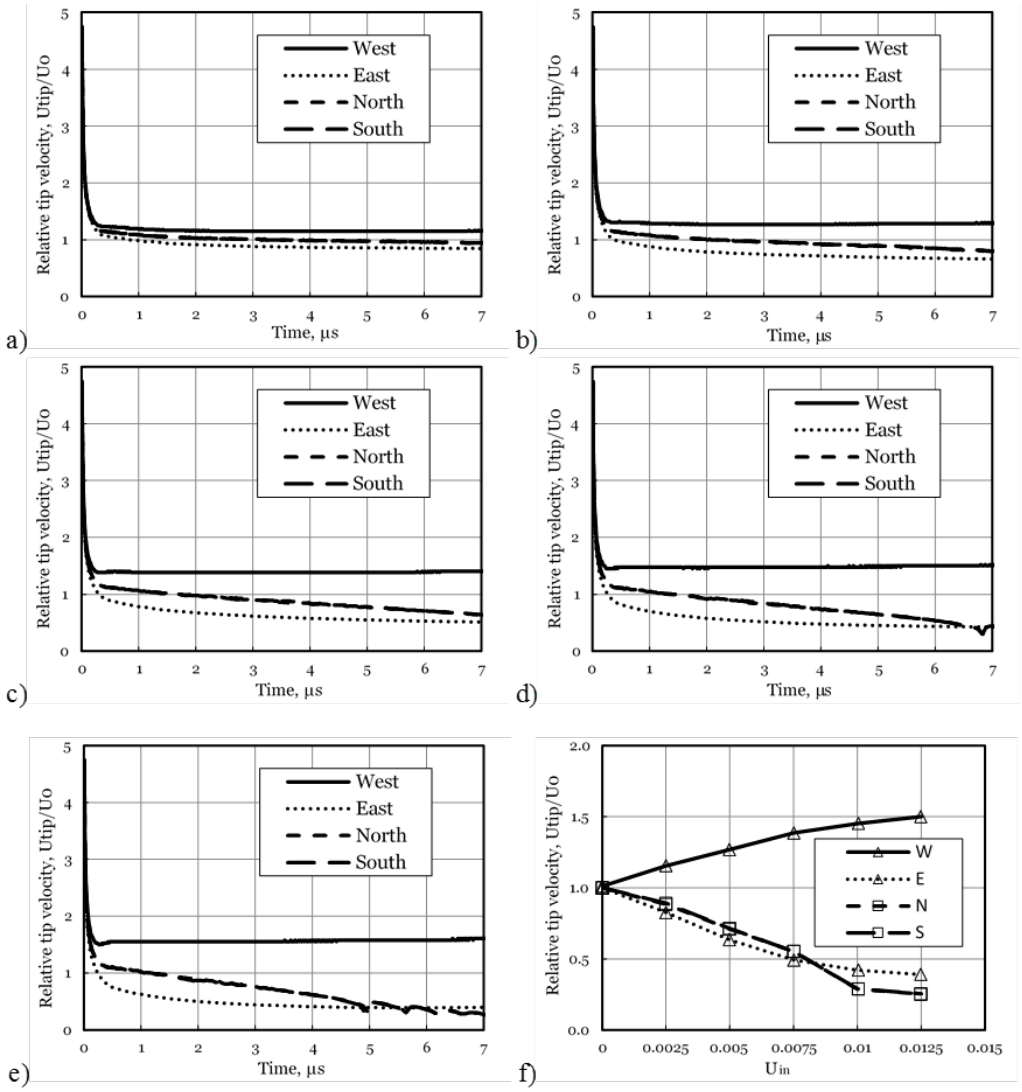


Figure 5.18: Time histories of the relative tip velocities of the growing crystal with undercooling temperature of $T_{\text{uc}} = -0.5$. (a) $U_{\text{in}} = 0.0025$, (b) $U_{\text{in}} = 0.005$, (c) $U_{\text{in}} = 0.0075$, (d) $U_{\text{in}} = 0.01$ and (e) $U_{\text{in}} = 0.0125$. (f) Steady-state relative tip velocities of the growing crystal.

5.7.1 Free dendritic growth

As discussed earlier in Section 2.2, the morphology of a freely growing crystal in 3D differs in complexity compared to the 2D case that was covered in the previous section. Utilising the available computing resources in the research group, a low undercooled crystal growth in 3D using a grid of one billion elements is simulated demonstrating the full coupling between the CA and LBM. Two cases have been simulated, free growth with and without forced convection, and the differences have been discussed. The material properties of the alloy are given in Table 5.4. The problem setup is as follows, a single seed is placed in the middle of the undercooled, $T_{\mathrm{uc}} = 20$ K, domain and solidification process begins extracting the latent heat and the solute into the melt. Although the LB flow solver is disabled in the stagnant flow case, the solidified symmetric equiaxed dendrite serves as a basis for morphological comparison when simulating growth in forced convection. An inlet velocity of 400 µm/s is applied forcing the thermal and solutal boundary layer to skew downstream. This leads to preferential growth in the upstream direction and stunted growth downstream. This behaviour is in accordance to the results reported in the literature [30–35; 94]. Thanks to the large simulated domain size, the dendritic arms can be captured in a high detail revealing ternary and even quaternary arms as seen in Figure 5.19.

5.7.2 Alloy solidification in DHC

Another example of a successful coupling between the CA and LBM is the alloy solidification subject to a horizontal temperature gradient [170], an experiment of which has been conducted by the HZDR (*Helmholtz-Zentrum Dresden-*

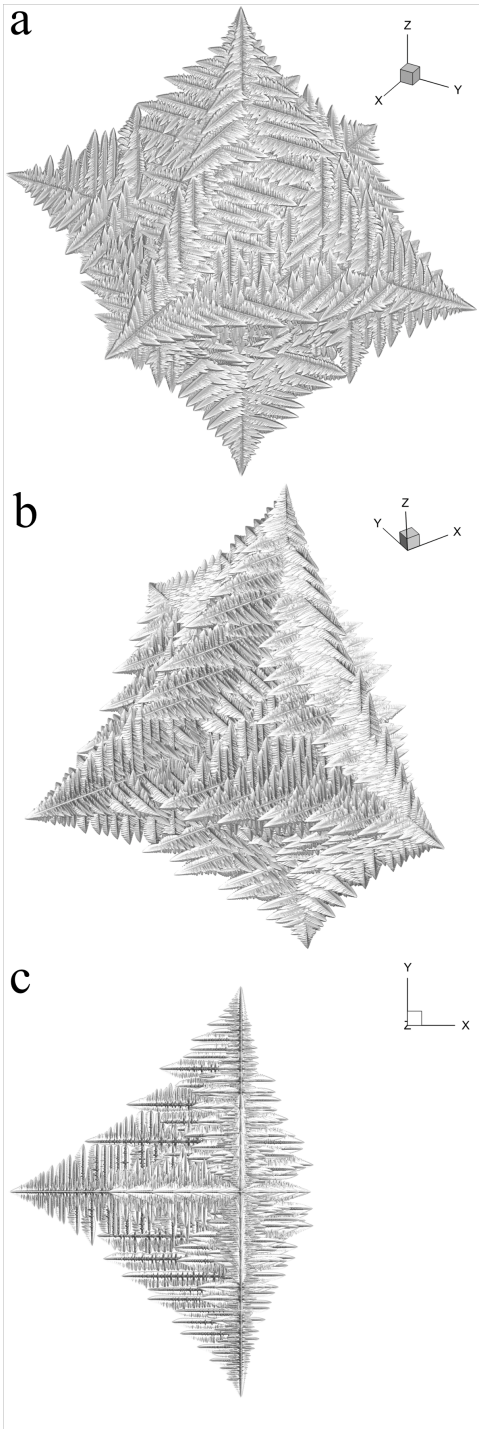


Figure 5.19: Free equiaxed growth [170]. a) No flow, b,c) forced convection of $u = 400$ µm/s in $+x$ viewed at different angles.

Table 5.4: Material properties of the Ga-In alloy used in simulations.

Density Ga, ρ_{Ga} (kg/m^3)	6095
Density In, ρ_{In} (kg/m^3)	7020
Kinematic viscosity, ν (m^2/s)	$3.28 \cdot 10^{-7}$
Partitioning coefficient, k_p	0.5
Solute diffusivity, D_C (m^2/s)	$2 \cdot 10^{-9}$
Liquidus slope, m_l (K/wt%)	2.9
Solute expansion coefficient, β_C (wt%$^{-1}$)	$1.66 \cdot 10^{-3}$
Thermal expansion coefficient, β (K^{-1})	$1.18 \cdot 10^{-4}$

Rossendorf) research team. The microstructure solidification of Ga-25%wt.In alloy in a DHC is simulated using a grid of $3072 \times 16 \times 3072 = 151$ million elements with a grid step size $\Delta x = 9.375$ µm. The physical size of the model, $28.8 \times 0.15 \times 28.8$ mm, fully matches the experimental sample size and directly represents the physical processes occurring in the cavity. Initially, 32 seeds with random crystallographic orientations are placed on the cold west wall. Before the initiation of the solidification and the accompanying processes, such as the solute ejection into the melt, the thermal buoyancy drives the fluid flow in a counter-clockwise direction. Once the dendritic growth begins and the lighter solute gets ejected travelling to the top of the sample, the solutal buoyancy overtakes the thermal buoyancy in the vicinity of the advancing solid/liquid interface. The solutal buoyancy generates a clockwise fluid flow motion competing with the thermally induced one. The result is a stable situation with two major co-existing circular motions that leads to a stratification of concentration as shown in Figure 5.20. The accumulated solute at the top of the sample suppresses the dendritic growth leading to a curved shape of the solid region. All the excess solute produced in the interdendritic region is carried by the flow to the west side where

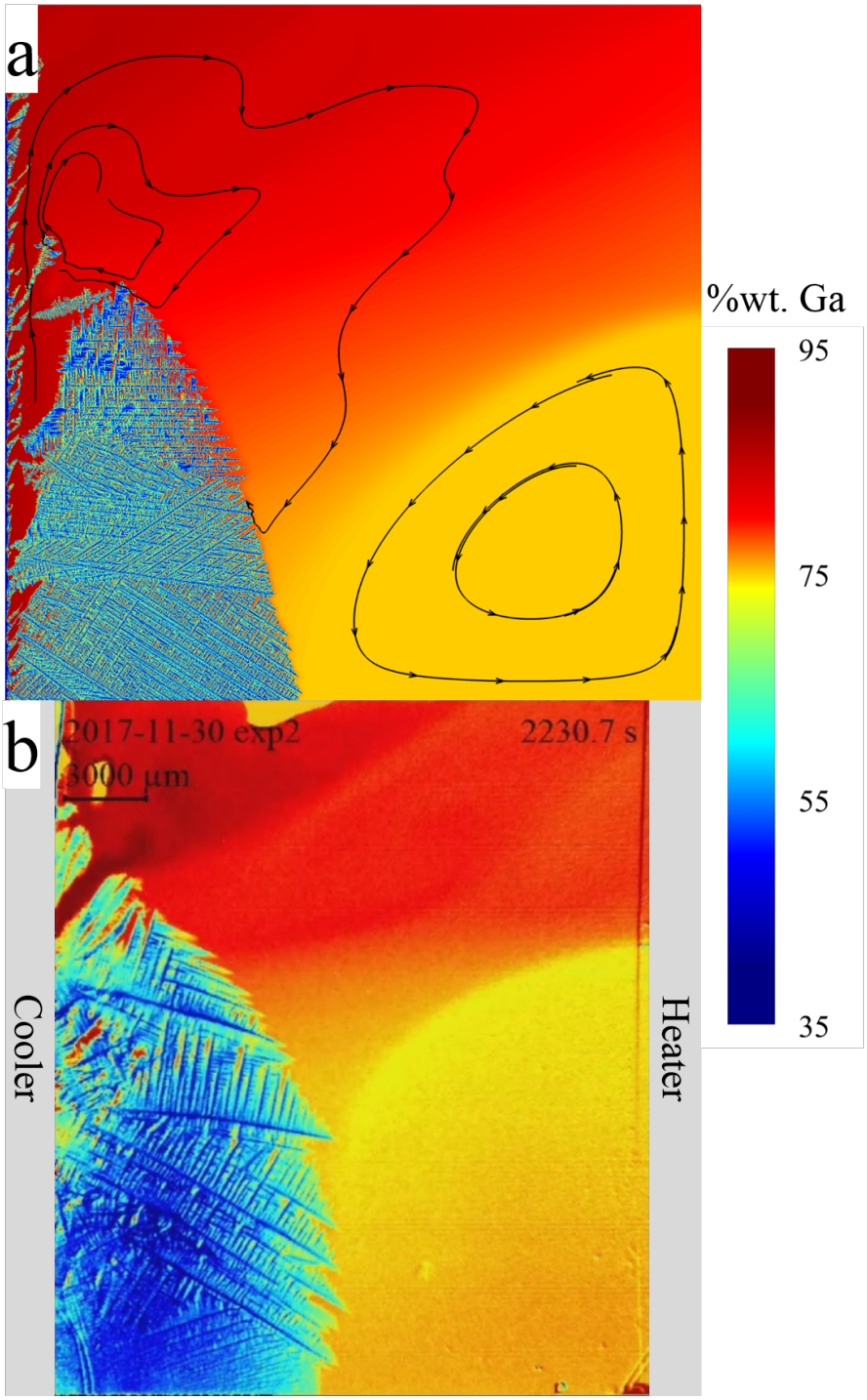


Figure 5.20: Solidification with a horizontal thermal gradient [170]. a) Numerical, b) experimental.

it escapes to the top through a chimney. It is more expressed in the numerical model than the experimental. One explanation might be that the chimney is there but it is not visible due to the placement of the heat sinks which limit the field of view. On the other half of the sample, the thermally driven flow delivers the bulk concentration to the interface promoting growth.

The numerical model closely reproduces the processes occurring in the experiment highlighting the capability of the coupled CA-LB system to capture microstructure solidification on a macroscale by modelling the whole sample and applying appropriate boundary conditions.

5.7.3 Channel formation in directional solidification

The CA-LB coupled system is applied to model the channel formation in directional solidification of Ga-25wt.%In alloy. The material properties are given in Table 5.4. The vertical thermal gradient is 1.6 K/mm and the lateral gradient is 0.25 K/mm, giving relatively lower temperatures in the middle of the domain compared to the sides. The growth speed is 4 µm/s. A numerical domain of $3200 \times 16 \times 3200 = 164$ million elements with a grid step size $\Delta x = 10$ µm and a time step $\Delta t = 5$ ms has been simulated for 10^6 time steps. It takes around 12 hours to simulate one million time steps when running the problem on 400 processors. The physical size of $32 \times 0.16 \times 32$ mm closely matches the size of the experimental setup [183], in fact it is slightly larger. The no-slip condition for velocity is used on the bottom wall and both sidewalls. A pressure outlet is used at the top boundary. The evolution process has been captured at three stages, as shown in Figure 5.21. The early stage after $3 \cdot 10^5$ time steps or 1500 seconds shows a fairly common sight of homogeneous solidification with the grain growth

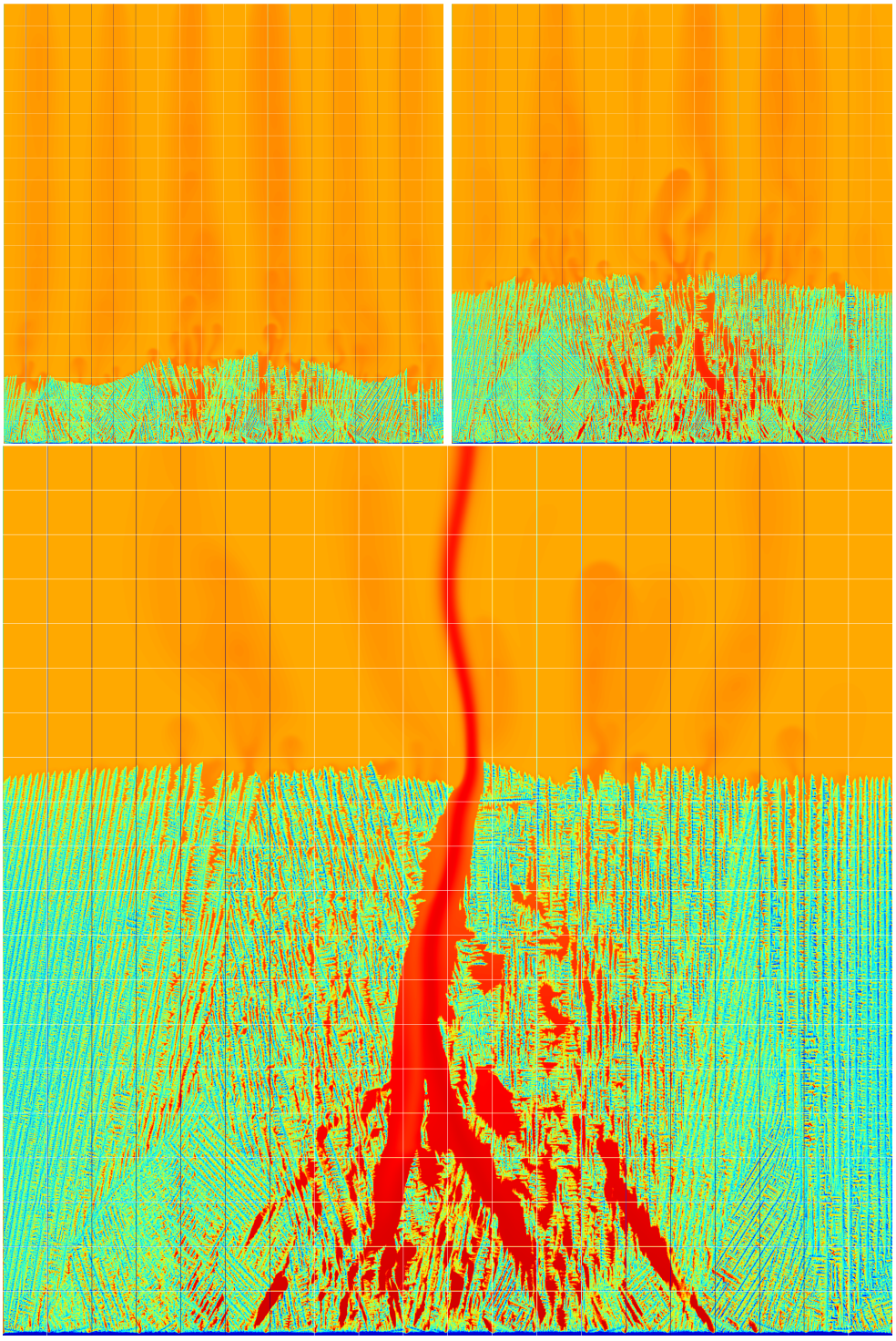


Figure 5.21: Evolution of the freckle formation in directional solidification. Top left: $t = 1500$ s, top right: $t = 3000$ s, bottom: $t = 5000$ s. [183]

and competition and Ga plume ejection due to natural convection. The intermediate stage at $6 \cdot 10^5$ time steps or 3000 seconds shows signs of the concentration build-up in the middle of the domain due to the presence of the lateral thermal gradient that introduces a bias in the equilibrium concentration profile. The high concentration causes remelting of the already solidified regions carving the way for the less dense solute to escape. The final stage is at 10^6 time steps or 5000 seconds after the start of the solidification. A single channel has formed in the middle of the domain driving the ejected Ga upwards. The material to sustain the channel is being drawn from the interdendritic region around it. This is in excellent agreement with the experimental results presented in [183].

5.8 Summary

The developed LBM has been validated on various simple benchmark cases in 2D and 3D testing the accuracy, stability and ability to couple the LBM to other solvers for modelling multi-physics problems. Firstly, the grid convergence studies have been conducted showing that the LBM together with the newly proposed 3D moment-based boundary method is second order accurate. This is an important result demonstrating the same order accuracy as the NSE. The stability analysis has shown that the moment-based method is generally independent of the relaxation time in contrast to the bounce-back rule. Moreover, the moment-based method can exactly recover the no-slip condition for velocity, which for the bounce-back rule is another disadvantage.

Secondly, for some physical problems, the LB code has been coupled to an external code, which handles either the heat transfer alone or the heat transfer

together with solidification to model, for example, the undercooled crystal growth. The steady-state or the time-dependent results of the coupled systems have been compared to the data in the literature in each case showing very good agreement.

The developed LB code has been fully coupled to the CA method and successfully applied to model large-scale multi-physics problems on the order of 100 million to 1 billion elements. A 3D undercooled crystal growth in forced convection has been modelled showing the expected behaviour of preferential and stunted growth. The large scale of the domain allows for the ternary and quaternary dendritic arms of the crystal to be observed in high detail.

The numerical study of the alloy solidification in a DHC has shown that the model closely reproduces the processes occurring in the experiment highlighting the capability of the coupled CA-LB system to capture microstructure solidification on a macroscale by modelling the whole sample and applying appropriate boundary conditions. The moment-based method has been applied to the flat domain boundaries and the bounce-back rule has been used to describe the complex interior boundaries.

The channel formation in directional solidification of Ga-25wt.%In alloy has been successfully modelled using the CA-LB method. The moment-based boundary method has been successfully used to describe the pressure and velocity boundaries of the domain demonstrating its capabilities. Matching the numerical domain size, conditions and material properties to those of the experiment has enabled to directly model the physical process in its entirety over several thousand seconds as it occurs on a macroscale.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

This work seeks to find the best numerical method to model 3D large-scale convection-driven fluid flow during microstructure solidification of metal alloys.

So, the first and main question to answer is:

What is the most appropriate and efficient way to model fluid flow during microstructure solidification on a macroscale?

The task is not straightforward as several requirements must be met. The flow solver is typically the most time-consuming part of the numerical algorithm. The method should span across multiple scales capturing microscopic flow in the inter-dendritic region as well as the macroscopic flow in the bulk melt. The inter-dendritic region has irregular and complex structures that act as obstacles

for the melt flow. Large-scale modelling implies the use of parallelisation and furthermore domain decomposition. The lattice Boltzmann method ticks all the boxes. It can easily describe micro-, meso- and macroscopic flows, handle complex boundaries and can be massively parallelised due to the local nature of its algorithm. The LBM has been tested on various benchmark problems assessing its suitability for the application. The assessment includes a performance analysis where it has been compared to different Navier-Stokes solvers and the GPU parallelisation potential has been investigated. It has been found that the LBM outperforms the Navier-Stokes solvers considered here several times. The reasons being the explicitness, structure and specification of the LBM compared to the other more general solvers. This and the easy coupling of the LBM and the solidification algorithm, where the effect of the temperature and solute change gets passed down to the fluid solver as thermal and solutal buoyancy forces. The GPU study has revealed that there is a potential of gaining additional speed-up by parallelising the code using GPUs, which would no longer see the flow solver as the bottleneck of the multi-physics algorithm, but the full implementation of CUDA LBM remains as a future work.

The second research question is more algorithm orientated:

Can the numerical technique be improved to produce more accurate or stable results?

While exploring the LBM it has been found that there is no simple and consistent way of implementing both velocity and pressure boundaries in 3D. Normally they are written using either the purely kinetic bounce-back schemes or hybrid

approach combining the hydrodynamic and bounce-back variation. However, as shown by the moment analysis the boundary methods using some variation of the bounce-back rule are arbitrarily setting constraints on the third order velocity moments that do not have a clear physical interpretation. A new purely hydrodynamic moment-based method for imposing boundary conditions in 3D has been proposed as a part of this thesis. It is an extension of the current 2D Moment Method, and it imposes pressure and velocity constraints directly onto the velocity moments. The developed LBM with the Moment Method has been validated against simple 2D and 3D flow benchmark cases showing an excellent agreement with the analytical and other model results. Furthermore, the proposed 3D Moment Method is shown to have second order accuracy, a larger relaxation time stability interval than the modified bounce-back scheme and an exact on-node recovery of the no-slip condition for velocity. Although it has been developed for the D3Q19 lattice, it can be extended to the D3Q15 and D3Q27 models, which is a task for the future.

The state-of-the-art 3D models either look at a single or a few dendrite growth with convection or many dendrite growth without convection. The developed method has enabled the research group to achieve 3D large-scale convection-driven fluid flow during microstructure solidification. The developed LBM flow solver forming the part of the parallel CA-LB microscale numerical model has opened the possibility to investigate defect formation in directional solidification and reveal fundamental mechanisms and stability of large-scale freckles. The work of this thesis has helped in gaining valuable knowledge that can further be fed to the automotive industry to improve, for example, the casting process and hence the durability of gas turbine blades.

6.2 Future work

The majority of time has been spent reviewing the literature, developing the LB model, which includes the derivation of the moment-based boundary method, coupling the model to external codes and validating it against benchmark cases. Having done all that as the result of this thesis, has opened the door to different possibilities that can be explored and directions that this work can be advanced. Though the research questions might have been answered, many more questions can be asked as the outcome of the thesis. They can be grouped in categories depending on the subject. Currently four different paths in which the work could be taken have been recognised: physics, accuracy, efficiency and performance and applications. These topics are explored next.

6.2.1 Physics

At the moment the obstacles in fluid flow are assumed to be fixed in space. It means that they cannot change their position as a consequence of a body force, such as sinking or floating due to gravity or being carried away with the shear flow. This is not entirely physical, but it is sufficient for most of the problems considered. So, the developed LB code could be improved by allowing to handle moving internal boundaries. It is not a straightforward process, but it has been demonstrated to work within the framework of solidification and the LBM [89].

Currently only hydrodynamics are handled by the LBM, but there is no reason why other physics cannot be introduced. In addition to the momentum equation, the heat and mass transport could also be calculated within the LBM by introducing extra distribution functions and potentially speeding up the calculations.

6.2.2 Accuracy

The accuracy of the results of the modelled problems depends on the numerical methods used. The LBM is second order accurate in time and space. The newly proposed 3D moment-based method has been shown to match the accuracy of the LBM. Although the bounce-back scheme that is used to handle the complex internal boundaries is efficient, easy to implement and reportedly second order accurate, it introduces some inaccuracies. The artificial slip velocity due to the inexact position of the boundary that depends on the relaxation time is considered a drawback. But because the merits outweigh its flaws, the method is still very popular. However, handling of the internal boundaries could be improved by employing, for example, the volumetric lattice Boltzmann method [75; 76; 184]. It fixes the slip velocity, is second order accurate and does not require spatial interpolation when dealing with arbitrarily curved stationary or moving boundaries. It could contribute greatly to improving the handling of the internal boundaries.

6.2.3 Efficiency and performance

The same implementation on different architectures can have different results performance-wise, as discovered in Section 4.5.3 where the parallel CUDA moment-based boundary subroutine does not experience reasonable speed-up. Rather than speeding up 10 or more times, the parallel GPU version is observed to be only 1.1 times faster than the serial CPU version. This problem can be fixed if addressed properly. The restructuring of the whole moment-based method subroutine might be required. Once that is done, the CUDA LB code can then be coupled to the CA method to model multi-physics problems in less time or achieving larger domains.

Another improvement can be made to increase efficiency – modifying the calls to the MPI libraries. Currently the whole 3D variable arrays are being passed down to the boundary update call, where only the surface area values get updated. The MPI boundary updates could be improved by extracting and passing only the 2D surface arrays. That would considerably improve the efficiency of the parallel code, especially the parallel GPU implementation using CUDA.

To reduce the inter-processor communications, the use of OpenMP can be assessed. It would work in tandem with MPI, where OpenMP would be responsible for distributing the tasks in each node and MPI would cover the inter-node communications. This is a common practice in high performance computing in the field of solidification modelling [82; 84].

Last but not least, reducing the number of the discrete velocities from D3Q19 to D3Q15 might be considered to minimise the memory usage and improve the efficiency. It has been shown that the D3Q15 model can be successfully applied to model dendritic growth with and without flow [91; 92; 94], but it is also known that the D3Q15 lattice lacks isotropy compared to D3Q19. It might be interesting to see the comparison between the two models to assess the suitability of using the D3Q15 lattice for the physical problems considered.

6.2.4 Applications

There are many applications where the developed LBM could be applied. One field that continues to become more popular is additive manufacturing (AM). In AM of metals, a laser interacts and travels across a powder bed, forming a molten, unstable melt pool. With the capabilities of the developed LBM, these complex flow dynamics can be captured and explored. Due to the modular form

of the developed LBM algorithm it has already been adopted in the research of AM. Other potential future applications include high undercooled growth and macroscopic fluid dynamics problems such as fire modelling.

Appendix A

PUBLICATIONS PRODUCED BY THIS RESEARCH

1. I. Krastins, A. Kao, and K. Pericleous, "Parallel GPU Lattice Boltzmann Method for Fluid Dynamics in Microstructure Modelling," Proceedings of SP17 6th Dec. Int. Conf. on Solidification Processing, pp. 342–345, 2017.

2. A. Kao, I. Krastins, M. Alexandrakis, N. Shevchenko, S. Eckert and K. Pericleous, "A Parallel Cellular Automata Lattice Boltzmann Method for Convection-Driven Solidification," *JOM*, 71 (1), pp. 48–58, 2019.

3. A. Kao, N. Shevchenko, M. Alexandrakis, I. Krastins, S. Eckert and K. Pericleous, "Thermal dependence of large-scale freckle defect formation," *Phil. Trans. R. Soc. A*, 377 (2143), p. 20180206, 2019.

4. I. Krastins, A. Kao, K. Pericleous and T. Reis, "3D Moment Method for the D3Q19 lattice Boltzmann equation," International Journal for Numerical Methods in Fluids, 2019. [SUBMITTED]

REFERENCES

[1] A. Kao and K. Pericleous, "A numerical model coupling thermoelectricity, magnetohydrodynamics and dendritic growth," *Journal of Algorithms & Computational Technology*, vol. 6, no. 1, pp. 173–201, 2012.

[2] S. Bennett, *A Lattice Boltzmann model for diffusion of binary gas mixtures*. PhD thesis, University of Cambridge, 2010.

[3] N. Noel, H. Jamgotchian, and B. Billia, "Influence of grain boundaries and natural convection on microstructure formation in cellular directional solidification of dilute succinonitrile alloys in a cylinder," *Journal of crystal growth*, vol. 187, no. 3-4, pp. 516–526, 1998.

[4] H. Jamgotchian, N. Bergeon, D. Benielli, P. Voge, B. Billia, and R. Guerin, "Localized microstructures induced by fluid flow in directional solidification," *Physical review letters*, vol. 87, no. 16, p. 166105, 2001.

[5] C. Lan and C. Tu, "Morphological instability due to double diffusive convection in directional solidification: the pit formation," *Journal of crystal growth*, vol. 220, no. 4, pp. 619–630, 2000.

[6] C. Lan, Y. Yang, H. Chen, and I. Lee, "Segregation and morphological

instability due to double-diffusive convection in rotational directional solidification," *Metallurgical and Materials Transactions A*, vol. 33, no. 9, pp. 3011–3017, 2002.

[7] C.-W. Lan, M.-H. Lee, M. Chuang, and C.-J. Shih, "Phase field modeling of convective and morphological instability during directional solidification of an alloy," *Journal of crystal growth*, vol. 295, no. 2, pp. 202–208, 2006.

[8] M. Burden and J. Hunt, "Some observations on primary dendrite spacings," *Metal Science*, vol. 10, no. 5, pp. 156–158, 1976.

[9] M. Glicksman and S. Huang, "Convective heat transfer during dendritic solidification," in *16th Aerospace Sciences Meeting*, p. 220, 1978.

[10] J. A. Dantzig and M. Rappaz, *Solidification*. EPFL press, 2009.

[11] N. Shevchenko, S. Boden, S. Eckert, and G. Gerbeth, "Observation of segregation freckle formation under the influence of melt convection," in *IOP Conference Series: Materials Science and Engineering*, vol. 27, p. 012085, IOP Publishing, 2012.

[12] N. Shevchenko, S. Eckert, S. Boden, and G. Gerbeth, "In situ X-ray monitoring of convection effects on segregation freckle formation," in *IOP Conference Series: Materials Science and Engineering*, vol. 33, p. 012035, IOP Publishing, 2012.

[13] J. Hong, D. Ma, J. Wang, F. Wang, B. Sun, A. Dong, F. Li, and A. Bührig-Polaczek, "Freckle defect formation near the casting interfaces of directionally solidified superalloys," *Materials*, vol. 9, no. 11, p. 929, 2016.

[14] S. Steinbach and L. Ratke, "The effect of rotating magnetic fields on the microstructure of directionally solidified Al–Si–Mg alloys," *Materials Science and Engineering: A*, vol. 413, pp. 200–204, 2005.

[15] M. Hainke, S. Steinbach, J. Dagner, L. Ratke, and G. Müller, "Solidification of AlSi alloys in the ARTEMIS and ARTEX facilities including rotating magnetic fields–A combined experimental and numerical analysis," in *Materials Science Forum*, vol. 508, pp. 199–204, Trans Tech Publ, 2006.

[16] L. Ratke, S. Steinbach, G. Müller, M. Hainke, A. Roósz, Y. Fautrelle, M. Dupouy, G. Zimmermann, A. Weiss, H.-J. Diepers, *et al.*, "MICAST–Microstructure Formation in Casting of technical alloys under diffusive and magnetically controlled convective conditions," in *Materials Science Forum*, vol. 508, pp. 131–144, Trans Tech Publ, 2006.

[17] N. Shevchenko, S. Boden, S. Eckert, D. Borin, M. Heinze, and S. Odenbach, "Application of X-ray radioscopic methods for characterization of two-phase phenomena and solidification processes in metallic melts," *The European Physical Journal Special Topics*, vol. 220, no. 1, pp. 63–77, 2013.

[18] N. Shevchenko, S. Boden, G. Gerbeth, and S. Eckert, "Chimney formation in solidifying Ga-25wt pct In alloys under the influence of thermosolutal melt convection," *Metallurgical and Materials Transactions A*, vol. 44, no. 8, pp. 3797–3808, 2013.

[19] N. Shevchenko, O. Roshchupkina, O. Sokolova, and S. Eckert, "The effect of natural and forced melt convection on dendritic solidification in Ga–In alloys," *Journal of Crystal Growth*, vol. 417, pp. 1–8, 2015.

[20] O. Roshchupkina, N. Shevchenko, and S. Eckert, "Observation of dendritic growth under the influence of forced convection," in *IOP Conference Series: Materials Science and Engineering*, vol. 84, p. 012080, IOP Publishing, 2015.

[21] H.-J. Diepers and I. Steinbach, "Interaction of interdendritic convection and dendritic primary spacing: phase-field simulation and analytical modeling," in *Materials Science Forum*, vol. 508, pp. 145–150, Trans Tech Publ, 2006.

[22] M. Zhu and D. Stefanescu, "Virtual front tracking model for the quantitative modeling of dendritic growth in solidification of alloys," *Acta Materialia*, vol. 55, no. 5, pp. 1741–1755, 2007.

[23] G. Schmitz, B. Böttger, J. Eiken, M. Apel, A. Viardin, A. Carré, and G. Laschet, "Phase-field based simulation of microstructure evolution in technical alloy grades," *International Journal of Advances in Engineering Sciences and Applied Mathematics*, vol. 2, no. 4, pp. 126–139, 2010.

[24] C. Beckermann, H.-J. Diepers, I. Steinbach, A. Karma, and X. Tong, "Modeling melt convection in phase-field simulations of solidification," *Journal of Computational Physics*, vol. 154, no. 2, pp. 468–496, 1999.

[25] X. Tong, C. Beckermann, A. Karma, and Q. Li, "Phase-field simulations of dendritic crystal growth in a forced flow," *Physical Review E*, vol. 63, no. 6, p. 061601, 2001.

[26] N. Al-Rawahi and G. Tryggvason, "Numerical simulation of dendritic solidification with convection: two-dimensional geometry," *Journal of Computational Physics*, vol. 180, no. 2, pp. 471–496, 2002.

[27] M. Zhu, S. Lee, and C. Hong, "Modified cellular automaton model for the prediction of dendritic growth with melt convection," *Physical Review E*, vol. 69, no. 6, p. 061610, 2004.

[28] P. Zhao, J. Heinrich, and D. Poirier, "Dendritic solidification of binary alloys with free and forced convection," *International journal for numerical methods in fluids*, vol. 49, no. 3, pp. 233–266, 2005.

[29] M. Zhu, D. Sun, S. Pan, Q. Zhang, and D. Raabe, "Modelling of dendritic growth during alloy solidification under natural convection," *Modelling and Simulation in Materials Science and Engineering*, vol. 22, no. 3, p. 034006, 2014.

[30] J.-H. Jeong, N. Goldenfeld, and J. A. Dantzig, "Phase field model for three-dimensional dendritic growth with fluid flow," *Physical Review E*, vol. 64, no. 4, p. 041602, 2001.

[31] Y. Lu, C. Beckermann, and A. Karma, "Convection effects in three-dimensional dendritic growth," in *ASME 2002 International Mechanical Engineering Congress and Exposition*, pp. 197–202, American Society of Mechanical Engineers, 2002.

[32] N. Al-Rawahi and G. Tryggvason, "Numerical simulation of dendritic solidification with convection: Three-dimensional flow," *Journal of Computational physics*, vol. 194, no. 2, pp. 677–696, 2004.

[33] Y. Lu, C. Beckermann, and J. Ramirez, "Three-dimensional phase-field simulations of the effect of convection on free dendritic growth," *Journal of crystal growth*, vol. 280, no. 1-2, pp. 320–334, 2005.

[34] L. Tan and N. Zabaras, "A level set simulation of dendritic solidification with combined features of front-tracking and fixed-domain methods," *Journal of Computational Physics*, vol. 211, no. 1, pp. 36–63, 2006.

[35] L. Tan and N. Zabaras, "A level set simulation of dendritic solidification of multi-component alloys," *Journal of Computational Physics*, vol. 221, no. 1, pp. 9–40, 2007.

[36] H. Dong and P. D. Lee, "Simulation of the columnar-to-equiaxed transition in directionally solidified Al–Cu alloys," *Acta Materialia*, vol. 53, no. 3, pp. 659–668, 2005.

[37] M. Wu and A. Ludwig, "Using a three-phase deterministic model for the columnar-to-equiaxed transition," *Metallurgical and Materials Transactions A*, vol. 38, no. 7, pp. 1465–1475, 2007.

[38] M. Wu, A. Fjeld, and A. Ludwig, "Modelling mixed columnar-equiaxed solidification with melt convection and grain sedimentation–Part I: Model description," *Computational Materials Science*, vol. 50, no. 1, pp. 32–42, 2010.

[39] M. Wu, A. Ludwig, and A. Fjeld, "Modelling mixed columnar-equiaxed solidification with melt convection and grain sedimentation–Part II: Illustrative modelling results and parameter studies," *Computational Materials Science*, vol. 50, no. 1, pp. 43–58, 2010.

[40] L. Yuan and P. D. Lee, "Dendritic solidification under natural and forced convection in binary alloys: 2D versus 3D simulation," *Modelling and sim-*

ulation in Materials Science and Engineering, vol. 18, no. 5, p. 055008, 2010.

[41] µMatIC Microstructural Simulation Software. `http://www.imperial.ac.uk/engineering-alloys/research/software/`. Accessed July 21, 2018.

[42] L. Yuan and P. D. Lee, "A new mechanism for freckle initiation based on microstructural level simulation," *Acta Materialia*, vol. 60, no. 12, pp. 4917–4926, 2012.

[43] S. Karagadde, L. Yuan, N. Shevchenko, S. Eckert, and P. Lee, "3-D microstructural model of freckle formation validated using in situ experiments," *Acta Materialia*, vol. 79, pp. 168–180, 2014.

[44] A. Kao, N. Shevchenko, O. Roshchupinka, S. Eckert, and K. Pericleous, "The effects of natural, forced and thermoelectric magnetohydrodynamic convection during the solidification of thin sample alloys," in *IOP Conference series: Materials science and Engineering*, vol. 84, p. 012018, IOP Publishing, 2015.

[45] A. Kao, K. Pericleous, M. Patel, and V. Voller, "Effects of magnetic fields on crystal growth," *International Journal of Cast Metals Research*, vol. 22, no. 1-4, pp. 147–150, 2009.

[46] A. Kao, G. Djambazov, K. Pericleous, and V. Voller, "Thermoelectric MHD in dendritic solidification," *Magnetohydrodynamics*, vol. 45, no. 3, pp. 305–315, 2009.

[47] A. Kao and K. Pericleous, "The effect of secondary arm growth on thermoelectric magnetohydrodynamics.," *Magnetohydrodynamics (0024-998X)*, vol. 48, no. 2, 2012.

[48] A. Kao, P. D. Lee, and K. Pericleous, "Influence of a slow rotating magnetic field in thermoelectric magnetohydrodynamic processing of alloys," *ISIJ international*, vol. 54, no. 6, pp. 1283–1287, 2014.

[49] A. Kao, "Analytic solutions to determine critical magnetic fields for thermoelectric magnetohydrodynamics in alloy solidification," *Metallurgical and Materials Transactions A*, vol. 46, no. 9, pp. 4215–4233, 2015.

[50] J. Gao, M. Han, A. Kao, K. Pericleous, D. V. Alexandrov, and P. K. Galenko, "Dendritic growth velocities in an undercooled melt of pure nickel under static magnetic fields: a test of theory with convection," *Acta Materialia*, vol. 103, pp. 184–191, 2016.

[51] A. Kao, B. Cai, P. Lee, and K. Pericleous, "The effects of Thermoelectric Magnetohydrodynamics in directional solidification under a transverse magnetic field," *Journal of Crystal Growth*, vol. 457, pp. 270–274, 2017.

[52] R. Zhao, J. Gao, A. Kao, and K. Pericleous, "Measurements and modelling of dendritic growth velocities of pure Fe with thermoelectric magnetohydrodynamics convection," *Journal of Crystal Growth*, vol. 475, pp. 354–361, 2017.

[53] A. Kao, J. Gao, and K. Pericleous, "Thermoelectric magnetohydrodynamic effects on the crystal growth rate of undercooled Ni dendrites," *Phil. Trans. R. Soc. A*, vol. 376, no. 2113, p. 20170206, 2018.

[54] A. Kao, *Thermoelectric magnetohydrodynamics in dendritic solidification*. PhD thesis, University of Greenwich, 2010.

[55] Y. Shin and C. Hong, "Modeling of dendritic growth with convection using a modified cellular automaton model with a diffuse interface," *ISIJ international*, vol. 42, no. 4, pp. 359–367, 2002.

[56] S. Chen and G. D. Doolen, "Lattice Boltzmann method for fluid flows," *Annual review of fluid mechanics*, vol. 30, no. 1, pp. 329–364, 1998.

[57] C. K. Aidun and J. R. Clausen, "Lattice-Boltzmann method for complex flows," *Annual review of fluid mechanics*, vol. 42, pp. 439–472, 2010.

[58] Q. Li, K. H. Luo, Q. Kang, Y. He, Q. Chen, and Q. Liu, "Lattice Boltzmann methods for multiphase flow and phase-change heat transfer," *Progress in Energy and Combustion Science*, vol. 52, pp. 62–105, 2016.

[59] W.-S. Jiaung, J.-R. Ho, and C.-P. Kuo, "Lattice Boltzmann method for the heat conduction problem with phase change," *Numerical Heat Transfer: Part B: Fundamentals*, vol. 39, no. 2, pp. 167–187, 2001.

[60] W. Miller, S. Succi, and D. Mansutti, "Lattice Boltzmann model for anisotropic liquid-solid phase transition," *Physical review letters*, vol. 86, no. 16, p. 3578, 2001.

[61] W. Miller and S. Succi, "A lattice Boltzmann model for anisotropic crystal growth from melt," *Journal of Statistical Physics*, vol. 107, no. 1-2, pp. 173–186, 2002.

[62] D. Chatterjee and S. Chakraborty, "An enthalpy-based lattice Boltzmann model for diffusion dominated solid–liquid phase transformation," *Physics Letters A*, vol. 341, no. 1-4, pp. 320–330, 2005.

[63] D. Chatterjee and S. Chakraborty, "A hybrid lattice Boltzmann model for solid–liquid phase transition in presence of fluid flow," *Physics Letters A*, vol. 351, no. 4-5, pp. 359–367, 2006.

[64] D. Medvedev and K. Kassner, "Lattice Boltzmann scheme for crystal growth in external flows," *Physical Review E*, vol. 72, no. 5, p. 056703, 2005.

[65] M. El Ganaoui, R. Bennacer, *et al.*, "Lattice Boltzmann method for melting/solidification problems," *Comptes Rendus Mécanique*, vol. 335, no. 5-6, pp. 295–303, 2007.

[66] E. Semma, M. El Ganaoui, R. Bennacer, and A. Mohamad, "Investigation of flows in solidification by using the lattice Boltzmann method," *International Journal of Thermal Sciences*, vol. 47, no. 3, pp. 201–208, 2008.

[67] D. Sun, M. Zhu, S. Pan, and D. Raabe, "Lattice Boltzmann modeling of dendritic growth in a forced melt convection," *Acta Materialia*, vol. 57, no. 6, pp. 1755–1767, 2009.

[68] D. Sun, M. Zhu, S. Pan, C. Yang, and D. Raabe, "Lattice Boltzmann modeling of dendritic growth in forced and natural convection," *Computers & Mathematics with Applications*, vol. 61, no. 12, pp. 3585–3592, 2011.

[69] D. Sun, Y. Wang, H. Yu, and Q. Han, "A lattice Boltzmann study on

dendritic growth of a binary alloy in the presence of melt convection," *International Journal of Heat and Mass Transfer*, vol. 123, pp. 213–226, 2018.

[70] H. Yin, S. Felicelli, and L. Wang, "Simulation of a dendritic microstructure with the lattice Boltzmann and cellular automaton methods," *Acta Materialia*, vol. 59, no. 8, pp. 3124–3136, 2011.

[71] F. Talati and M. Taghilou, "Lattice Boltzmann application on the PCM solidification within a rectangular finned container," *Applied Thermal Engineering*, vol. 83, pp. 108–120, 2015.

[72] R. Rojas, T. Takaki, and M. Ohno, "A phase-field-lattice Boltzmann method for modeling motion and growth of a dendrite for binary alloy solidification in the presence of melt convection," *Journal of Computational Physics*, vol. 298, pp. 29–40, 2015.

[73] D. Sun, M. Zhu, J. Wang, and B. Sun, "Lattice Boltzmann modeling of bubble formation and dendritic growth in solidification of binary alloys," *International Journal of Heat and Mass Transfer*, vol. 94, pp. 474–487, 2016.

[74] D. Sun, S. Pan, Q. Han, and B. Sun, "Numerical simulation of dendritic growth in directional solidification of binary alloys using a lattice Boltzmann scheme," *International Journal of Heat and Mass Transfer*, vol. 103, pp. 821–831, 2016.

[75] Q. Liu, Y.-L. He, and Q. Li, "Enthalpy-based multiple-relaxation-time lat-

tice Boltzmann method for solid-liquid phase-change heat transfer in metal foams," *Physical Review E*, vol. 96, no. 2, p. 023303, 2017.

[76] R. Huang and H. Wu, "Total enthalpy-based lattice Boltzmann method with adaptive mesh refinement for solid-liquid phase change," *Journal of Computational Physics*, vol. 315, pp. 65–83, 2016.

[77] M. Cross, S. Johnson, and P. Chow, "Mapping enthalpy-based solidification algorithms onto vector and parallel architectures," *Applied mathematical modelling*, vol. 13, no. 12, pp. 702–709, 1989.

[78] P. Chow, *Control volume unstructured mesh procedure for convection-diffusion solidification processes*. PhD thesis, PhD Thesis, University of Greenwich, 1993.

[79] K. McManus, A. Williams, M. Cross, N. Croft, and C. Walshaw, "Assessing the scalability of multiphysics tools for modeling solidification and melting processes on parallel clusters," *The International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 1–27, 2005.

[80] M. Cross, P. Chow, C. Bailey, N. Croft, J. Ewer, P. Leggett, K. McManus, K. Pericleous, and M. Patel, "PHYSICA-a software environment for the modelling of multi-physics phenomena," *ZAMM-Zeitschrift fur Angewandte Mathematik und Mechanik*, vol. 76, no. 4, pp. 105–108, 1996.

[81] W. L. George and J. A. Warren, "A parallel 3D dendritic growth simulator using the phase-field method," *Journal of Computational Physics*, vol. 177, no. 2, pp. 264–283, 2002.

[82] B. Nestler, "A 3D parallel simulator for crystal growth and solidification in complex alloy systems," *Journal of Crystal Growth*, vol. 275, no. 1-2, pp. e273–e278, 2005.

[83] K. Wang, A. Chang, L. V. Kale, and J. A. Dantzig, "Parallelization of a level set method for simulating dendritic growth," *Journal of Parallel and Distributed Computing*, vol. 66, no. 11, pp. 1379–1386, 2006.

[84] Z. Guo, J. Mi, and P. Grant, "An implicit parallel multigrid computing scheme to solve coupled thermal-solute phase-field equations for dendrite evolution," *Journal of Computational Physics*, vol. 231, no. 4, pp. 1781–1796, 2012.

[85] NVIDIA CUDA Parallel Computing. `https://www.nvidia.co.uk/object/cuda-parallel-computing-uk.html`. Accessed August 26, 2018.

[86] TOP 500 The List. `https://www.top500.org`. Accessed August 26, 2018.

[87] T. Shimokawabe, T. Aoki, T. Takaki, T. Endo, A. Yamanaka, N. Maruyama, A. Nukada, and S. Matsuoka, "Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 3, ACM, 2011.

[88] T. Takaki, T. Shimokawabe, M. Ohno, A. Yamanaka, and T. Aoki, "Unexpected selection of growing dendrites by very-large-scale phase-field simulation," *Journal of Crystal Growth*, vol. 382, pp. 21–25, 2013.

[89] T. Takaki, R. Rojas, M. Ohno, T. Shimokawabe, and T. Aoki, "GPU phase-field lattice Boltzmann simulations of growth and motion of a binary alloy dendrite," in *IOP Conference Series: Materials Science and Engineering*, vol. 84, p. 012066, IOP Publishing, 2015.

[90] S. Sakane, T. Takaki, R. Rojas, M. Ohno, Y. Shibuta, T. Shimokawabe, and T. Aoki, "Multi-GPUs parallel computation of dendrite growth in forced convection using the phase-field-lattice Boltzmann model," *Journal of Crystal Growth*, vol. 474, pp. 154–159, 2017.

[91] M. Eshraghi, S. D. Felicelli, and B. Jelinek, "Three dimensional simulation of solutal dendrite growth using lattice Boltzmann and cellular automaton methods," *Journal of Crystal growth*, vol. 354, no. 1, pp. 129–134, 2012.

[92] M. Eshraghi, B. Jelinek, and S. D. Felicelli, "Large-scale three-dimensional simulation of dendritic solidification using lattice Boltzmann method," *JOM*, vol. 67, no. 8, pp. 1786–1792, 2015.

[93] B. Jelinek, M. Eshraghi, S. Felicelli, and J. F. Peters, "Large-scale parallel lattice Boltzmann–cellular automaton model of two-dimensional dendritic growth," *Computer Physics Communications*, vol. 185, no. 3, pp. 939–947, 2014.

[94] M. Eshraghi, M. Hashemi, B. Jelinek, and S. D. Felicelli, "Three-dimensional lattice Boltzmann modeling of dendritic solidification under forced and natural convection," *Metals*, vol. 7, no. 11, p. 474, 2017.

[95] M. Alexandrakis, *Parallelised Micro-Macroscale Modelling of Convection*

Driven Freckles in Binary Alloys. PhD thesis, University of Greenwich, 2017.

[96] P. Lee, R. Atwood, R. Dashwood, and H. Nagaumi, "Modeling of porosity formation in direct chill cast aluminum–magnesium alloys," *Materials Science and Engineering: A*, vol. 328, no. 1-2, pp. 213–222, 2002.

[97] P. D. Lee, A. Chirazi, R. Atwood, and W. Wang, "Multiscale modelling of solidification microstructures, including microsegregation and microporosity, in an Al–Si–Cu alloy," *Materials Science and Engineering: A*, vol. 365, no. 1-2, pp. 57–65, 2004.

[98] W. Wang, P. D. Lee, and M. Mclean, "A model of solidification microstructures in nickel-based superalloys: predicting primary dendrite spacing selection," *Acta materialia*, vol. 51, no. 10, pp. 2971–2987, 2003.

[99] B. M. Boghosian, "Lattice gases and cellular automata," *Future Generation Computer Systems*, vol. 16, no. 2-3, pp. 171–185, 1999.

[100] D. A. Wolf-Gladrow, *Lattice-gas cellular automata and lattice Boltzmann models: an introduction*. Springer, 2004.

[101] Z. Guo and C. Shu, *Lattice Boltzmann method and its applications in engineering*, vol. 3. World Scientific, 2013.

[102] L. P. Kadanoff and J. Swift, "Transport coefficients near the liquid-gas critical point," *Physical Review*, vol. 166, no. 1, p. 89, 1968.

[103] J. Hardy, Y. Pomeau, and O. De Pazzis, "Time evolution of a two-dimensional model system. I. Invariant states and time correlation func-

tions," *Journal of Mathematical Physics*, vol. 14, no. 12, pp. 1746–1759, 1973.

[104] U. Frisch, B. Hasslacher, and Y. Pomeau, "Lattice-gas automata for the Navier-Stokes equation," *Physical review letters*, vol. 56, no. 14, p. 1505, 1986.

[105] S. Wolfram, "Cellular automaton fluids 1: Basic theory," *Journal of statistical physics*, vol. 45, no. 3-4, pp. 471–526, 1986.

[106] U. Frisch, D. d'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet, "Lattice gas hydrodynamics in two and three dimensions," tech. rep., Los Alamos National Lab., NM (USA); Observatoire de Nice, 06 (France); Ecole Normale Superieure, 75-Paris (France), 1986.

[107] D. d'Humieres, P. Lallemand, and U. Frisch, "Lattice gas models for 3D hydrodynamics," *EPL (Europhysics Letters)*, vol. 2, no. 4, p. 291, 1986.

[108] G. R. McNamara and G. Zanetti, "Use of the Boltzmann equation to simulate lattice-gas automata," *Physical review letters*, vol. 61, no. 20, p. 2332, 1988.

[109] F. J. Higuera and J. Jimenez, "Boltzmann approach to lattice gas simulations," *EPL (Europhysics Letters)*, vol. 9, no. 7, p. 663, 1989.

[110] Y. H. Qian, "Lattice gas and lattice kinetic theory applied to the Navier-Stokes equations," *Doktorarbeit, Universite Pierre et Marie Curie, Paris*, 1990.

[111] S. Chen, H. Chen, D. Martnez, and W. Matthaeus, "Lattice Boltzmann model for simulation of magnetohydrodynamics," *Physical Review Letters*, vol. 67, no. 27, p. 3776, 1991.

[112] J. Koelman, "A simple lattice Boltzmann scheme for Navier-Stokes fluid flow," *EPL (Europhysics Letters)*, vol. 15, no. 6, p. 603, 1991.

[113] Y. Qian, D. d'Humières, and P. Lallemand, "Lattice BGK models for Navier-Stokes equation," *EPL (Europhysics Letters)*, vol. 17, no. 6, p. 479, 1992.

[114] P. L. Bhatnagar, E. P. Gross, and M. Krook, "A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems," *Physical review*, vol. 94, no. 3, p. 511, 1954.

[115] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, "The lattice Boltzmann method," *Springer International Publishing*, vol. 10, pp. 978–3, 2017.

[116] S. Succi, *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford university press, 2001.

[117] J. Latt, *Hydrodynamic limit of lattice Boltzmann equations*. PhD thesis, University of Geneva, 2007.

[118] W. Miller, "Flow in the driven cavity calculated by the lattice Boltzmann method," *Physical Review E*, vol. 51, no. 4, p. 3659, 1995.

[119] R. Mei, W. Shyy, D. Yu, and L.-S. Luo, "Lattice Boltzmann method for 3-D

flows with curved boundary," *Journal of Computational Physics*, vol. 161, no. 2, pp. 680–699, 2000.

[120] D. Kandhai, A. Koponen, A. Hoekstra, M. Kataja, J. Timonen, and P. Sloot, "Implementation aspects of 3D lattice-BGK: boundaries, accuracy, and a new fast relaxation method," *Journal of Computational Physics*, vol. 150, no. 2, pp. 482–501, 1999.

[121] P. Lallemand and L.-S. Luo, "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability," *Physical Review E*, vol. 61, no. 6, p. 6546, 2000.

[122] K. Suga, Y. Kuwata, K. Takashima, and R. Chikasue, "A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows," *Computers & Mathematics with Applications*, vol. 69, no. 6, pp. 518–529, 2015.

[123] J. G. Zhou, "Rectangular lattice Boltzmann method," *Physical Review E*, vol. 81, no. 2, p. 026705, 2010.

[124] I. Ginzburg, D. dâĂŹHumières, and A. Kuzmin, "Optimal stability of advection-diffusion lattice Boltzmann models with two relaxation times for positive/negative equilibrium," *Journal of Statistical Physics*, vol. 139, no. 6, pp. 1090–1143, 2010.

[125] X. Niu, C. Shu, Y. Chew, and T. Wang, "Investigation of stability and hydrodynamics of different lattice Boltzmann models," *Journal of statistical physics*, vol. 117, no. 3-4, pp. 665–680, 2004.

[126] I. Ginzburg, "Truncation errors, exact and heuristic stability analysis of

two-relaxation-times lattice Boltzmann schemes for anisotropic advection-diffusion equation," *Communications in Computational Physics*, vol. 11, no. 5, pp. 1439–1502, 2012.

[127] G. Silva and V. Semiao, "Truncation errors and the rotational invariance of three-dimensional lattice models in the lattice Boltzmann method," *Journal of Computational Physics*, vol. 269, pp. 259–279, 2014.

[128] I. Ginzburg, F. Verhaeghe, and D. d'Humieres, "Two-relaxation-time lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions," *Communications in computational physics*, vol. 3, no. 2, pp. 427–478, 2008.

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

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

[131] Q. Zou and X. He, "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model," *Physics of fluids*, vol. 9, no. 6, pp. 1591–1598, 1997.

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

[133] M. E. Kutay, A. H. Aydilek, and E. Masad, "Laboratory validation of lattice Boltzmann method for modeling pore-scale flow in granular materials," *Computers and Geotechnics*, vol. 33, no. 8, pp. 381–395, 2006.

[134] M. Hecht and J. Harting, "Implementation of on-site velocity boundary conditions for D3Q19 lattice Boltzmann simulations," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no. 01, p. P01018, 2010.

[135] K. Mattila, J. Hyväluoma, and T. Rossi, "Mass-flux-based outlet boundary conditions for the lattice Boltzmann method," *Journal of Statistical Mechanics: theory and experiment*, vol. 2009, no. 06, p. P06015, 2009.

[136] T. Reis and P. J. Dellar, "Lattice Boltzmann simulations of pressure-driven flows in microchannels using Navier–Maxwell slip boundary conditions," *Physics of Fluids*, vol. 24, no. 11, p. 112001, 2012.

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

[138] D. R. Noble, J. G. Georgiadis, and R. O. Buckius, "Comparison of accuracy and performance for lattice Boltzmann and finite difference simulations of steady viscous flow," *International Journal for Numerical Methods in Fluids*, vol. 23, no. 1, pp. 1–18, 1996.

[139] S. Bennett, P. Asinari, and P. J. Dellar, "A lattice Boltzmann model for diffusion of binary gas mixtures that includes diffusion slip," *International journal for numerical methods in fluids*, vol. 69, no. 1, pp. 171–189, 2012.

[140] A. Hantsch, T. Reis, and U. Gross, "Moment method boundary conditions for multiphase lattice Boltzmann simulations with partially-wetted walls," *The Journal of Computational Multiphase Flows*, vol. 7, no. 1, pp. 1–14, 2015.

[141] R. Allen and T. Reis, "Moment-based boundary conditions for lattice Boltzmann simulations of natural convection in cavities," *Progress in Computational Fluid Dynamics, An International Journal (PFCD)*, vol. 16, no. 4, p. 216, 2016.

[142] S. Mohammed, D. Graham, and T. Reis, "Assessing moment-based boundary conditions for the lattice Boltzmann equation: A study of dipole-wall collisions," *Computers & Fluids*, vol. 176, pp. 79–96, 2018.

[143] S. Mohammed and T. Reis, "Using the lid-driven cavity flow to validate moment-based boundary conditions for the Lattice Boltzmann Equation," *Archive of Mechanical Engineering*, vol. 64, no. 1, pp. 57–74, 2017.

[144] F. Verhaeghe, L.-S. Luo, and B. Blanpain, "Lattice Boltzmann modeling of microchannel flow in slip flow regime," *Journal of Computational Physics*, vol. 228, no. 1, pp. 147–157, 2009.

[145] D. J. Holdych, D. R. Noble, J. G. Georgiadis, and R. O. Buckius, "Truncation error analysis of lattice Boltzmann methods," *Journal of Computational Physics*, vol. 193, no. 2, pp. 595–619, 2004.

[146] I. Ginzburg, "Lattice Boltzmann modeling with discontinuous collision components: Hydrodynamic and advection-diffusion equations," *Journal of Statistical Physics*, vol. 126, no. 1, pp. 157–206, 2007.

[147] D. d'Humieres, "Generalized lattice-Boltzmann equations," *Rarefied gas dynamics*, 1992.

[148] M. Geier, A. Greiner, and J. G. Korvink, "Cascaded digital lattice Boltzmann automata for high Reynolds number flow," *Physical Review E*, vol. 73, no. 6, p. 066705, 2006.

[149] L. Fei, K. H. Luo, C. Lin, and Q. Li, "Modeling incompressible thermal flows using a central-moments-based lattice Boltzmann method," *International Journal of Heat and Mass Transfer*, vol. 120, pp. 624–634, 2018.

[150] L. Fei and K. H. Luo, "Consistent forcing scheme in the cascaded lattice Boltzmann method," *Physical Review E*, vol. 96, no. 5, p. 053307, 2017.

[151] D. Lycett-Brown, K. H. Luo, R. Liu, and P. Lv, "Binary droplet collision simulations by a multiphase cascaded lattice Boltzmann method," *Physics of Fluids*, vol. 26, no. 2, p. 023303, 2014.

[152] A. De Rosis, "A central moments-based lattice Boltzmann scheme for shallow water equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 319, pp. 379–392, 2017.

[153] D. d'Humières, "Multiple–relaxation–time lattice Boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002.

[154] L.-S. Luo, "Analytic solutions of linearized lattice Boltzmann equation for

simple flows," *Journal of statistical physics*, vol. 88, no. 3-4, pp. 913–926, 1997.

[155] L.-S. Luo, "Theory of the lattice Boltzmann method: Lattice Boltzmann models for nonideal gases," *Physical Review E*, vol. 62, no. 4, p. 4982, 2000.

[156] Z. Guo, C. Zheng, and B. Shi, "Discrete lattice effects on the forcing term in the lattice Boltzmann method," *Physical Review E*, vol. 65, no. 4, p. 046308, 2002.

[157] X. He, X. Shan, and G. D. Doolen, "Discrete Boltzmann equation model for nonideal gases," *Physical Review E*, vol. 57, no. 1, p. R13, 1998.

[158] L.-S. Luo, "Unified theory of lattice Boltzmann models for nonideal gases," *Physical review letters*, vol. 81, no. 8, p. 1618, 1998.

[159] MPI. `https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf`. Accessed August 25, 2018.

[160] OpenCL: The open standard for parallel programming of heterogeneous systems. `https://www.khronos.org/opencl/`. Accessed January 13, 2019.

[161] OpenCL: NVIDIA ACCELERATED COMPUTING. `https://developer.nvidia.com/opencl`. Accessed January 18, 2019.

[162] K. Karimi, N. G. Dickson, and F. Hamze, "A performance comparison of CUDA and OpenCL," *arXiv preprint arXiv:1005.2581*, 2010.

[163] J. Fang, A. L. Varbanescu, and H. Sips, "A comprehensive performance comparison of CUDA and OpenCL," in *Parallel Processing (ICPP), 2011 International Conference on*, pp. 216–225, IEEE, 2011.

[164] D. Demidov, K. Ahnert, K. Rupp, and P. Gottschling, "Programming CUDA and OpenCL: A case study using modern C++ libraries," *SIAM Journal on Scientific Computing*, vol. 35, no. 5, pp. C453–C472, 2013.

[165] Fortran Interface to OpenCL. `http://www.cass-hpc.com/solutions/libraries/clfortran-pure-fortran-interface-to-opencl/`. Accessed January 18, 2019.

[166] OpenACC: More Science Less Programming. `https://developer.nvidia.com/openacc/`. Accessed August 26, 2018.

[167] The OpenMP API specification for parallel programming. `https://www.openmp.org/`. Accessed August 26, 2018.

[168] G. Ruetsch and M. Fatica, *CUDA Fortran for scientists and engineers: best practices for efficient CUDA Fortran programming*. Elsevier, 2013.

[169] CUDA C PROGRAMMING GUIDE, August 2018. `https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf`. Accessed August 27, 2018.

[170] A. Kao, I. Krastins, M. Alexandrakis, N. Shevchenko, S. Eckert, and K. Pericleous, "A parallel cellular automata lattice Boltzmann method for convection-driven solidification," *JOM*, vol. 71, no. 1, pp. 48–58, 2019.

[171] V. Voller, "An enthalpy method for modeling dendritic growth in a binary alloy," *International Journal of Heat and Mass Transfer*, vol. 51, no. 3-4, pp. 823–834, 2008.

[172] COMSOL Multiphysics. `https://www.comsol.com/products`. Accessed October 18, 2018.

[173] ANSYS Fluent Software. `https://www.ansys.com/products/fluids/ansys-fluent`. Accessed October 18, 2018.

[174] PHOENICS. `http://www.cham.co.uk/phoenics.php`. Accessed October 18, 2018.

[175] U. Ghia, K. N. Ghia, and C. Shin, "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method," *Journal of computational physics*, vol. 48, no. 3, pp. 387–411, 1982.

[176] O. Botella and R. Peyret, "Benchmark spectral results on the lid-driven cavity flow," *Computers & Fluids*, vol. 27, no. 4, pp. 421–433, 1998.

[177] C. H. Williamson, "Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers," *Journal of Fluid Mechanics*, vol. 206, pp. 579–627, 1989.

[178] O. Posdziech and R. Grundmann, "A systematic approach to the numerical calculation of fundamental quantities of the two-dimensional flow over a circular cylinder," *Journal of Fluids and Structures*, vol. 23, no. 3, pp. 479–499, 2007.

[179] S. Timoshenko, *Theory of elasticity* . New York ; London : McGraw-Hill book company, inc, 1st ed., 1934.

[180] G. de Vahl Davis, "Natural convection of air in a square cavity: a bench

mark numerical solution," *International Journal for numerical methods in fluids*, vol. 3, no. 3, pp. 249–264, 1983.

[181] N. C. Markatos and K. Pericleous, "Laminar and turbulent natural convection in an enclosed cavity," *International Journal of Heat and Mass Transfer*, vol. 27, no. 5, pp. 755–772, 1984.

[182] V. R. Voller and C. Prakash, "A fixed grid numerical modelling methodology for convection-diffusion mushy region phase-change problems," *International Journal of Heat and Mass Transfer*, vol. 30, no. 8, pp. 1709–1719, 1987.

[183] A. Kao, N. Shevchenko, M. Alexandrakis, I. Krastins, S. Eckert, and K. Pericleous, "Thermal dependence of large-scale freckle defect formation," *Phil. Trans. R. Soc. A*, vol. 377, no. 2143, p. 20180206, 2019.

[184] Z. Wang, Y. Zhao, A. P. Sawchuck, M. C. Dalsing, and H. W. Yu, "GPU acceleration of volumetric lattice Boltzmann method for patient-specific computational hemodynamics," *Computers & Fluids*, vol. 115, pp. 192–200, 2015.