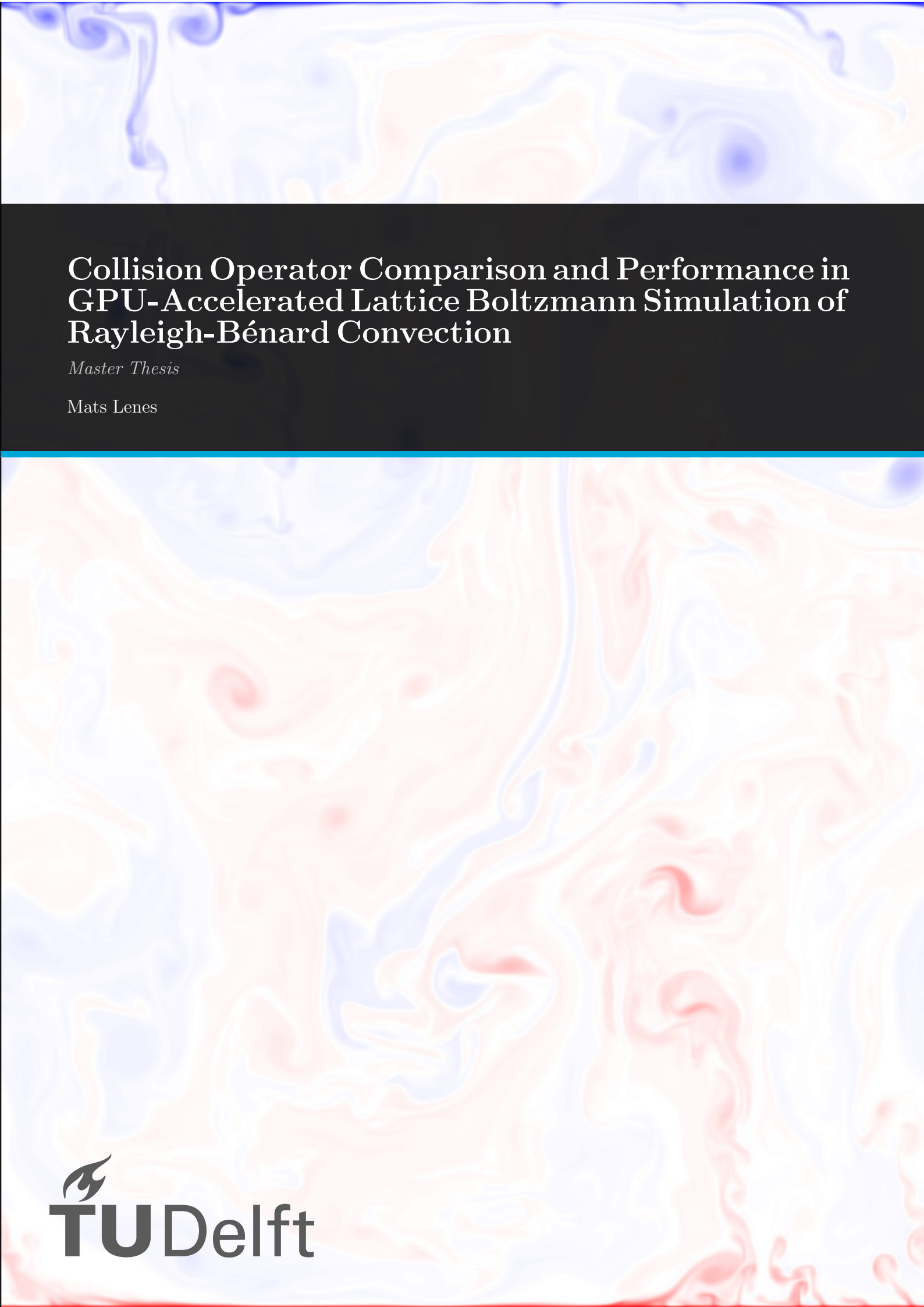




Collision Operator Comparison and Performance in GPU-Accelerated Lattice Boltzmann Simulation of Rayleigh-Bénard Convection

Master Thesis

Mats Lenes

Collision Operator Equivalence and Performance in GPU-Accelerated Lattice Boltzmann Simulation of Rayleigh-Bénard Convection

Master Thesis

by

Mats Lenes

to obtain the degree of Master of Science at the Delft University of Technology,
to be defended publicly on Thursday July 16, 2026 at 09:00

Student number: 5182360

Project duration: December 18 2025 – July 16, 2026

Thesis committee:	Dr. Ir. M. Rohde,	TU Delft, Supervisor
	Dr. Ir. D. Lathouwers,	TU Delft
	Dr. Ir. R.M. Hartkamp,	TU Delft



Highlights

The highlights of this thesis include:

- Mathematical equivalence proof between the filter matrix-, matrix projected Boltzmann-, and Multiple Relaxation Times methods.
- Highly optimized GPU implementation of lattice Boltzmann methods using kernel fusing, memory coalescence, and avoiding global memory traffic.
- Comparison of accuracy and stability for lattice Boltzmann methods in simulating 2D Rayleigh-Bénard convection at Rayleigh numbers from 10^3 to 10^{10} .
- Effect of variations in Prandtl number from 1 to 35 at a Rayleigh number of 10^7 on Nusselt number in 2D Rayleigh-Bénard convection using the Cumulant lattice Boltzmann method.

Abstract

The lattice Boltzmann method is a technique for simulating fluids offering high-performance codes by utilizing parallel computation. To simulate the conditions found in molten salt reactors, it is important for the simulation methods to allow for low viscosity or high thermal diffusivity. In this specific regime, the standard "Single Relaxation Time" (SRT) lattice Boltzmann method becomes unstable. Over the past decades, new lattice Boltzmann methods have been presented promising higher stability and/or accuracy by altering the collision operator at the core of the lattice Boltzmann algorithm. In this thesis, six lattice Boltzmann methods are compared on isothermal and thermal flows with the goal of selecting the best method to simulate thermal flow at high Prandtl numbers, or low viscosities. The methods investigated in this thesis are the Single Relaxation Time, Multiple Relaxation Times (MRT), filter matrix, matrix projected Boltzmann-, Entropic-, and Cumulant method. These methods are implemented in highly efficient GPU code using the Julia programming language achieving approximately 90% of the theoretical memory bandwidth. In this thesis, it is proven for the first time that the filter matrix and multiple relaxation time methods are algorithmically equivalent, with the only differences coming from a selection of higher order relaxation parameters, and the treatment of body forces. The Entropic and Cumulant method are shown to achieve the highest stability, with the Cumulant method also demonstrating the highest accuracy. The Cumulant method is therefore selected to simulate Rayleigh-Bénard convection to find the Nusselt number as a function of the Prandtl number over a range of Prandtl numbers from 1 to 35 at a Rayleigh number of 10^7 . In this regime, no clear influence of the Prandtl number is found, with all simulations achieving a Nusselt number of 14.2 ± 0.4 . This suggests that heat transfer in 2D Rayleigh-Bénard convection is robust across the Prandtl number range relevant to molten salt reactors.

Nomenclature

Latin symbols

c	Lattice speed
c_s	Lattice speed of sound ($c_s = c/\sqrt{3}$ for D2Q9)
f_i	Distribution function in direction i
f_i^{eq}	Equilibrium distribution function
N_x	Amount of lattice nodes in x-direction
N_y	Amount of lattice nodes in y-direction
$\vec{x}$	Position
$\vec{c}_i$	Weighted discrete microscopic lattice velocity
g_i	Thermal distribution function in direction i
g_i^{eq}	Equilibrium thermal distribution function
m_k	Moment k of distributions
m_k^{eq}	Equilibrium moment k of distributions
C_{ab}	Cumulant ab of distributions
C_{ab}^{eq}	Equilibrium cumulant ab of distributions
$T_{1,2}, F$	Higher order moments
R	Gas constant
D	Spatial dimensions
L	Domain length
H	Domain height
$\mathcal{H}^{(n)}$	n^{th} order Hermite polynomial
σ	Hermite polynomial weighting function
Nu	Nusselt number
p	Pressure
Ra	Rayleigh number
Re	Reynolds number
Ma	Mach number
Pr	Prandtl number
T	Temperature
T_h	Hot wall temperature
T_c	Cold wall temperature
ΔT	Temperature difference ($T_h - T_c$)
$\vec{g}$	Acceleration due to gravity
F_α	Body force in spatial direction α
F_i	Force distribution in distribution direction i
t	Time
u_x, u_y	Fluid velocity components
U	Reference velocity
$\vec{u}$	Fluid velocity vector
M_{ki}	Transformation matrix to moment space
w_i	Lattice weights
H	Entropic Lyapunov functional

Greek symbols

$\vec{\xi}_i$	Discrete microscopic lattice velocity
α	Entropic stabilizer
β	Thermal expansion coefficient
κ	Thermal diffusivity
ν	Kinematic viscosity
ρ	Fluid density
$\bar{\omega}$	Uncorrected momentum relaxation rate
ω	Momentum relaxation rate
ω_T	Thermal relaxation rate
$\gamma_{1,2,3}$	Higher order momentum relaxation rates
$\gamma_{T,1,2,3}$	Higher order thermal relaxation rate
Ω	Collision operator
ϵ	Smallness/dummy parameter
ν_B	Bulk viscosity

Subscripts & superscripts

$(\cdot)^{eq}$	Equilibrium quantity
$(\cdot)_i$	Discrete velocity direction index
$(\cdot)^*$	Post-collision quantity
$(\cdot)_\alpha$	Spatial index

Abbreviations

NSE	Navier–Stokes equations
ADE	Advection–Diffusion equation
BGK	Bhatnagar–Gross–Krook
FMLB	Filter Matrix Lattice Boltzmann
MPB	Matrix projected Boltzmann
LBM	Lattice Boltzmann Method
MLUPS	Million Lattice Updates Per Second
RB	Rayleigh–Bénard
SRT	Single Relaxation Time
MRT	Multiple Relaxation Times
GPU	Graphics Processing Unit
CPU	Central Processing Unit
SM	Streaming multiprocessor
CFD	Computational fluid dynamics

Contents

1	Introduction	1
1.1	Computational Fluid Dynamics	1
1.2	The lattice Boltzmann method	1
1.3	GPU implementation	2
1.4	Collision operators	2
1.5	Thermal LBM	3
1.6	Rayleigh-Bénard convection	3
1.7	Literature review	4
1.8	Research questions	4
1.9	Outline	4
2	Theory	6
2.1	Flow equations	6
2.2	Energy transport	6
2.3	Boltzmann equation	9
2.4	Velocity discretization	10
2.5	Spatial and Temporal discretization	11
2.5.1	First order forward Euler rule	11
2.5.2	Second order trapezoidal rule	11
2.5.3	Second order midpoint rule	12
2.6	Chapman-Enskog analysis	12
2.7	Advection-Diffusion solver	14
2.8	Coupling	14
3	Collision Operators	16
3.1	Filter Matrix method	16
3.2	Matrix projected Boltzmann	18
3.3	Multiple relaxation time method	19
3.4	Entropic Method	20
3.4.1	Essentially entropic method	22
3.5	Cumulant Method	22
3.6	Advection diffusion	23
4	Pre-study of collision schemes	24
4.1	Lid-driven cavity flow	24
4.1.1	Velocity fields	24
4.1.2	Stability grid	27
4.2	Doubly periodic shear layers	27
4.3	Initial conclusions	29
5	Filter Matrix and Multiple relaxation equivalence	32
5.1	Equivalent streaming	32
5.2	Equivalent collision	33
6	GPU optimization	36
6.1	GPU architecture	36
6.2	General Implementation	39
6.2.1	Kernel fusing	39

6.2.2	Sparse field saving	40
6.2.3	Coalescing	40
6.2.4	Constants	41
6.2.5	Loop-unrolling	41
6.3	Method specific implementation	41
6.3.1	Multiple relaxation time	41
6.3.2	Cumulant method	42
6.3.3	Entropic method	42
6.4	Thermal implementation	42
7	Results	43
7.1	Taylor-Green vortex	43
7.2	Lid-driven cavity	45
7.3	Rayleigh-Bénard convection	46
7.3.1	Nusselt number results	49
7.3.2	Simulation time	51
7.4	High Prandtl simulations	52
8	Conclusion	55
8.1	Findings	55
8.1.1	Qualitative Comparison	55
8.1.2	MRT and FMLB equivalence	56
8.1.3	Rayleigh-Bénard convection varying Rayleigh	56
8.1.4	Rayleigh-Bénard convection varying Prandtl	56
8.2	GPU optimization	56
8.3	Research questions	56
8.4	Limitations	57
8.5	Recommendations	57
A	Appendix	59
A.1	MRT and FMLB equivalence for general bulk viscosity	59
A.2	MRT and FMLB comparison for non-zero body forces	60

1 Introduction

As the demand for clean and reliable energy continues to grow, the development of advanced nuclear reactor technologies has become an increasingly important area of research. Among the proposed Generation IV reactor concepts, molten salt reactors (MSRs) have attracted considerable interest because of their high operating temperatures, passive safety characteristics, and potential for improved thermal efficiency. The design and optimization of these reactors rely heavily on accurate fluid simulations. The governing equations of fluid dynamics, such as the Navier-Stokes equations (NSE), are well-established but remain analytically intractable for all but the most simplified cases. Experiments can help to bridge this gap by validating models and establishing empirical relations between key variables. Some cases are intractable even to experiments. Examples include the turbulent motion of the Earth's core, large-scale movement of low and high pressure regions in the earth's atmosphere, or the operation of a nuclear reactor. Consequently, numerical simulations are often preferred when physical experiments are constrained by prohibitive costs, safety hazards, or extreme spatial and temporal scales.

1.1 Computational Fluid Dynamics

Computational fluid dynamics (CFD) aims to solve the equations governing fluid motion. Through the application of mathematical discretization techniques, the continuous governing equations are transformed into algebraic systems suitable for numerical computation. From these numerical solutions, quantities of interest such as velocity, temperature, or pressure fields can be obtained. Because the equations are discretized in space and time, the quantities extracted using this method are generally approximations to the exact solution. Numerous discretization methods exist, including finite difference, finite volume, and finite element methods, each offering different trade-offs in computational efficiency, accuracy, and stability. For nonlinear equations such as the Navier-Stokes equations, the discretization typically produces a nonlinear algebraic system that is solved through a sequence of linear systems. Although each discrete equation directly couples only a small number of neighboring variables, solving the resulting system propagates information throughout the domain, meaning that changes in one region can ultimately influence the solution everywhere. This reflects the fact that physical interactions are predominantly local, while global coupling arises naturally through the governing equations, particularly through the pressure field in incompressible flows.

1.2 The lattice Boltzmann method

An alternative approach is provided by the lattice Boltzmann method (LBM), a mesoscopic numerical method derived from kinetic theory [1, 2]. Rather than solving the macroscopic fluid equations directly, LBM solves a discretized Boltzmann equation whose macroscopic moments recover the desired governing equations. One of the method's most attractive features [1] is that its collision and streaming operations are highly local, making it exceptionally well suited to parallel computing architectures. In LBM a lattice is constructed over the entirety of the domain with, at every node, a finite collection of distributions f_i . The unknown variables of the target equation are obtained as moments of the distribution functions. At every timestep, two processes occur at every node in the lattice. First, every node undergoes a collision step which relaxes the values of the distributions defined at the nodes to a certain equilibrium distribution f_i^{eq} , then, every node undergoes streaming meaning that the distributions at the nodes are propagated to their neighboring nodes. The choice of equilibrium distribution and collision operator determines the macroscopic equations recovered by the method. For example, when solving for the Navier-Stokes equations, collision conserves mass and momentum. For diffusion-type

problems, the collision conserves the zeroth moment of the distribution functions, corresponding to the transported scalar quantity. When implementing the LBM algorithms in computer code, one can now make use of parts of the computer designed with parallelism in mind such as the graphics processing unit (GPU). Modern GPUs provide thousands of lightweight processing cores capable of executing many threads concurrently. Owing to this locality, LBM maps naturally onto GPU architectures and can achieve high computational throughput, significantly reducing computation time.

1.3 GPU implementation

The implementation of efficient LBM codes requires programming a GPU. The technical nuances of GPU-accelerated LBM methods significantly influence computational throughput, which is critical for high-resolution simulations. The primary performance bottleneck of LBM codes comes from reading and writing data to memory. Limiting the amount of data transfer, and understanding the different types of GPU memory locations is therefore vital to performance. In this thesis, LBM algorithms are implemented with performance in mind on a consumer RTX 3060 TI GPU with attention to coalesced memory access, kernel structure, and register pressure. The resulting implementation achieves up to 93% of the theoretical memory bandwidth ceiling of the hardware in 2D, comparable to state-of-the-art open-source LBM implementations [3].

1.4 Collision operators

Although LBM can achieve excellent computational performance, improving its numerical stability and extending its robustness across a broad range of flow conditions remain active areas of research. Under certain conditions, such as low kinematic viscosities, the original LBM method called "single relaxation time" (SRT) can become unstable or suffer from significant numerical errors. Numerous collision models have been proposed to improve the stability, accuracy, and robustness of the original SRT formulation [4–8]. SRT uses a single relaxation parameter linked to viscosity to relax all distributions towards their equilibrium at the same rate. One of the earliest and most widely used extensions developed by d’Humières et al. [4] called "multiple relaxation time" (MRT) changes the collision process by first transforming the distributions into moment space, and then relaxes each moment independently towards its equilibrium value. Additionally, each moment is relaxed with a separate relaxation rate, so that different moments can be relaxed independently of each other. Concurrently with the development of the MRT method, Somers introduced the filter matrix lattice Boltzmann (FMLB) method using a staggered timestepping approach and a newly defined collision function together with a moment transformation. Since its introduction, FMLB has received relatively little attention. Apart from the work of Zhuo et al. [9], it has primarily been investigated in a series of student projects [10–21]. Recently, Sundström [21] proposed the Matrix Projected Boltzmann (MPB) method, which builds upon the ideas introduced in FMLB while providing a more general framework for LBM methods where the target equation can be specified directly. Parallel to the development of FMLB and MRT, Karlin et al. [7] designed an LBM collision process based on the theory of entropy. By defining a discrete entropy functional, the resulting Entropic method enforces a discrete entropy condition to improve nonlinear stability, although this may introduce additional numerical dissipation. Building on the MRT method, Geier et al. [8] proposed the Cumulant method which performs relaxation in cumulant space rather than moment space. These four methods represent only a subset of the many collision models proposed for LBM. Given the proliferation of collision models, establishing the mathematical and algorithmic relationship between these methods is essential. In the paper by Coreixas et al. [22] this question is explored for the SRT, MRT, and Cumulant method, along with other methods called "regularized lattice Boltzmann methods"

originally proposed by Latt and Chopard [23]. Coreixas et al. demonstrate that both the regularized lattice Boltzmann method, as well as a variant called the "recursively regularized lattice Boltzmann method" are algorithmically equivalent to MRT and the Central moment method (a precursor to the cumulant method) respectively given a certain choice of the transformation to moments. In the analysis by Coreixas et al. [22], the FMLB method is not included, and the Entropic method is not compared. Consequently, it remains unclear how FMLB relates to the more established collision models. In particular, it is unknown whether FMLB is fundamentally distinct or whether it can be interpreted within one of the existing LBM frameworks. Addressing this question is one of the primary objectives of this thesis.

1.5 Thermal LBM

Similarly to more conventional CFD methods, LBM can also be used to simulate transport of energy. The relaxation rates and equilibria can be redefined exactly so that an advection-diffusion equation (ADE) is solved instead of the Navier-Stokes equations. The ADE requires knowing the velocity of the host medium at every point in the domain. This velocity can either come from a predefined velocity field, or, more interestingly, can be given by an LBM simulation simultaneously solving the Navier-Stokes equation. This so-called "double distribution function" approach defines a one-way coupled system where the temperature field is advected by the velocity field. In reality the velocity field is influenced through buoyancy forces. An increased temperature expands the fluid, decreasing its density at a rate depending on the thermal expansion coefficient. A fluid parcel whose density differs from that of its surroundings experiences a buoyancy force. A two-way coupled system, where both fields affect each other, can then be defined by expressing the buoyancy force as a function of temperature. This coupled system can be characterized by the Rayleigh number indicating the relative strength of buoyancy forces, and the Prandtl number expressing the ratio of momentum diffusion to thermal diffusion. Just like for the Navier-Stokes solver, the temperature solver becomes unstable as thermal diffusivity decreases. This suggests a limit in terms of Rayleigh and Prandtl numbers. For fixed geometry and driving conditions, increasing the Rayleigh number corresponds to decreasing the momentum and/or thermal diffusivity. For fixed kinematic viscosity, increasing the Prandtl number corresponds to decreasing the thermal diffusivity. In a system where both the Rayleigh number and the Prandtl number are large, the thermal diffusivity would be pushed to increasingly small numbers. The regime of large Rayleigh and Prandtl numbers is very interesting to nuclear reactor physics, as this is the exact regime of molten salts inside molten salt reactors with $Pr \sim 5 - 35$ [24]. To research the functionality of molten salt reactors it is therefore important to be able to obtain accurate and stable simulations in the high Rayleigh and Prandtl regime.

1.6 Rayleigh-Bénard convection

The canonical test case for simulating buoyancy driven flow is given by Rayleigh-Bénard (RB) convection. An RB setup consists of a cold plate above a hot plate. Near the hot plate, the fluid is heated causing an upwards buoyancy force. Near the cool plate, the fluid is cooled causing a downwards buoyancy force. The interplay between these buoyancy forces results in a circulating flow called a Bénard convection cell. The behavior of Bénard convection cells depends on the Rayleigh and Prandtl number. At large Rayleigh number, buoyancy dominates over viscous forces, and increasingly turbulent convection cells form. At low Rayleigh numbers, viscous forces dominate so that convection is suppressed and heat transfer occurs primarily through conduction. The increase in heat transfer between low Rayleigh numbers and high Rayleigh numbers is encapsulated in the Nusselt number. A Nusselt number of 1 means heat is transferred only through conduction, a higher Nusselt number means heat is also transported

by fluid motion. To quantify the relation of Nusselt numbers on Rayleigh and Prandtl numbers, researchers try to find scaling laws in the form $Nu \sim Ra^\alpha Pr^\beta$. These scaling laws are then fitted for α and β either empirically or by doing high resolution simulations. In this thesis LBM is used in 2D to simulate RB convection. The choice for 2D is primarily for computational efficiency. 2D simulations are much faster allowing for fast iteration and testing. 2D RB convection differs from 3D. 3D flows have a fundamentally different energy cascade compared to 2D meaning that their relations between variables are not directly comparable.

1.7 Literature review

In the context of 2D RB convection, Wang et al. [25] performed simulations with the finite differences method using a Rayleigh number of $10^7 - 10^9$, with Prandtl numbers $1 - 100$ using free-slip boundary conditions for the bottom and top wall, and periodic boundary conditions for the side-walls. Wang et al. found no significant impact of the Prandtl number on the Nusselt number. However, free-slip boundary conditions suppress momentum boundary layer formation, which is the primary mechanism through which the Prandtl number influences heat transfer, making these results difficult to generalise to no-slip setups. Similarly, Johnston and Doering [26] used spectral methods to simulate Rayleigh numbers $1 - 10^{10}$ at Prandtl number $Pr = 1$ using no-slip bottom and top boundary conditions, with periodic side-walls and found a $Nu = 0.138Ra^{0.285}$ relation. Poel et al. used the finite differences method [27] to simulate Rayleigh of 10^8 and varied Prandtl from $0.045 - 55$ using no-slip boundaries for the bottom and top walls, as well as the side walls. One region of parameter space that has received relatively little attention is that of no-slip bottom and top boundaries, with periodic side walls, and varying Prandtl. To the author's knowledge, the combination of no-slip bottom and top walls with periodic side-walls and varying Prandtl across a range of Rayleigh numbers has not been systematically studied in 2D LBM, leaving open the question whether Prandtl dependence emerges under these conditions.

1.8 Research questions

This thesis addresses these gaps through four research questions:

- How do SRT, MRT, FMLB, and MPB compare in terms of stability on isothermal simulations?
- How do the FMLB and MPB methods compare mathematically to the MRT method?
- How do the SRT, MRT, Entropic, and Cumulant LBM compare when simulating thermal Rayleigh-Bénard flow in terms of stability and accuracy?
- What is the effect of variations in Prandtl number from 1 to 35 in 2D Rayleigh-Bénard flow with no-slip bottom and top walls and periodic side-walls?

1.9 Outline

The outline of this thesis is as follows. Chapter 2 recaps the theory of the standard lattice Boltzmann method along with the extension to thermal coupling. Chapter 3, the advanced collision operators are presented along with their derivation where applicable. Chapter 4 is a pre-study qualitatively comparing the LBM methods in terms of accuracy and stability using lid-driven cavity flow, and the doubly periodic shear layer. Chapter 5, the filter matrix and matrix projected Boltzmann methods are compared to the multiple relaxation method, where it is proven that the former are a subset of the latter. Chapter 6 provides details on the numerical

implementation used for this thesis focusing on methods to achieve high-performance GPU codes. Chapter 7, the quantitative results are presented using the Taylor-Green vortex, lid-driven cavity flow, and Rayleigh-Bénard convection benchmarks. Finally, chapter 8 concludes the thesis reporting findings, acknowledging limitations, and giving recommendations for future research.

2 Theory

2.1 Flow equations

The Navier-Stokes equations govern the evolution of mass and momentum in a fluid and are given by:

$$\begin{aligned}\partial_t \rho + \partial_\alpha \rho u_\alpha &= 0 \\ \partial_t \rho u_\alpha + \partial_\beta \rho u_\alpha u_\beta &= -\partial_\alpha p + \partial_\beta (\rho \nu [\partial_\beta u_\alpha + \partial_\alpha u_\beta - \frac{2}{3} \partial_\gamma u_\gamma \delta_{\alpha\beta}] + \zeta \partial_\gamma u_\gamma \delta_{\alpha\beta}) + F_\alpha\end{aligned}\tag{1}$$

where Greek indices are subject to Einstein summation convention, ρ denotes the fluid density, u_α the velocity in direction α , p pressure, ν kinematic viscosity, ζ bulk viscosity, and F_α a body force in direction α . The first equation is commonly referred to as the continuity equation, and the second as the momentum equation. Under the assumption of incompressible flow, the density of each fluid parcel remains constant along its trajectory, and the continuity equation reduces to:

$$\partial_\alpha u_\alpha = 0.\tag{2}$$

If the viscosity is additionally taken to be constant throughout the fluid, the momentum equation simplifies to:

$$\rho \partial_t u_\alpha + \rho u_\beta \partial_\beta u_\alpha = -\partial_\alpha p + \rho \nu \partial_\beta^2 u_\alpha + F_\alpha.\tag{3}$$

Equations (2) and (3) together constitute the incompressible Navier-Stokes equations. Throughout this thesis, the term "Navier-Stokes equations" refers to this incompressible formulation, while Eq. (1) is referred to explicitly as the "compressible Navier-Stokes equations" where relevant. Through non-dimensionalization, the characteristic behavior of the Navier-Stokes equations can be characterized by the Reynolds number:

$$Re = \frac{UL}{\nu}\tag{4}$$

where U and L denote the characteristic velocity and length scale of the system, respectively. At low Reynolds numbers, momentum diffusion dominates, and the flow is described as laminar, whereas at high Reynolds numbers, inertia effects dominate and the flow becomes turbulent. A second relevant non-dimensional parameter is the Mach number:

$$Ma = \frac{|\mathbf{u}|}{c_s}\tag{5}$$

with c_s the speed of sound in the fluid. The Mach number is a physical dimensionless parameter characterizing compressibility effects. In lattice Boltzmann simulations, the lattice Mach number is chosen sufficiently small (typically $Ma < 0.3$ [1]) to ensure recovery of the incompressible Navier-Stokes equations and to minimize compressibility errors.

2.2 Energy transport

To describe thermally driven flows, the Navier-Stokes equations must be supplemented by an energy equation governing the evolution of temperature. In this work, fluid properties are assumed constant and heat conduction is assumed to obey Fourier's law. Furthermore, viscous

dissipation, compressibility effects, and internal heat generation are neglected. Under these assumptions, the temperature field is governed by the advection-diffusion equation:

$$\partial_t T + u_\alpha \partial_\alpha T = \kappa \partial_\alpha^2 T \quad (6)$$

with temperature T , and κ the thermal diffusivity analogous to ν for momentum evolution. u_α is the velocity of the fluid.

When a fluid is heated locally, it expands decreasing its density. At constant pressure the thermal expansion coefficient of a fluid is given by [1]:

$$\beta = -\frac{\rho_0}{\rho^2} \left(\frac{\partial \rho}{\partial T} \right)_p \quad (7)$$

with ρ_0 the density at some reference temperature T_0 , and β the expansion coefficient of the fluid. If temperature and density changes are assumed to be small, Eq. (7) can be linearized around ρ_0 and T_0 so that:

$$\rho \approx \rho_0(1 - \beta(T - T_0)). \quad (8)$$

If gravity is taken as the only body force acting on the system, the corresponding body force term can be expressed as:

$$F_{b,\alpha} = -\beta \rho_0 (T - T_0) g_\alpha \quad (9)$$

with g_α the gravitational acceleration and $F_{b,\alpha}$ the buoyancy force per volume. Eq. (9) is called the Boussinesq approximation. A coupled system of equations describing the evolution of thermal flows can now be modeled using Eqs. (2, 3, 6, 9). This coupled system gives rise to additional non-dimensional numbers characterizing the flow. The Prandtl number:

$$Pr = \frac{\nu}{\kappa}, \quad (10)$$

is the ratio of kinematic viscosity to thermal diffusivity. Consider the case of Rayleigh-Bénard convection which consists of two parallel, horizontal plates, the bottom one at temperature T_h , the top one at T_c with $T_h > T_c$. For this configuration, the Rayleigh number can be defined:

$$Ra = \frac{|g|\beta(T_h - T_c)H^3}{\kappa\nu}, \quad (11)$$

with H the height between the plates. The Rayleigh number indicates the ratio of buoyancy forces over viscous forces. At low Rayleigh numbers the system will stay in a pure diffusion state with a linear temperature profile between the top and bottom plate. At high Rayleigh numbers the buoyancy forces are able to overcome the viscous forces, and convection cells form, transporting hot fluid from the bottom to the top plate and cold fluid in the opposite direction. An increase in Rayleigh number correspondingly increases the heat transferred from bottom to top wall. The heat transfer can be characterized by an additional non-dimensional number called the Nusselt number:

$$Nu = \frac{q_{tot}}{q_{cond}}, \quad (12)$$

with q_{tot} the total vertical heat flux, and q_{cond} the heatflux that would be present if heat were transported purely by conduction. The Nusselt number shows the increase of heat transfer due to convection. If $Nu = 1$, heat is transferred by conduction alone in a static velocity field. For $Nu > 1$ convection contributes to heat transfer. To derive an explicit expression for Nu in terms of the flow field, consider Eq. (6) written in flux form, using incompressibility ($\partial_\alpha u_\alpha = 0$):

$$\partial_t T + \partial_y(u_y T) = \kappa \partial_y^2 T, \quad (13)$$

where the horizontal advective and diffusive terms have already been dropped in anticipation of the averaging procedure below, and y denotes the wall-normal direction. Averaging over the horizontal direction and over time, denoted $\langle \cdot \rangle$, and invoking statistical stationarity ($\langle \partial_t T \rangle = 0$) together with periodicity in the horizontal direction, gives:

$$\frac{\partial}{\partial y} \langle u_y T \rangle = \kappa \frac{\partial^2 \langle T \rangle}{\partial y^2}, \quad (14)$$

which can be rewritten as:

$$\frac{\partial}{\partial y} \left[\langle u_y T \rangle - \kappa \frac{\partial \langle T \rangle}{\partial y} \right] = 0. \quad (15)$$

Eq. (15) shows that the quantity in brackets, the total vertical heat flux q_{tot} is independent of y :

$$q_{tot} = \langle u_y T \rangle - \kappa \frac{\partial \langle T \rangle}{\partial y} = \text{const.} \quad (16)$$

This is a statement of energy conservation: whatever heat flux enters the domain at the bottom plate must leave at the top plate. At the plates, the no-slip conditions require $u_y = 0$, so that q_{tot} reduces there to a purely conductive flux. Since q_{tot} is constant in y , it may equivalently be evaluated by averaging over the full height H of the domain rather than at the wall. Using:

$$\int_0^H \frac{\partial \langle T \rangle}{\partial y} dy = T_c - T_h, \quad (17)$$

the diffusive contribution to q_{tot} averaged over the domain height becomes:

$$\left\langle \frac{\partial \langle T \rangle}{\partial y} \right\rangle_y = \frac{T_c - T_h}{H}, \quad (18)$$

so that:

$$q_{tot} = \langle u_y T \rangle - \kappa \frac{T_c - T_h}{H} = \langle u_y T \rangle + \frac{\kappa(T_h - T_c)}{H}, \quad (19)$$

where $\langle u_y T \rangle$ now denotes an average over the full domain and over time. The reference conductive flux q_{cond} is the flux that would occur in the absence of flow, i.e. for a linear conductive profile between the plates:

$$q_{cond} = \frac{\kappa(T_h - T_c)}{H}. \quad (20)$$

Substituting q_{tot} and q_{cond} into Eq. (12) yields the working expression for the Nusselt number used throughout this thesis:

$$Nu = 1 + \frac{H}{\kappa(T_h - T_c)} \langle u_y T \rangle. \quad (21)$$

The first term represents the conductive baseline ($Nu = 1$), while the second term quantifies the enhancement of heat transfer due to convective transport of temperature by the velocity field.

2.3 Boltzmann equation

Rather than directly solving the Navier-Stokes equation, the LBM method solves the Boltzmann equation. The Boltzmann equation describes the evolution of distributions of particles. At the atomic level, any fluid is comprised of many discrete particles. When taking a system large enough that the fluid can be described as a continuum, the Boltzmann equation is expected to recover the same behaviour of these particles as the Navier-Stokes equation. The Boltzmann equation in the absence of body forces is given by:

$$\frac{\partial f(\vec{x}, t, \vec{\xi})}{\partial t} + \vec{\xi} \cdot \nabla f(\vec{x}, t, \vec{\xi}) = \Omega(f)(\vec{x}, t, \vec{\xi}), \quad (22)$$

where f is the density distribution at position $\vec{x}$, time t , and with microscopic velocity $\vec{\xi}$. $\Omega(f)$ governs the temporal evolution of f over time and is called the collision operator. The distribution is defined so that:

$$\begin{aligned} \int_{\vec{\xi}} f(\vec{x}, t, \vec{\xi}) d\vec{\xi} &= \rho, \\ \int_{\vec{\xi}} \vec{\xi} f(\vec{x}, t, \vec{\xi}) d\vec{\xi} &= \rho \vec{u}, \end{aligned} \quad (23)$$

with ρ and $\vec{u}$ the density and macroscopic velocity at that $(\vec{x}, t)$. $\Omega(f)$ is defined so that:

$$\begin{aligned} \int_{\vec{\xi}} \Omega(f)(\vec{x}, t, \vec{\xi}) d\vec{\xi} &= 0, \\ \int_{\vec{\xi}} \vec{\xi} \Omega(f)(\vec{x}, t, \vec{\xi}) d\vec{\xi} &= 0, \end{aligned} \quad (24)$$

reflecting the conservation of density and momentum under collision. For the recovery of the Navier-Stokes equations (or any other PDE), an appropriate $\Omega(f)$ must be chosen. The most widely used collision operator was introduced by Bhatnagar, Gross, and Krook (BGK) [28]:

$$\Omega(f) = \bar{\omega}(f^{eq} - f), \quad (25)$$

in which $\bar{\omega} = \frac{c_s^2}{\nu}$ is called the relaxation rate with c_s the speed of sound of the fluid, and ν its kinematic viscosity. f^{eq} is called the equilibrium distribution, and, using BGK, is given by the Maxwell-Boltzmann distribution [1]:

$$f^{eq} = \rho \left(\frac{1}{2\pi RT} \right)^{D/2} e^{-\frac{|\vec{\xi} - \vec{u}|^2}{2RT}}, \quad (26)$$

with R the gas constant, D the spatial dimension, and T the temperature. Taking the integrals over $\Omega(f)$: (which can be verified by taking integrals of the Maxwell-Boltzmann distribution)

$$\begin{aligned} \int_{\vec{\xi}} f_i^{eq}(\vec{x}, t, \vec{\xi}) d\vec{\xi} &= \rho(\vec{x}, t) \\ \int_{\vec{\xi}} \vec{\xi} f_i^{eq}(\vec{x}, t, \vec{\xi}) d\vec{\xi} &= \rho \vec{u}(\vec{x}, t) \end{aligned} \quad (27)$$

2.4 Velocity discretization

To solve (22) using computational methods, the equation must first be discretized. To discretize the velocity, f and f^{eq} are projected onto set of multidimensional Hermite polynomials [29] defined as:

$$\mathcal{H}^{(n)}(\vec{\xi}) = \frac{(-1)^n}{\sigma(\vec{\xi})} \nabla^n \sigma(\vec{\xi}) \quad (28)$$

with $\sigma(\vec{\xi})$ so that:

$$\sigma(\vec{\xi}) = \frac{1}{(2\pi)^{D/2}} e^{-\vec{\xi}^2/2}. \quad (29)$$

In these equations, (n) is defined as the rank of the tensor, e.g. $n = 0$ gives a scalar, $n = 1$ gives a vector, etc. In the end, the velocity discretized distribution f_i is defined so that:

$$f_i(\vec{x}, t) = \frac{w_i}{\sigma(\vec{\xi}_i)} f(\vec{x}, t, \vec{\xi}_i), \quad (30)$$

in which w_i and $\vec{\xi}_i$ are quadrature weights and points respectively. Both $\vec{\xi}_i$, and w_i are determined by the velocity discretization scheme. For example, in the widely used two-dimensional nine-velocity lattice notated by D2Q9,

$$w_i \in \{4/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, 1/36\}, \quad (31)$$

and

$$\vec{c}_i = \vec{\xi}_i / \sqrt{3} \in \{(0, 0), (1, 0), (0, -1), (-1, 0), (0, 1), (1, 1), (1, -1), (-1, -1), (-1, 1)\}. \quad (32)$$

The discretized equilibrium distribution takes the form:

$$f_i^{eq} = w_i \rho \left(1 + \frac{\vec{u} \cdot \vec{c}_i}{c_s^2} + \frac{(\vec{u} \cdot \vec{c}_i)^2}{2c_s^4} - \frac{\vec{u} \cdot \vec{u}}{2c_s^2} \right), \quad (33)$$

now with c_s^2 redefined as the lattice speed of sound, which, for D2Q9 is given by $c_s^2 = 1/3$. The density ρ , and velocity $\vec{u}$ can now also be expressed as a function of f_i and $\vec{c}_i$:

$$\begin{aligned} \sum_i f_i &= \rho \\ \sum_i \vec{c}_i f_i &= \rho \vec{u}, \end{aligned} \quad (34)$$

and also:

$$\begin{aligned} \sum_i f_i^{eq} &= \rho \\ \sum_i \vec{c}_i f_i^{eq} &= \rho \vec{u}, \end{aligned} \quad (35)$$

Finally, the velocity discretized Boltzmann equation reads:

$$\frac{\partial f_i(\vec{x}, t)}{\partial t} + \vec{c}_i \cdot \nabla f_i(\vec{x}, t) = \Omega_i(f)(\vec{x}, t) \quad (36)$$

where, for BGK, the collision operator is $\Omega_i(f) = \bar{\omega}(f_i^{eq} - f_i)$

2.5 Spatial and Temporal discretization

2.5.1 First order forward Euler rule

Next up, space and time are discretized by recognizing that Eq. (36) is written as a material derivative, with f_i being advected with velocity $\vec{c}_i$. This allows writing Eq. (36) as:

$$\frac{Df_i}{Dt}(\vec{x}(t), t) = \Omega_i(f)(\vec{x}(t), t) \quad (37)$$

with $\vec{x}(t) = \vec{x}_0 + \vec{c}_i t$. Integrating both sides from t to $t + \Delta t$:

$$f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) - f_i(\vec{x}, t) = \int_t^{t+\Delta t} \Omega_i(f)(\vec{x}(t), t) dt. \quad (38)$$

At this point the right hand side of Eq. (38) must be approximated in order to solve this system. A simple first order accurate approximation can be done by taking $\int_t^{t+\Delta t} \Omega_i(f)(\vec{x}(t), t) dt = \Delta t \Omega_i(f)(\vec{x}, t) + \mathcal{O}(\Delta t)$. A solvable explicit timestepping system can now be written down:

$$f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) - f_i(\vec{x}, t) = \Delta t \bar{\omega} (f_i^{eq}(\vec{x}, t) - f_i(\vec{x}, t)) + \mathcal{O}(\Delta t) \quad (39)$$

with the BGK operator.

2.5.2 Second order trapezoidal rule

In subsequent analysis it will be shown that Eq. (39) does not converge to the Navier-Stokes equation for $\Delta t \rightarrow 0$. This however, can be fixed by using a second order approximation of the right hand side of Eq. (38). By using the trapezoidal rule, Eq. (38) becomes:

$$f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) - f_i(\vec{x}, t) = \frac{\Delta t}{2} \left(\Omega_i(f)(\vec{x}, t) + \Omega_i(f)(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) \right) + \mathcal{O}(\Delta t^2). \quad (40)$$

Taking the sum over Eq. (40), and recognizing that $\sum_i \Omega_i(f) = 0$, Eq. (40) shows that:

$$\rho(\vec{x}, t) = \rho(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) + \mathcal{O}(\Delta t^2). \quad (41)$$

Just like with density, the same can be said for the momentum so that:

$$\vec{u}(\vec{x}, t) = \vec{u}(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) + \mathcal{O}(\Delta t^2). \quad (42)$$

This means that, because the equilibrium function is purely a function of density and velocity:

$$f_i^{eq}(\vec{x}, t) = f_i^{eq}(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) + \mathcal{O}(\Delta t^2). \quad (43)$$

Using this fact, Eq. (40) together with BGK can now be rewritten:

$$f_i(\vec{x} + \vec{c}_i \Delta t) - f_i(\vec{x}, t) = \frac{\Delta t \bar{\omega}}{2} (f_i^{eq}(\vec{x}, t) - f_i(\vec{x}, t) + f_i^{eq}(\vec{x}, t) - f_i(\vec{x} + \vec{c}_i \Delta t)) + \mathcal{O}(\Delta t^2), \quad (44)$$

resulting in the computational scheme:

$$f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) - f_i(\vec{x}, t) = \Delta t \omega (f_i^{eq}(\vec{x}, t) - f_i(\vec{x}, t)). \quad (45)$$

This derivation provides a redefined relaxation rate $\omega = \frac{1}{\frac{1}{\bar{\omega}} + \frac{\Delta t}{2}}$. This scheme is called the Single Relaxation Time (SRT) method, referring to ω being the only tunable parameter.

2.5.3 Second order midpoint rule

While Eq. (45) is the standard LBM method used in literature [2, 30, 31], some papers [6, 9, 32] use a slightly different spatial and temporal discretization based on the midpoint rule. In Eq.(38), instead of integrating from t to $t + \Delta t$, the integration can be done from $t - \Delta t/2$ to $t + \Delta t/2$. The right hand side is then approximated by taking the value of $\Omega(f)$ at the middle of the integration range. Doing this gives:

$$f_i(\vec{x} + \vec{c}_i \Delta t/2, t + \Delta t/2) - f_i(\vec{x} - \vec{c}_i \Delta t/2, t - \Delta t/2) = \Delta t \Omega_i(f)(\vec{x}, t) + \mathcal{O}(\Delta t^2). \quad (46)$$

Evaluating the right hand side requires knowing f_i at the point $(\vec{x}, t)$. This can be approximated by using a first order Taylor Expansion together with Eq. (37):

$$f_i(\vec{x} - \vec{c}_i \Delta t/2, t - \Delta t/2) = f_i(\vec{x}, t) - \frac{\Delta t}{2} \Omega(f)(\vec{x}, t) + \mathcal{O}(\Delta t^2). \quad (47)$$

Again using the BGK operator, finding equivalence in equilibria up to first order, and solving for $f_i(\vec{x}, t)$ gives:

$$f_i(\vec{x}, t) = \frac{1}{1 + \frac{\Delta t}{2} \bar{\omega}} f_i(\vec{x} - \vec{c}_i \Delta t/2, t - \Delta t/2) + \frac{\frac{\Delta t}{2} \bar{\omega}}{1 + \frac{\Delta t}{2} \bar{\omega}} f_i^{eq}(\vec{x} - \vec{c}_i \Delta t/2, t - \Delta t/2) + \mathcal{O}(\Delta t^2). \quad (48)$$

Which is then plugged into Eq. (46) so that the computational algorithm becomes:

$$\begin{aligned} & f_i(\vec{x} + \vec{c}_i \Delta t/2, t + \Delta t/2) - f_i(\vec{x} - \vec{c}_i \Delta t/2, t - \Delta t/2) = \\ & \Delta t \omega (f_i^{eq}(\vec{x} - \vec{c}_i \Delta t/2, t - \Delta t/2) - f_i(\vec{x} - \vec{c}_i \Delta t/2, t - \Delta t/2)) \end{aligned} \quad (49)$$

with the same definition for ω as in Eq. (45). It is important to realize that Eq. (45) and Eq. (49) would be implemented on a computer in the exact same way. One would take an incoming distribution, calculate its density and velocity, calculate the equilibrium distribution, relax the incoming distribution towards the equilibrium distribution with relaxation rate ω , and finally assign the resulting distribution the next timestep on the nearby nodes. Throughout the rest of this thesis, Eq. (45) will be preferred where possible for ease of notation unless stated otherwise.

2.6 Chapman-Enskog analysis

With the Boltzmann equation now discretized, is necessary to analyze under which conditions the Navier-Stokes equation is recovered. The canonical analysis method for this task is the so-called Chapman-Enskog analysis [1]. The Chapman-Enskog analysis consists of a Taylor expansion followed by a multiscale expansion, and a moment transformation. In the multiscale expansion, f_i , ∂_t , and ∂_α (where α represents a spatial index subject to Einstein notation) are expanded in orders of ϵ . ϵ is a dummy parameter, used to separate spatial and temporal scales. The multiscale expansion is as follows [33]:

$$\begin{aligned} f_i &= f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} \\ \partial_t &= \epsilon \partial_t^{(1)} + \epsilon^2 \partial_t^{(2)} \\ \partial_\alpha &= \epsilon \partial_\alpha^{(1)} \end{aligned} \quad (50)$$

To understand the moment transformation, it is first important to know what moments are in LBM. The definition for a moment m_k is given by:

$$\sum_i c_{ix}^a c_{iy}^b c_{iz}^c f_i \quad (51)$$

where $a + b + c$ is the order of the moments. Using this definition, the density can be found as the zeroth order moment, and the momentum can be found as the first order moments. Starting from Eq. (45), the Taylor expansion up to second order is given by:

$$\begin{aligned} c_{i\alpha} \partial_\alpha f_i(\vec{x}, t) + \partial_t f_i(\vec{x}, t) + \frac{1}{2} \Delta t (c_{i\alpha} \partial_\alpha)^2 f_i(\vec{x}, t) + \frac{1}{2} \Delta t \partial_t^2 f_i(\vec{x}, t) + \Delta t c_{i\alpha} \partial_\alpha \partial_t f_i(\vec{x}, t) + \mathcal{O}(\Delta t^2) \\ = \omega(f_i^{eq}(\vec{x}, t) - f_i(\vec{x}, t)). \end{aligned} \quad (52)$$

For brevity, function arguments are omitted in what follows. The next step is to use the multiscale expansion from Eq. (50) and collect equations for every order of ϵ . This procedure then gives the following equations:

$$\mathcal{O}(\epsilon^0) : \quad f_i^{(0)} = f_i^{eq} \quad (53)$$

$$\mathcal{O}(\epsilon^1) : \quad \partial_t^{(1)} f_i^{(0)} + c_{i\alpha} \partial_\alpha^{(1)} f_i^{(0)} = -\omega f_i^{(1)} \quad (54)$$

$$\begin{aligned} \mathcal{O}(\epsilon^2) : \quad \partial_t^{(2)} f_i^{(0)} + \partial_t^{(1)} f_i^{(1)} + c_{i\alpha} \partial_\alpha^{(1)} f_i^{(1)} \\ + \frac{\Delta t}{2} \partial_t^{2(1)} f_i^{(0)} + \frac{\Delta t}{2} (c_{i\alpha} \partial_\alpha^{(1)})^2 f_i^{(0)} + \Delta t c_{i\alpha} \partial_\alpha^{(1)} \partial_t^{(1)} f_i^{(0)} = -\omega f_i^{(2)} \end{aligned} \quad (55)$$

$$\mathcal{O}(\epsilon^2) : \quad \rightarrow \partial_t^{(2)} f_i^{(0)} + (\partial_t^{(1)} + c_{i\alpha} \partial_\alpha^{(1)}) (1 - \frac{\Delta t}{2} \omega) f_i^{(1)} = -\omega f_i^{(2)}.$$

Prior to the transformation into moment space, an extra assumption is made. From Eq. (35) and Eq. (54), $\sum_i (f_i^{(1)} + \epsilon f_i^{(2)}) = 0$, and also $\sum_i c_{i\alpha} (f_i^{(1)} + \epsilon f_i^{(2)}) = 0$. To Solve the subsequent system, it is now also assumed that first and second order individually sum to 0:

$$\begin{aligned} \sum_i f_i^{(1)} = \sum_i f_i^{(2)} = 0 \\ \sum_i c_{i\alpha} f_i^{(1)} = \sum_i c_{i\alpha} f_i^{(2)} = 0 \end{aligned} \quad (56)$$

Following the derivation in [1], the following equations are obtained:

$$\begin{aligned} \partial_t \rho + \partial_\alpha \rho u_\alpha + \mathcal{O}(\Delta t^2, \epsilon^3) = 0 \\ \partial_t \rho u_\alpha + \partial_\beta^{(1)} \left(\Pi_{\alpha\beta}^{(0)} + \Pi_{\alpha\beta}^{(1)} + \frac{\Delta t}{2} (\partial_t^{(1)} \Pi_{\alpha\beta}^{(0)} + \partial_\gamma^{(1)} \Xi_{\alpha\beta\gamma}^{(0)}) \right) + \mathcal{O}(\Delta t^2, \epsilon^3) = 0 \end{aligned} \quad (57)$$

where $\Pi_{\alpha\beta}^{(0)} = \sum_i c_{i\alpha} c_{i\beta} f_i^{(0)}$, $\Pi_{\alpha\beta}^{(1)} = \sum_i c_{i\alpha} c_{i\beta} f_i^{(1)}$, and $\Xi_{\alpha\beta\gamma}^{(0)} = \sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} f_i^{(0)}$. Further algebraic manipulation yields:

$$\begin{aligned} \Pi_{\alpha\beta}^{(0)} &= c_s^2 \rho \delta_{\alpha\beta} + \rho u_\beta u_\alpha \\ \Pi_{\alpha\beta}^{(1)} &= -\frac{1}{\omega} \rho c_s^2 \partial_\alpha^{(1)} u_\beta - \frac{1}{\omega} \rho c_s^2 \partial_\beta^{(1)} u_\alpha + \frac{1}{\omega} \partial_\gamma^{(1)} (\rho u_\alpha u_\beta u_\gamma) \\ \Xi_{\alpha\beta\gamma}^{(0)} &= \rho c_s^2 (\delta_{\alpha\beta} u_\gamma + \delta_{\alpha\gamma} u_\beta + \delta_{\beta\gamma} u_\alpha) \end{aligned} \quad (58)$$

which can be inserted into Eq. (57) to finally get:

$$\begin{aligned} \partial_t \rho u_\alpha + \partial_\beta \rho u_\alpha u_\beta = \\ -\partial_\alpha p + \partial_\beta (\rho \nu [\partial_\beta u_\alpha + \partial_\alpha u_\beta - \frac{2}{3} \partial_\gamma u_\gamma \delta_{\alpha\beta}]) + \frac{2}{3} \rho \nu \partial_\gamma u_\gamma \delta_{\alpha\beta} - \frac{\nu}{c_s^2} \partial_\gamma (\rho u_\alpha u_\beta u_\gamma) + \mathcal{O}(\Delta t^2, \epsilon^3) \end{aligned} \quad (59)$$

where $\nu = c_s^2(\frac{1}{\omega} - \frac{\Delta t}{2})$ or equivalently $\omega = \frac{1}{\frac{\nu}{c_s^2} + \frac{\Delta t}{2}}$, and $p = c_s^2 \rho$. Eq. (59) can be found as the compressible Navier-Stokes equation with bulk viscosity $\zeta = \frac{2}{3} \rho \nu$, and error term $\frac{\nu}{c_s^2} \partial_\gamma (\rho u_\alpha u_\beta u_\gamma)$ and terms proportional to $\mathcal{O}(\Delta t^2, \epsilon^3)$. Under constant viscosity, and near incompressibility the incompressible Navier-Stokes equation is approximately recovered:

$$\partial_t \rho u_\alpha + \partial_\beta \rho u_\alpha u_\beta = -\partial_\alpha p + \rho \nu \partial_\beta^2 u_\alpha - \frac{\nu}{c_s^2} \partial_\beta \partial_\gamma (\rho u_\alpha u_\beta u_\gamma) + \mathcal{O}(\Delta t^2, \epsilon^3). \quad (60)$$

The unphysical term of order $\mathcal{O}(u^3)$ is small when the Mach number $Ma = u/c_s \ll 1$. The Navier-Stokes equation is therefore recovered under the assumptions that $\Delta t, \epsilon$, and the Mach number are all small. Finally, the requirements on the equilibrium distribution are summarized, which will be important for analysis in subsequent chapters. For convergence to the Navier-Stokes equation, these are:

$$\begin{aligned} \sum_i f_i^{eq} &= \rho \\ \sum_i c_{i\alpha} f_i^{eq} &= \rho u_\alpha \\ \sum_i c_{i\alpha} c_{i\beta} f_i^{eq} &= \Pi_{\alpha\beta}^{(0)} = c_s^2 \rho \delta_{\alpha\beta} + \rho u_\beta u_\alpha. \end{aligned} \quad (61)$$

2.7 Advection-Diffusion solver

LBM is most naturally understood as a fluid dynamics solver in which groups of particles propagate, collide, and propagate again. Nevertheless, LBM can equally be applied to solve the advection-diffusion equation in Eq. (6) [1]. Instead of f_i , the ADE solver distribution is usually notated with g_i so that $\sum_i g_i = T(x_\alpha, t)$. A standard SRT-based ADE algorithm is given by:

$$g_i(x_\alpha + c_{i\alpha} \Delta t, t + \Delta t) = g_i(x_\alpha, t) + \Delta t \omega_T (g_i^{eq}(x_\alpha, t) - g_i(x_\alpha, t)) \quad (62)$$

with $\omega_T = \frac{1}{\frac{c_s^2}{\kappa} + \frac{\Delta t}{2}}$, and for the equilibrium:

$$g_i^{eq} = w_i T \left(1 + \frac{c_{i\alpha} u_\alpha}{c_s^2} + \frac{(c_{i\alpha} u_\alpha)^2}{2c_s^4} - \frac{u_\alpha u_\alpha}{2c_s^2} \right) \quad (63)$$

where u_α can either be a pre-determined velocity field, or can be obtained through some other simulation.

2.8 Coupling

Coupling is achieved by employing two LBM solvers on the same grid, one solving the NSE, and one solving the ADE. At every lattice node, the ADE solver then uses the local velocity, and uses it to advect energy. In the NSE solver, the local temperature can be used to calculate a buoyancy force using the Boussinesq equation from Eq. (9). To add a force in LBM, a force model has to

be picked. For all of the methods used in this thesis, the Guo forcing scheme [34] is used, which is commonly referred as the standard forcing scheme [1]. The Guo forcing scheme modifies the algorithms in two respects. First the calculated velocity, used in the equilibria of each method is updated so that $u_{\alpha, new} = u_{\alpha} + \frac{\Delta t}{2\rho} F_{\alpha}$, which places the velocity used in the equilibria at the midpoint of the timestep. Second, an extra source terms is added, which changes the collision so that, for SRT:

$$f_i(x_{\alpha} + c_{i\alpha}\Delta t, t + \Delta t) = f_i(x_{\alpha}, t) + \Delta t\omega(f_i^{eq}(x_{\alpha}, t) - f_i(x_{\alpha}, t)) + (1 - \frac{\Delta t}{2}\omega)F_i \quad (64)$$

F_i is the defined as:

$$F_i = w_i \left(\frac{c_{i\alpha}}{c_s^2} + \frac{(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})u_{\beta}}{c_s^4} \right) F_{\alpha}. \quad (65)$$

For the force, a greek letter index represents the direction of the actual force while a latin letter represents a sort of "force distribution".

3 Collision Operators

3.1 Filter Matrix method

The Filter Matrix lattice Boltzmann method (FMLB) is an attempt at making SRT more stable and accurate. The method was first introduced by Somers [6] in 1993 and then expanded to the thermal flows by Somers and Eggels [32] in 1995. Other than these two papers, the method has remained fairly unused in the LBM community, used in only a small number of publications in the 2010s by Zhuo et al. [9, 35–38] and more recently by Rohde et al. [39] and his students [10–21]. In the original papers by Somers, not much of any derivation is given, therefore, recently, in an internal memo, Rohde [33] has made an attempt at recreating the original derivation resulting in the same formulation. The main points of that derivation will be given here.

The derivation of the FMLB is based purely on arguments from Chapman-Enskog analysis. First the assumption is made that $f_i = f_i^{(0)} + f_i^{(1)}$ is sufficient for recovery of the Navier-Stokes equations. $f_i^{(0)}$ is then the equilibrium from Eq. (33), and $f_i^{(1)}$ is derived from Eq. (54):

$$w_i \rho \left(1 + \frac{c_{i\alpha} u_\alpha}{c_s^2} + \frac{(c_{i\alpha} u_\alpha)(c_{i\beta} u_\beta)}{2c_s^4} - \frac{u_\alpha^2}{2c_s^2} \right) - \frac{1}{\omega} \left(\partial_t^{(1)} f_i^{eq} + c_{i\alpha} \partial_\alpha^{(1)} f_i^{eq} \right). \quad (66)$$

This implies a new collision operator:

$$\Omega_i = -\omega(f_i - f_i^{eq}) = \partial_t^{(1)} f_i^{eq} + c_{i\alpha} \partial_\alpha^{(1)} f_i^{eq} \quad (67)$$

Where $\partial_\alpha^{(1)} = \partial_\alpha$. By taking the zeroth and first moments of Eq. (54) it can be shown that this collision operator still fulfills:

$$\begin{aligned} \sum_i \Omega_i &= 0 \\ \sum_i c_{i\alpha} \Omega_i &= 0. \end{aligned} \quad (68)$$

Using moments of Eqs. (54, 55) the time derivatives can be written in terms of spatial derivatives of macroscopic quantities:

$$\begin{aligned} \partial_t^{(1)} \rho &= -\partial_\alpha \rho u_\alpha \\ \partial_t^{(1)} \rho u_\alpha &= -c_s^2 \partial_\alpha \rho - \partial_\beta \rho u_\alpha u_\beta \\ \partial_t^{(1)} \rho u_\alpha u_\beta &= -u_\alpha c_s^2 \partial_\beta \rho - u_\beta c_s^2 \partial_\alpha \rho - \partial_\gamma (\rho u_\alpha u_\beta u_\gamma) \\ \partial_t^{(1)} \rho u_\alpha^2 &= -2u_\alpha c_s^2 \partial_\alpha \rho - 2u_\alpha \partial_\beta \rho u_\alpha u_\beta + u_\alpha^2 \partial_\alpha \rho u_\alpha. \end{aligned} \quad (69)$$

The spatial derivative in Eq. (67) can simply be found by taking the spatial derivative of the equilibrium distribution. Plugging all of these derivatives back into Eq. (67), the FMLB collision operator is found:

$$\Omega_i = \rho w_i \left(\frac{(c_{i\alpha} \partial_\alpha)(c_{i\beta} u_\beta)}{c_s^2} - \partial_\alpha u_\alpha \right) \quad (70)$$

where the argument is made that terms with $\partial \rho u^2, u \partial \rho u^2, u^2 \partial u, \partial u^3$ should be left out as they would not lead to terms in the Navier-Stokes equation. Using this new collision operator in the Chapman-Enskog analysis [33], convergence to the Navier-Stokes equation is now found

using the uncorrect relaxation rate $\bar{\omega}$. Finally, f_i can now be expressed in terms of macroscopic quantities:

$$f_i = f_i^{eq} - \frac{1}{\bar{\omega}} \Omega_i \quad (71)$$

with f_i^{eq} and Ω_i as in Eq. (33) Eq. (70) respectively.

The derivation of the FMLB algorithm now proceeds by doing staggered time-stepping:

$$f_i(x_\alpha \pm c_{i\alpha} \frac{\Delta t}{2}, t \pm \Delta t/2) = f_i(x_\alpha, t) \pm \frac{1}{2} \Delta t \Omega_i(f)(x_\alpha, t). \quad (72)$$

Using Eq. (71) this becomes:

$$f_i(x_\alpha \pm c_{i\alpha} \frac{\Delta t}{2}, t \pm \Delta t/2) = f_i^{eq}(x_\alpha, t) + (-\frac{1}{\bar{\omega}} \pm \frac{1}{2} \Delta t) \Omega_i(f)(x_\alpha, t). \quad (73)$$

Eq. (73) can be rewritten so that:

$$f_i(x_\alpha + c_{i\alpha} \frac{\Delta t}{2}, t + \frac{\Delta t}{2}) - f_i(x_\alpha - c_{i\alpha} \frac{\Delta t}{2}, t - \frac{\Delta t}{2}) = \Delta t \Omega_i(f)(x_\alpha, t) \quad (74)$$

shows exactly the same form as the midpoint rule from Eq. (46) but with a new collision operator. At this point Eq. (73) is transformed into moment space. Transformation in moment space is done by multiplying f_i by a transformation matrix M_{ki} . The rows of this matrix are defined by polynomials of $c_{i\alpha}$, for example, the first row is filled with ones and gives the density moment. The second row giving the moment relating to momentum in x direction is given by c_{ix} . Higher order moments are defined in different ways by different authors [5, 6, 8, 9, 22, 32], but for FMLB are often related to the hermite polynomials which results in very clean definitions of moments. For D2Q9, the rows of the hermite based transformation matrix are given by:

$$[1, c_{ix}, c_{iy}, c_{ix}^2 - 1/3, c_{ix}c_{iy}, c_{iy}^2 - 1/3, c_{iy}(c_{iy}^2 - 1/3), c_{ix}(c_{iy}^2 - 1/3), (c_{ix}^2 - 1/3)(c_{iy}^2 - 1/3)]^T. \quad (75)$$

Multiplying M_{ki} by Eq. (73) gives [9]:

$$m^\pm = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_x u_x + \rho(-\frac{1}{\bar{\omega}} \pm \frac{\Delta t}{2}) 2c_s^2 \frac{\partial u_x}{\partial x} \\ \rho u_x u_y + \rho(-\frac{1}{\bar{\omega}} \pm \frac{\Delta t}{2}) c_s^2 (\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) \\ \rho u_y u_y + \rho(-\frac{1}{\bar{\omega}} \pm \frac{\Delta t}{2}) 2c_s^2 \frac{\partial u_y}{\partial y} \\ T_1^\pm \\ T_2^\pm \\ F^\pm \end{bmatrix} \quad (76)$$

where $T_1^\pm, T_2^\pm$, and $F^\pm$ correspond to higher order terms which are not part of the hydrodynamic behavior, but do influence leading order errors [8]. Collision in the FMLB algorithm is now done by expressing the (+) moments in terms of the (-) moments. For example, to get the third (+) moment:

$$m_3^+ = \frac{-\frac{1}{\bar{\omega}} + \frac{\Delta t}{2}}{-\frac{1}{\bar{\omega}} - \frac{\Delta t}{2}} (m_3^- - \rho u_x u_x) + \rho u_x u_x. \quad (77)$$

The higher order moments are obtained as follows:

$$\begin{aligned}
T_1^+ &= -\bar{\gamma}_1 T_1^- \\
T_2^+ &= -\bar{\gamma}_1 T_2^- \\
F^+ &= -\bar{\gamma}_2 F^-
\end{aligned} \tag{78}$$

With $\bar{\gamma}_1$ and $\bar{\gamma}_2$ tunable relaxation parameters. The original authors [9, 32] claim that these parameters do not have a significant effect on the simulation, and therefore they are often set to 0. The derivation thus far given has been for forceless incompressible flow. Additionally, in Zhuo et al. [9] independent relaxation of bulk viscosity, and forcing are added using similar arguments. The final result is an explicit writing of m^+ in terms of m^- :

$$\begin{aligned}
m_1^+ &= m_1^- \\
m_2^+ &= m_2^- + \Delta t F_x \\
m_3^+ &= m_3^- + \Delta t F_y \\
m_4^+ &= \rho u_x u_x + G_1(m_4^- - \rho u_x u_x) + G_2(m_6^- - \rho u_y u_y) \\
m_5^+ &= \rho u_x u_y + G(m_5^- - \rho u_x u_y) \\
m_6^+ &= \rho u_y u_y + G_1(m_6^- - \rho u_y u_y) + G_2(m_4^- - \rho u_x u_x) \\
m_7^+ &= -\bar{\gamma}_1 m_7^- \\
m_8^+ &= -\bar{\gamma}_1 m_8^- \\
m_9^+ &= -\bar{\gamma}_2 m_9^-
\end{aligned} \tag{79}$$

with $G^\pm = \rho(-\frac{2c_s^2}{\omega} \pm c_s^2 \Delta t)$, $G = G^+/G_-$, $G_1 = \frac{GB_1+B_2}{B_1+B_2}$, $G_2 = \frac{(1-G)B_2}{B_1+B_2}$, $B_1 = G^- - (B-1)\rho\nu$, $B_2 = -(B-1)\rho\nu$ and $\nu_B = B\nu - \frac{\nu}{3}$. The full FMLB algorithm can now be stated. For any node, take its distribution function and multiply by M_{ki} to transform it into moments so that $f^- \rightarrow m^-$. Second, calculate the post collision from the pre collision moments using Eq. (79) to get $m^- \rightarrow m^+$. Third, calculate the post collision moments and transform them back into distribution space by multiplying by M_{ik}^{-1} which gives $m^+ \rightarrow f^+$. Finally, assign the post distribution to the nearby nodes by following their velocity, then increment the timestep by Δt .

3.2 Matrix projected Boltzmann

With the FMLB method as a starting point, in 2025 Sundström [21] developed the matrix projected Boltzmann method with the goal of having a mathematical framework, where the target equation can be seen as an input to the framework for deriving an LBM method instead of using Chapman-Enskog analysis to recover the target equation. First, the continuous Boltzmann equation, Eq. (22), is transformed into moment space by using a hermite polynomial based transformation matrix giving:

$$\begin{bmatrix} \Omega_0^m \\ \Omega_\alpha^m \\ \Omega_{\alpha\beta}^m \end{bmatrix} = \partial_t \begin{bmatrix} m_0^{eq} \\ m_\alpha^{eq} \\ m_{\alpha\beta}^{eq} \end{bmatrix} + \begin{bmatrix} \partial_\alpha m_\alpha^{eq} \\ c_s^2 \partial_\alpha m_0^{eq} + \partial_\beta m_{\alpha\beta}^{eq} \\ c_s^2 (\partial_\alpha m_{\alpha\beta}^{eq} + \partial_\beta m_\alpha^{eq}) \end{bmatrix} + \partial_t \begin{bmatrix} m_0^{neq} \\ m_\alpha^{neq} \\ m_{\alpha\beta}^{neq} \end{bmatrix} + \begin{bmatrix} \partial_\alpha m_\alpha^{neq} \\ c_s^2 \partial_\alpha m_0^{neq} + \partial_\beta m_{\alpha\beta}^{neq} \\ c_s^2 (\partial_\alpha m_{\alpha\beta}^{neq} + \partial_\beta m_\alpha^{neq}) \end{bmatrix} \tag{80}$$

with $\Omega_{\alpha\beta}^m$ the collision operator transformed into moments by using second order hermite polynomials with α and β spatial directions, similarly for Ω_0^m and Ω_α^m . m^{eq} denote equilibrium moments, and $m^{neq} = m - m^{eq}$ denote non-equilibrium moments. Note that moments of higher order than 2 are not present in this formulation. By using the Navier-Stokes equations, suitable expressions for the unknown variables Ω^m and m^{neq} are found so that:

$$\begin{aligned}
m_0^{neq} &= 0, & \Omega_0^m &= 0 \\
m_\alpha^{neq} &= 0, & \Omega_\alpha^m &= F_\alpha \\
m_{\alpha\beta}^{neq} &= -(\rho\nu(\partial_\beta u_\alpha + \partial_\alpha u_\beta - \frac{2}{3}\partial_\gamma u_\gamma \delta_{\alpha\beta}) + \nu_B \partial_\gamma u_\gamma \delta_{\alpha\beta}) \\
\Omega_{\alpha\beta}^m &= \partial_t(\rho u_\alpha u_\beta) + c_s^2(\partial_\alpha \rho u_\beta + \partial_\beta \rho u_\alpha) - \partial_t(\rho\nu(\partial_\beta u_\alpha + \partial_\alpha u_\beta - \frac{2}{3}\partial_\gamma u_\gamma \delta_{\alpha\beta}) + \nu_B \partial_\gamma u_\gamma \delta_{\alpha\beta}).
\end{aligned} \tag{81}$$

These moments and collision operators are now used in a staggered timestepping scheme. The derivation now continues by finding moments at integer timesteps to use in collision. Eventually the second order moments give:

$$m_{\alpha\beta} = \frac{2\nu}{2\nu + c_s^2 \Delta t} m_{\alpha\beta}^- + \frac{c_s^2 \Delta t}{2\nu + c_s^2 \Delta t} \rho u_\alpha u_\beta + \frac{\nu \Delta t}{2\nu + c_s^2 \Delta t} E \tag{82}$$

with $m_{\alpha\beta}^-$ moments at the previous half timestep, and E a term involving spatial and temporal derivatives which is neglected. The final D2Q9 formulation then becomes:

$$\begin{aligned}
m_0^+ &= m_0^- \\
m_\alpha^+ &= m_\alpha^- + \Delta t F_\alpha \\
m_{\alpha\beta}^+ &= \frac{2\nu - c_s^2 \Delta t}{2\nu + c_s^2 \Delta t} m_{\alpha\beta}^- + \frac{2c_s^2 \Delta t}{2\nu + c_s^2 \Delta t} \rho u_\alpha u_\beta
\end{aligned} \tag{83}$$

with $\rho = m_0^-$ and $\rho u_\alpha = m_\alpha^- + \Delta t F_\alpha/2$. Note that, with the usual $c_s^2 = 1/3$, the coefficients of the second order moments are defined so that:

$$m_{\alpha\beta}^+ = \Delta t \omega m_{\alpha\beta}^{eq} + (1 - \Delta t \omega) m_{\alpha\beta}^-. \tag{84}$$

3.3 Multiple relaxation time method

The multiple relaxation time (MRT) method is far more popular than the FMLB. MRT is the canonical "better method" than SRT, and was developed by d'Humi re [40] in 1992 and later popularized by Lallemand and Luo [5], and d'Humi re et al. [4]. The main idea of MRT is to relax different moments with potentially different rates, and to tune those relaxation rates to achieve optimal stability. The derivation of the method is much simpler than that of FMLB. First, start from Eq. (45), then transform the entire equation into moment space by premultiplying with a transformation matrix M_{ki} . For MRT, the rows of the transformation matrix are usually derived with a Gram-Schmidt process, where subsequent higher order rows are obtained by making them orthogonal to all previous rows [5]. In this thesis, a transformation matrix based on Hermite polynomials is used [1]. After transformation to moment space, the relaxation rate ω for SRT is replaced with a relaxation matrix S , which is usually defined to be diagonal with entries ω_k so that moments are relaxed independently. The MRT algorithm now takes the following form:

$$m_k(x_\alpha + c_{i\alpha} \Delta t, t + \Delta t) - m_k(x_\alpha, t) = \Delta t \omega_k (m_k^{eq}(x_\alpha, t) - m_k(x_\alpha, t)) \tag{85}$$

with m_k^{eq} the moment equilibria, usually defined by $M_{ki} f_i^{eq}$. The zeroth and first order relaxation rates are always kept at 0 in accordance with mass and momentum conservation. Only the second order relaxation rates show up at the Navier-Stokes level [8] and are related to viscosity with

the relation $\omega_{2^{nd} \text{ order}} = \frac{1}{\frac{\nu}{c_s^2} + \frac{\Delta t}{2}}$. Just like for FMLB, the bulk viscosity can be independently relaxed by using a non-diagonal relaxation matrix with:

$$\begin{aligned} m_4^* &= m_4 + \Delta t \frac{\omega_b + \omega}{2} (m_4^{eq} - m_4) + \Delta t \frac{\omega_b - \omega}{2} (m_6^{eq} - m_6) \\ m_6^* &= m_6 + \Delta t \frac{\omega_b + \omega}{2} (m_6^{eq} - m_6) + \Delta t \frac{\omega_b - \omega}{2} (m_4^{eq} - m_4) \end{aligned} \quad (86)$$

ω_b being the bulk viscosity relaxation rate and m^* denoting post-collision moments. Forces can also be added by transforming Eq. (65) into moments giving [1]:

$$M_{ki} F_i = \begin{bmatrix} 0 \\ F_x \\ F_y \\ 2F_x u_x \\ 2F_y u_y \\ F_x u_x + F_y u_y \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (87)$$

Higher order relaxation rates contribute to the stability of the method, and have to be tuned or guessed on a case by case basis [5], this is often seen as a down-side to MRT as compared to FMLB [9]. The computational procedure is similar to FMLB, first, transform the incoming distribution to moment space using M_{ki} to get $m_k(x_\alpha, t)$, then, calculate the equilibrium moments which are functions of ρ, u_α and use those in the moment space collision to get the term $m_k^*(x_\alpha, t) = m_k(x_\alpha, t) + \Delta t \omega_k (m_k^{eq}(x_\alpha, t) - m_k(x_\alpha, t))$, now transform the resulting moment back into distribution space using M_{ik}^{-1} to get $f_i^*(x_\alpha, t)$, finally stream the distributions to the nearby nodes by following their velocity directions and increment the timestep to get $f_i(x_\alpha + c_{i\alpha} \Delta t, t + \Delta t)$.

3.4 Entropic Method

The entropic method first devised by Karlin et al. [7] in 1999 aims at ensuring perfect stability by introducing a new equilibrium and collision rule. According to the second law of thermodynamics entropy should always increase in a closed system eventually reaching an equilibrium state with maximum entropy. The entropic method tries to mimic this behavior by defining an "entropy like" functional called a Lyapunov functional [41]. A Lyapunov functional maps the state of a system to a real number. If the Lyapunov functional is convex, an equilibrium can be defined as the extremum of the functional. The strategy of the entropic method is to find a Lyapunov functional so that the equilibrium satisfies the requirements from the Chapman-Enskog analysis. The collision rule can then be manipulated so that any new state always has an equal or lower value of the functional. This way the method ensures non-linear stability, with the system eventually always having to approach equilibrium.

A satisfactory Lyapunov functional is found [42] to be:

$$H = \sum_i f_i \ln(f_i/w_i). \quad (88)$$

From Eq. (88) an equilibrium can be found by minimizing H under the restriction of conservation of density and momentum. The equilibrium found is then [41]

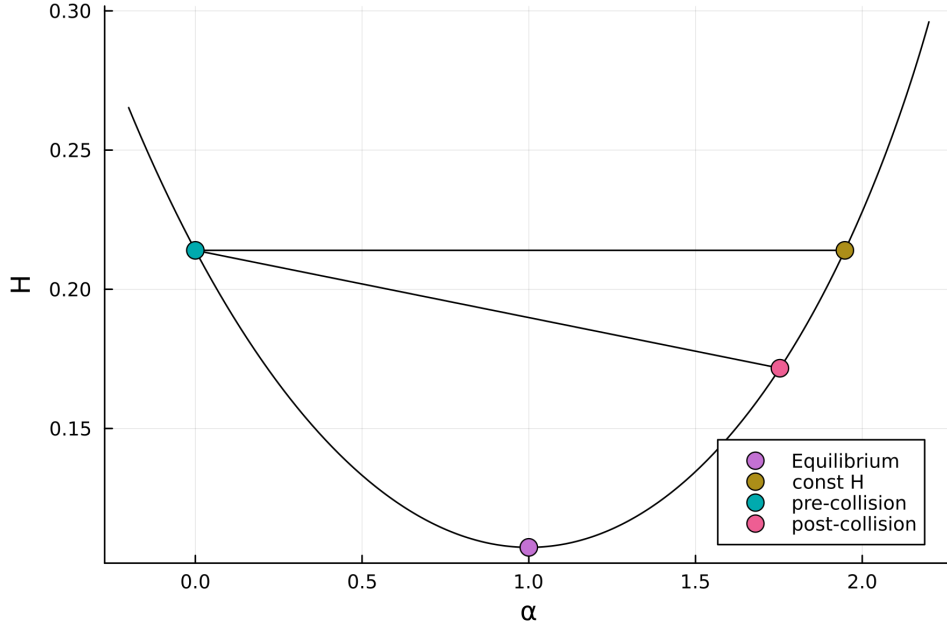


Figure 1: Relaxation in using the entropic collision operator. α is found so that the Lyapunov functional H stays constant during collision operator $f + \alpha(f^{eq} - f)$ with f the pre-collision distribution function, and f^{eq} its equilibrium. A step towards equilibrium is taken by taking a step smaller than α to ensure H always decreases

$$f_i^{eq} = w_i \rho \prod_{\alpha=1}^D (2 - \sqrt{(u_\alpha/c_s)^2 + 1}) \left(\frac{2u_\alpha + \sqrt{(u_\alpha/c_s)^2 + 1}}{1 - u_\alpha} \right)^{c_{i\alpha}} \quad (89)$$

with D the dimensions. Eq. (89) can be checked in accordance to the equilibrium requirements in Eq. (61). The next step is to make sure that the post-collision state always has the same or lower H than the pre-collision state. To this end, the collision rule is modified:

$$f_i(x_\alpha + c_{i\alpha}\Delta t, t + \Delta t) = f_i(x_\alpha, t) + \alpha \frac{\omega}{2} \Delta t (f_i^{eq}(x_\alpha, t) - f_i(x_\alpha, t)) \quad (90)$$

with α (not the spatial index) so that:

$$H(f(x_\alpha, t)) = H(f(x_\alpha, t) + \alpha \Delta t (f^{eq}(x_\alpha, t) - f(x_\alpha, t))). \quad (91)$$

This means that, if a collision is done with only α as the relaxation rate, H would stay the same. The full relaxation rate is then $\alpha \frac{\omega}{2}$, which, noting that $0 < \omega < 2$, and H being convex, is always smaller than α . The results is a collision that always reduces H . Eq. (91) is a non-linear equation which has to be solved for α at each node at each timestep in the simulation. This is a non-trivial problem, usually solved using iterative method like bisection or Newton-Raphson. An example of the entropic collision operator can be seen in Fig. 1, where α is found so that H stays constant. The post-collision distribution is then found by taking a step $\alpha \frac{\omega}{2}$ smaller than α . Through a Taylor expansion [41], it can be found that, near equilibrium, $\alpha \rightarrow 2$, which means the effective relaxation rate goes ω . Far away from equilibrium, α will be smaller than 2 which means the effect relaxation rate would be smaller than ω . From this it can be reasoned that the entropic method will add some extra numerical diffusion to avoid numerical instability. In later sections, this intuition will be confirmed. Guo forcing can also be implemented for the Entropic method very similar to SRT by changing the velocity to $u_{\alpha, new} = u_\alpha + \frac{\Delta t}{2\rho} F_\alpha$ and adding the extra source term:

$$f_i(x_\alpha + c_{i\alpha}\Delta t, t + \Delta t) = f_i(x_\alpha, t) + \Delta t \alpha (\omega/2) (f_i^{eq}(x_\alpha, t) - f_i(x_\alpha, t)) + (1 - \frac{\Delta t}{2}\omega) F_i. \quad (92)$$

The inclusion of the extra force means the post-collision distributions are not guaranteed to have a lower H than the pre-collision distributions. However, when the forces are relatively small, stability can still be maintained. In the simulations done for this thesis, no force-linked instability was ever recorded for the Entropic method.

3.4.1 Essentially entropic method

Finding α in Eq. (91) can be very expensive. To reduce this issue, Atif et al. [43] introduced the essential entropic method, which find an analytic closed form bounds on α using Padé approximations. α is now found using the root of a quadratic function $\alpha = \frac{2c}{b + \sqrt{b^2 - 4ac}}$ with a, b, c as:

$$\begin{aligned} x_i &= \frac{f_i^{eq}}{f_i} - 1 \\ a &= \sum_{i: x_i < 0} \frac{f_i x_i^3}{2} \\ b &= \sum_i \frac{f_i x_i^2}{2} \\ c &= \sum_i \frac{2f_i x_i^2}{2 + x_i}. \end{aligned} \quad (93)$$

Using this new definition, speed is improved so that the resulting algorithm achieves approximately 50% of the performance of the other methods discussed in this thesis, instead of being slower. In their paper, [43] also provide a higher order approximation to α which should reduce numerical diffusion. Finding this higher order α however, requires many more computations, and is therefore not implemented here.

3.5 Cumulant Method

The cumulant method by Geier et al. [8] from 2015 is a more recent addition to the group of collision operators. In his paper, Geier argues that moments are not independent, and better quantities to relax are actually cumulants. Using a similar technique to Chapman-Enskog analysis [44], it can be shown that MRT causes galilean invariance violating errors together with hyper-viscosity when simulating on some 3D lattices. Cumulants and their equilibria can be constructed from moments through a simple transformation and then relaxed in the same way that moments are in the MRT and FMLB methods. The notation in [8] differs significantly from many other LBM papers. In the analysis provided in [8], it is exactly this notation that helps clearly and efficiently analyzing methods. Instead of using f_i with $i \in \{1..9\}$ for D2Q9, notation is changed to f_{ij} with $i, j \in \{-1, 0, 1\}$ similar to miller indices. As an example, the SRT algorithm now changes to be written as:

$$f_{ij}(x + ic\Delta t, y + jc\Delta t, t + \Delta t) = f_{ij}(x, y, t) + \omega(f_{ij}^{eq}(x, y, t) - f_{ij}(x, y, t)) \quad (94)$$

with $c = \frac{\Delta x}{\Delta t}$ the lattice speed. "Raw" moments are now defined to be:

$$m_{ab} = c^{-a-b} \partial_{\Xi}^a \partial_{\Upsilon}^b F(\Xi, \Upsilon) \Big|_{\Xi=\Upsilon=0} = \sum_{i,j} i^a j^b f_{ij} \quad (95)$$

with $a + b$ the order of the moment and $F(\Xi, \Upsilon)$ the moment generating function attained by applying the Laplace transform to the distribution:

$$F(\Xi, \Upsilon) = \sum_{ij} f_{ij} e^{\Xi i c} e^{\Upsilon j c}. \quad (96)$$

Where, in lattice units, the lattice speed is $c = 1$. The validity of Eq. (95) can be checked by, for example, computing the $a = 1, b = 0$ moment:

$$m_{10} = \sum_{i,j} i f_{ij} e^{\Xi i} e^{\Upsilon j} \Big|_{\Xi=\Upsilon=0} = \sum_{i,j} i f_{ij}. \quad (97)$$

Cumulants are then defined by taking the logarithm of the moment generating function:

$$C_{ab} = \partial_{\Xi}^a \partial_{\Upsilon}^b \ln(F(\Xi, \Upsilon)) \Big|_{\Xi=\Upsilon=0}. \quad (98)$$

This definition can now be used to express cumulants in terms of moments, for example, the $a = 1, b = 0$ cumulant (neglecting function arguments):

$$C_{10} = \frac{1}{F} \partial_{\Xi} F \Big|_{\Xi=\Upsilon=0} = \frac{m_{10}}{m_{00}} = u_x \quad (99)$$

further, the $a = 2, b = 0$ cumulant is defined as:

$$C_{20} = -\frac{1}{F^2} (\partial_{\Xi} F)^2 + \frac{1}{F} \partial_{\Xi}^2 F \Big|_{\Xi=\Upsilon=0} = -\frac{m_{10}^2}{m_{00}^2} + \frac{m_{20}}{m_{00}} = -u_x^2 + m_{20}/\rho. \quad (100)$$

The collision algorithm is then very similar to MRT:

$$C_{ab}(x_{\alpha} + c_{i\alpha} \Delta t, t + \Delta t) - C_{ab}(x_{\alpha}, t) = \Delta t \omega_{ab} (C_{ab}^{eq}(x_{\alpha}, t) - C_{ab}(x_{\alpha}, t)). \quad (101)$$

Again, just like MRT, bulk viscosity can be relaxed independently by choosing a non-diagonal relaxation matrix. Forces can also be applied by transforming Eq. (65) into moments.

3.6 Advection diffusion

For most of the discussed methods, application to the advection-diffusion equation is obvious. For MRT and the Cumulant method, Eqs. (62 and 63) are simply transformed into moments or cumulants respectively. For the FMLB method, the thermal moments are given by Zhuo et al. [9]. For the MPB method, no thermal implementation exists so far, but Sundström did derive the mathematical framework on which to build a thermal extension. Finally, the Entropic method does not infer a simple thermal extension although a recent paper by Feng et al. [45] did derive an Entropic method for anisotropic advection-diffusion. In this thesis, when the Entropic method is used to simulate thermal flow, only the momentum transport is solved using the Entropic method, while the advection-diffusion equation is solved using the MRT method.

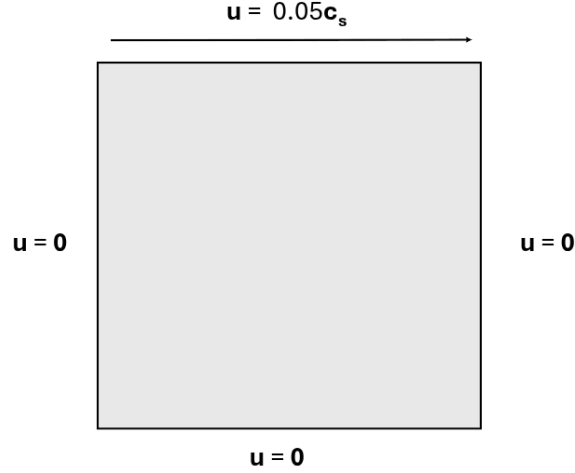


Figure 2: Schematic of lid-driven cavity flow. A square domain with no-slip side and bottom walls, and a top wall that is moving to the right at a speed of $0.05c_s$.

4 Pre-study of collision schemes

Now that the different LBM models used in this thesis have been defined, and before any rigorous comparisons to literature are performed, the models will be used to simulate two well known benchmarks, being lid-driven cavity flow and doubly periodic shear layers. The objective is to develop a qualitative understanding of the behavior of each model, and where they differ.

4.1 Lid-driven cavity flow

Lid-driven cavity is one of the canonical benchmarks for CFD [46]. The top wall is moving to the left with a velocity u_{wa} , and the rest of the walls apply a no-slip boundary condition. A schematic of the lid-driven cavity set up can be seen in Fig. 2.

4.1.1 Velocity fields

First, the lid-driven cavity is simulated on three different square grids with sidelengths of $N \in \{50, 100, 200\}$ nodes. The wall is set to move with velocity $u_{wa} = 0.05c_s$ (in lattice units, $\Delta x/\Delta t$) so that $Ma = 0.05 < 0.3$ [1]. The viscosity is chosen so that all simulations are performed at a $Re = 1000$ with $\nu = \frac{u_{wa}N}{Re}$. For the MRT, FMLB, and Cumulant method, the bulk viscosity relaxation rate is set at $\omega_b = 1.5$. The MRT and cumulant method use $\gamma_1 = 1.2$ and $\gamma_2 = 1.5$. The values for these relaxation rates are consistent with the range of values reported in literature [1, 5] and are chosen based on an initial testing phase, in which lid-driven cavity simulations are run at progressively smaller viscosities (i.e., increasing Reynolds number) until the resulting velocity field diverges to infinity or NaN. The values used throughout this thesis are those that remain stable up to the smallest viscosity tested, providing a safety margin below the instability threshold. After these tests, the relaxation rates are not changed in the rest of this thesis unless stated otherwise. The FMLB method uses $\bar{\gamma}_1 = \bar{\gamma}_2 = 0$. The results can be seen in Figs. 3, 4, 5. At $N = 50$, the SRT method diverges, such that no velocity field is recovered. It can be seen that the MRT, FMLB, and Cumulant method all yielded closely similar results in terms of velocity field, only showing some light numerical artefacts near the upper corners. The MPB method, did not diverge, however, the velocity field is clearly incorrect, with oscillating numerical artefacts at the top wall, an incorrectly orientated vortex, and vertical stripes in the middle of the domain.

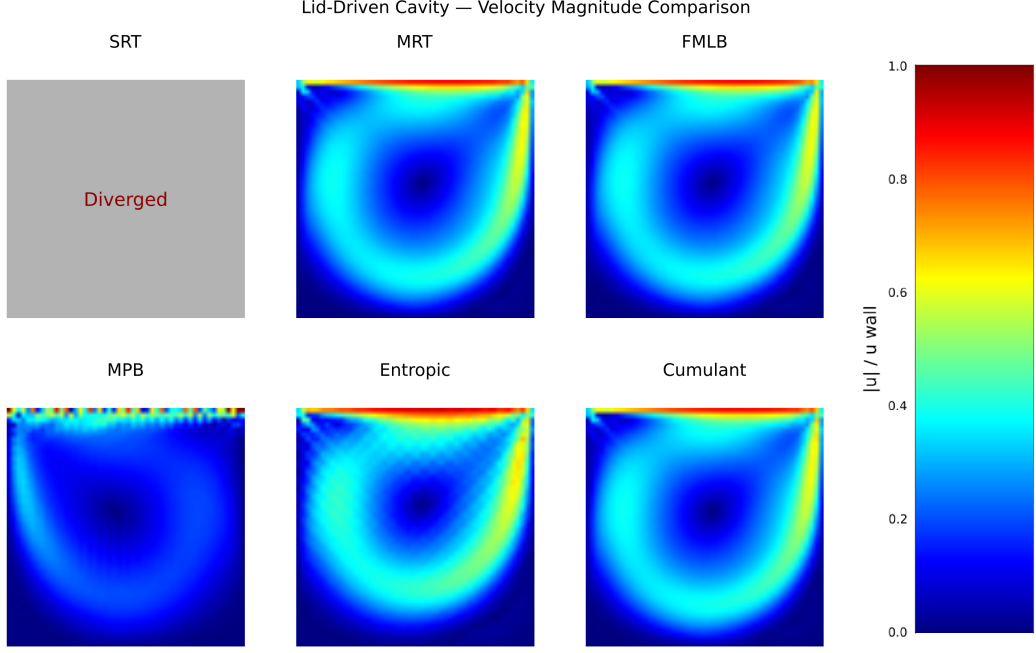


Figure 3: Lid-driven flow in a square cavity of size $N_x N_y = 50 \times 50$ at $Re = 1000$ with a viscosity of $1.4 \cdot 10^{-3}$ in lattice units using the SRT, MRT, FMLB, MPB, Entropic, and Cumulant lattice Boltzmann methods. The top wall is moving to the left with a velocity of $u_{wa} = 0.05c_s$.

The Entropic method correctly produced some of the large scale behavior of the lid-driven cavity, but the bottom of the vortex does not extend as far into the domain as the MRT, FMLB, and Cumulant method. Moreover, an unphysical checkerboard-like pattern emerges over the top half of the domain. At the next grid size of 200×200 , the MRT, FMLB, and Cumulant method still look near identical. The SRT method does not diverge, and looks very similar to the Entropic method with checkerboard-like patterns near the top of the domain. Again, the MPB method did not diverge, but does not produce correct behavior. At a grid size of 200×200 , all methods produce a very similar results. Looking closely at the top-left corner, it can be seen that SRT and the Entropic method still have some small numerical artefacts which, visually, seem to be the only difference. Interestingly, the two single relaxation rate models both show checkerboard patterns while the others do not. This suggests that these patterns are caused by higher order moments remaining in the distributions, which are otherwise filtered in the MRT, FMLB, MPB, and Cumulant method. Another observation is that the implementation of the FMLB and MPB method differ only insofar as the independent relaxation of the bulk viscosity. This suggests a connection between independent moment relaxation and model accuracy. Finally, the MRT, FMLB, and Cumulant method all show near identical results which points to a certain similarity between these methods. Zhuo et al. [9] reported similar findings when comparing FMLB and MRT. Additionally, they showed that certain choices of the higher-order relaxation rates in the MRT model can lead to incorrect flow fields in cases where the FMLB method remains accurate. Conversely, they also demonstrated that alternative relaxation-rate selections can produce the opposite outcome, with the FMLB method yielding incorrect flow fields while the MRT model predicts the correct solution.

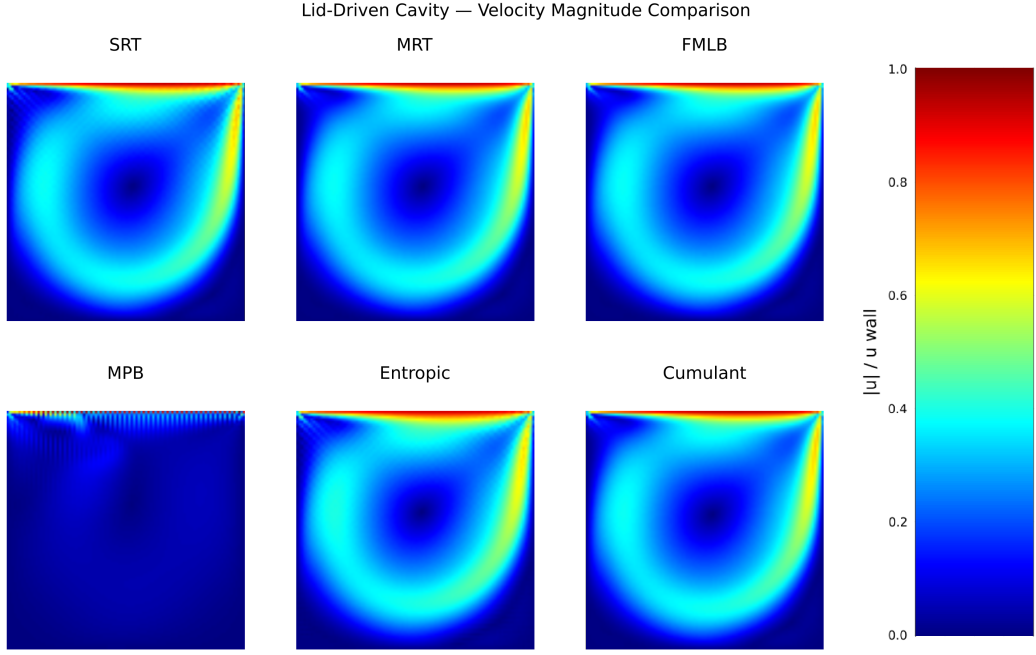


Figure 4: Lid-driven flow in a square cavity of size $N_x N_y = 100 \times 100$ at $Re = 1000$ with a viscosity of $2.8 \cdot 10^{-3}$ in lattice units using the SRT, MRT, FMLB, MPB, Entropic, and Cumulant lattice Boltzmann methods. The top wall is moving to the left with a velocity of $u_{wa} = 0.05c_s$.

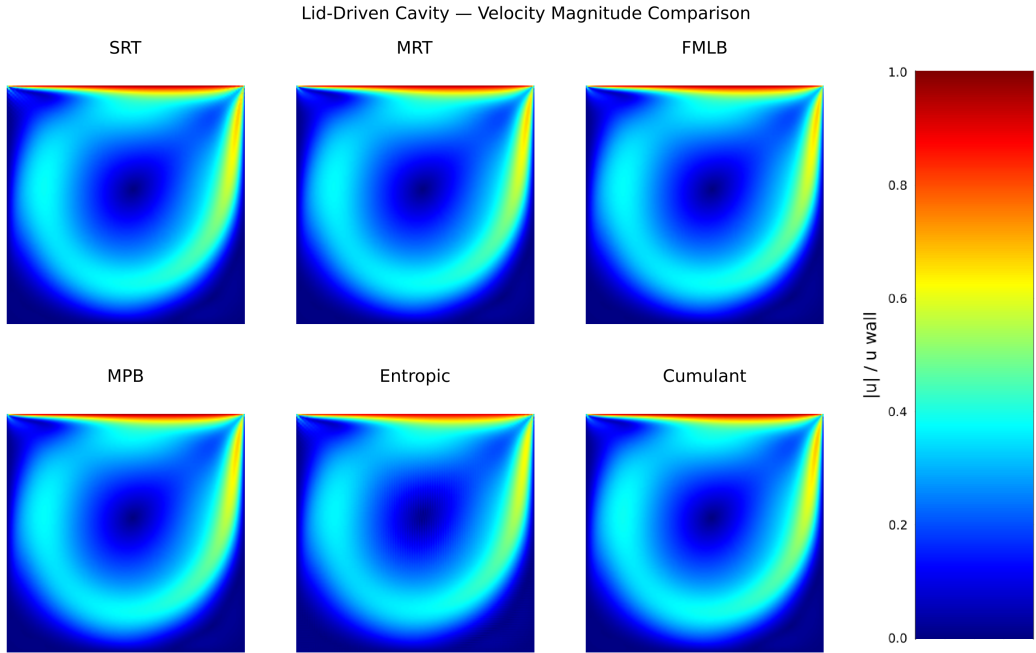


Figure 5: Lid-driven flow in a square cavity of size $N_x N_y = 200 \times 200$ at $Re = 1000$ with a viscosity of $5.8 \cdot 10^{-3}$ in lattice units using the SRT, MRT, FMLB, MPB, Entropic, and Cumulant lattice Boltzmann methods. The top wall is moving to the left with a velocity of $u_{wa} = 0.05c_s$.

4.1.2 Stability grid

The stability of the methods will be investigated by performing simulations at many different combinations of domain size and Reynolds number. The remaining settings are the same as in Sect. 4.1.1. The domain size range is $N \in [10..200]$ with Reynolds $Re \in [1..100,000]$. The Reynolds number is changed by modifying the viscosity. The simulations are done for 50,000 timesteps, recording the velocity magnitudes every 500 timesteps. If any infinite or NaN value is detected in the velocity field, the simulation is deemed unstable, and the corresponding datapoint is marked with a black cell. Fig. 6 shows the stability results for the different methods. All methods show diagonal borders between stable and unstable regions with diagonals characterized by a constant numerical viscosity. The lower right corner shows the smallest viscosities at $\nu = 1.9 \cdot 10^{-9} \frac{\Delta x^2}{\Delta t}$ and the upper left corner the largest viscosities at $\nu = 5.8 \frac{\Delta x^2}{\Delta t}$. It can immediately be noted that all methods become unstable at the smallest viscosities with the exception of the Entropic method which somehow only becomes unstable at the largest viscosities. The Entropic method is mathematically unconditionally stable, however, when lattice nodes are very close to equilibrium, catastrophic cancellation can occur in the Entropic algorithm causing divergence. The SRT method shows the worst stability, with the MPB method a close second exposing an island of stability for a smaller viscosity. The MRT and FMLB method show very similar results with seemingly only differences at the border between stable and unstable regions. For the Cumulant method, the stability border is less well-defined, however its stability appears superior to any other method (arguably with the exception of the Entropic method). Interestingly, the Cumulant method shows the opposite behavior to MPB: rather than an island of stability, it exhibits an island of instability, located at approximately the same viscosity as where the FMLB/MRT stability border lies. Tab. 1 shows the maximum and minimum numerical viscosity reached for each method evaluated at $N = 10$, and $N = 100$.

Table 1: Critical numerical viscosity ν at the stability boundary, evaluated at $N = 10$ and $N = 100$. A dash implies either no minimum or maximum viscosity is found in the tested range.

Method	$N = 10$		$N = 100$	
	$\nu_{\min} (\Delta x^2/\Delta t)$	$\nu_{\max} (\Delta x^2/\Delta t)$	$\nu_{\min} (\Delta x^2/\Delta t)$	$\nu_{\max} (\Delta x^2/\Delta t)$
SRT	2×10^{-3}	—	2×10^{-3}	—
MRT	3×10^{-4}	—	9×10^{-5}	—
FMLB	3×10^{-4}	—	9×10^{-5}	—
MPB	2×10^{-3}	—	9×10^{-4}	—
Entropic	—	—	—	9×10^{-1}
Cumulant	2×10^{-4}	—	5×10^{-5}	—

4.2 Doubly periodic shear layers

The doubly periodic shear layer consists of a fluid in a periodic box. The top half of the fluid is moving to the right with velocity u_0 , and the bottom half is moving to the left with velocity $-u_0$. A schematic of the doubly periodic shear layer setup is shown in Fig. 7. At the interface, a small perturbation is applied so that the fluid rolls up, generating two vortices. The initial conditions are given by [47]:

$$\begin{aligned}
u_x &= u_0 \tanh[k(y/H - 1/4)], & y &\leq H/2 \\
u_x &= u_0 \tanh[k(3/4 - y/H)], & y &> H/2 \\
u_y &= \delta u_0 \sin[2\pi(x/L + 1/4)]
\end{aligned} \tag{102}$$

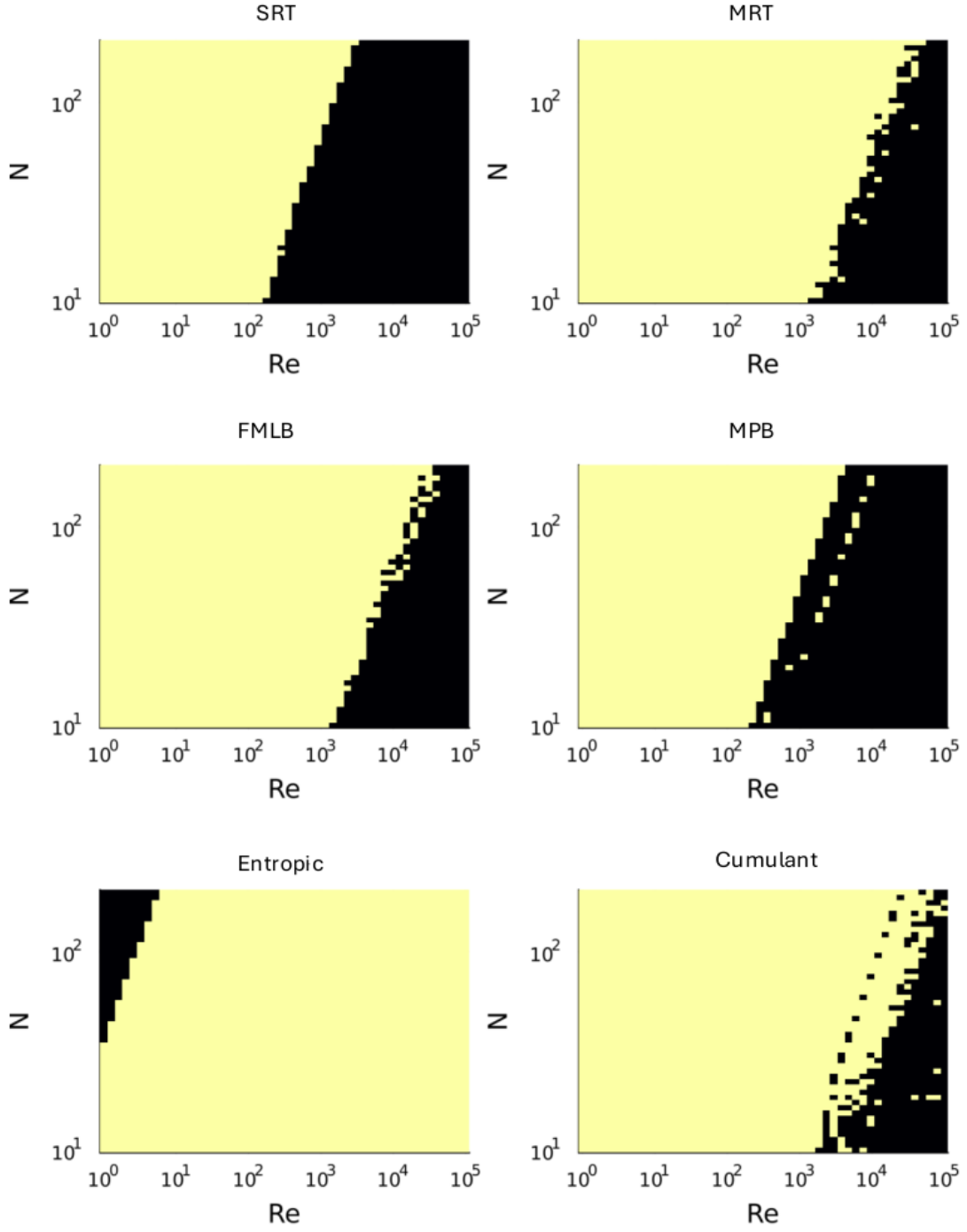


Figure 6: Stability grids for lid-driven cavity flows. The SRT, MRT, FMLB, MPB, Entropic, and Cumulant method are simulated at different grid sizes $N \times N$, and with Reynolds numbers Re with a lid speed of $u_{wa} = 0.05c_s$. The numerical viscosities range from $1.9 \cdot 10^{-9} \frac{\Delta x^2}{\Delta t}$ to $5.8 \frac{\Delta x^2}{\Delta t}$. A black cell indicates the simulation diverges due to values becoming NaN or infinite. A yellow cell indicates the simulation ran successfully without diverging.

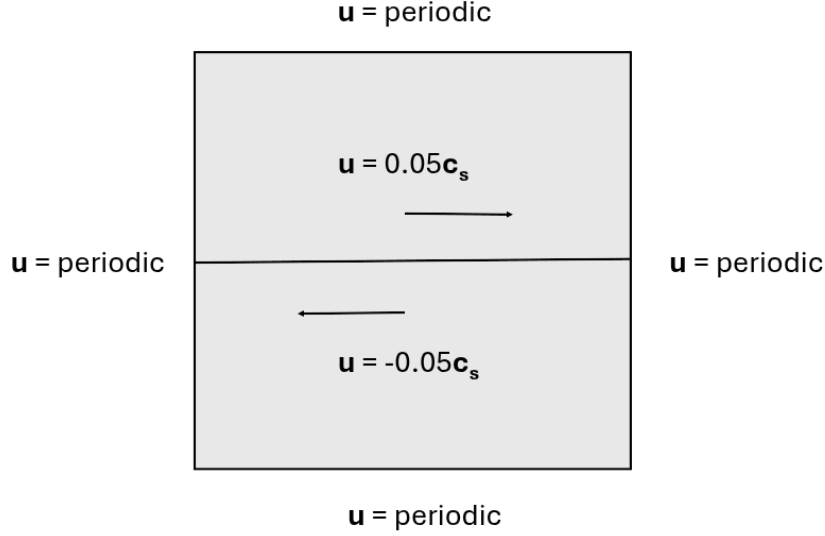


Figure 7: Schematic of a doubly periodic shear layer. A square domain with periodic boundary conditions. At the initial condition, the upper half is set at a speed of $u_0 = 0.05c_s$ moving to the right, with the lower half set at $u_0 = -0.05c_s$ moving to the left.

with k linked to the thickness of the shear layer, and δ denoting the relative strength of the perturbation. For these tests $k = 80$ and $\delta = 0.05$ is used similarly to Coreixas et al. [47] with $u_0 = 0.05c_s$ and with $Re = \frac{u_0 N}{\nu} = 10,000$. The simulations are run until the vortices reach the top and bottom respectively, and are then frozen. Fig. 8 shows a reference vorticity field for comparison found by simulating on a 2560×2560 grid using the Cumulant method. All of the methods discussed are used to simulate the shear flow on a 128×128 grid and a 256×256 grid. The vorticity field is calculated and shown in Figs. 9 and 10. With the 128×128 lattice, it can be seen that the SRT method incorrectly produces additional vortex-like patterns together with ringing patterns near the main vortex. The MRT, FMLB, MPB, and Cumulant method all produce near identical results, and only show some ringing near the main vortex. Looking at the Entropic method, it is clear that the algorithm includes extra numerical diffusion with the vorticity being attenuated, and less detail being visible in the main vortex. The 256×256 lattice shows visibly equivalent results for all methods except for the Entropic method, with now also the small ringing structures having disappeared. The Entropic method again results in softened vorticity structures, but, compared to the 128×128 case the lines are sharper. Note that the SRT method now produces the correct vortex pattern without extra vortices.

4.3 Initial conclusions

Using the SRT method as a base case, it can be seen that all other methods improve stability with the MPB method only showing slight improvements. The FMLB and MRT method both show about an order of magnitude increase in terms of stable numerical viscosity, suggesting a large effect of independent bulk viscosity relaxation. The Cumulant improves further showing an increase in stability compared to MRT and FMLB. Arguably the best stability is seen using the Entropic method, which only diverges for large viscosities, which is not a regime of practical interest.

In terms of accuracy, the MRT, FMLB, and Cumulant method all show qualitatively similar results with only some slight numerical artefacts with the low resolution shear layer flow. The MPB behaves differently sometimes showing completely incorrect results, and other times

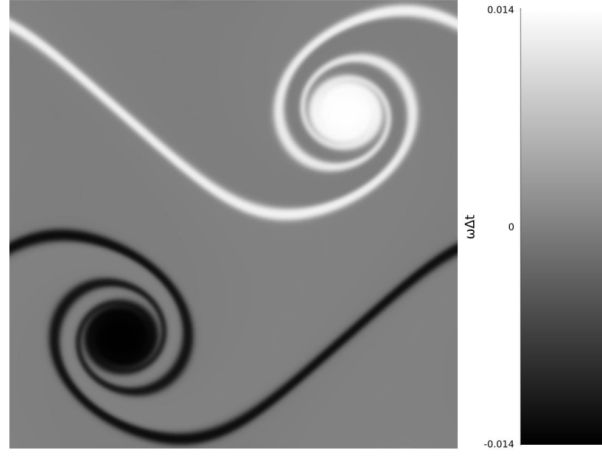


Figure 8: Vorticity in a doubly periodic shear layer simulations on a 2560x2560 grid with $Re = 10000$ and a numerical viscosity of $7.4 \times 10^{-3} \frac{\Delta x^2}{\Delta t}$.

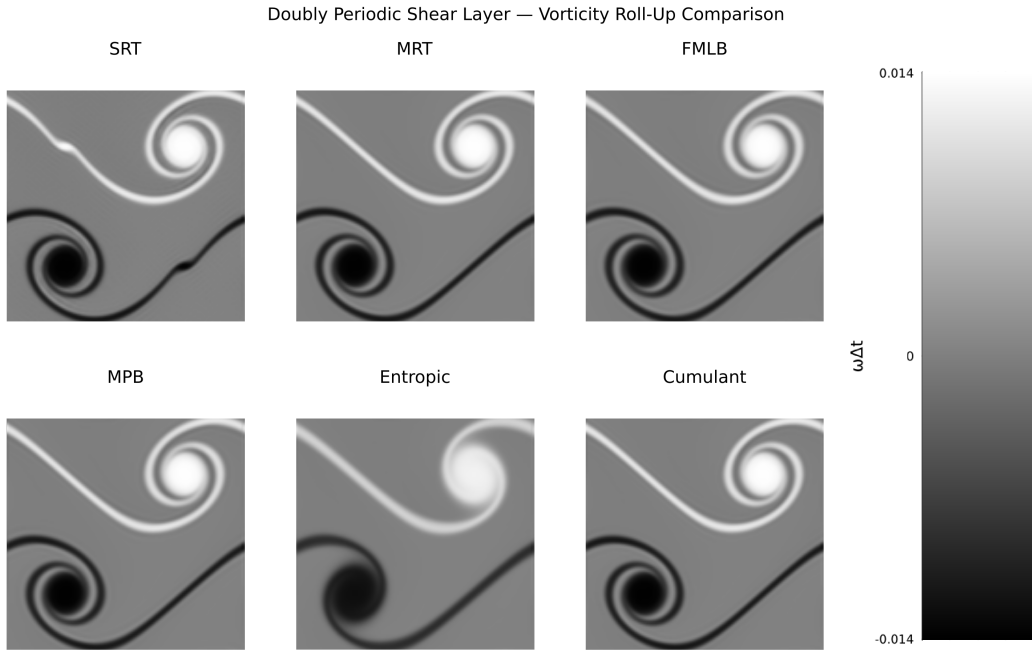


Figure 9: Vorticity in a doubly periodic shear layer simulations using different lattice Boltzmann methods on a 128x128 grid with $Re = 10000$ and a numerical viscosity of $3.7 \times 10^{-4} \frac{\Delta x^2}{\Delta t}$.

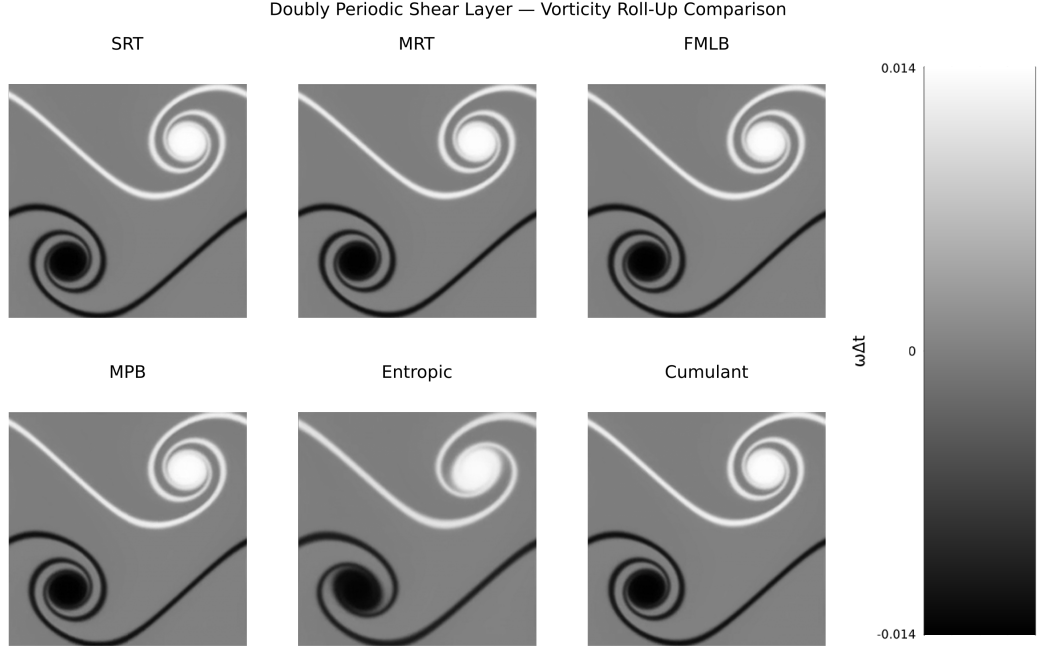


Figure 10: Vorticity in a doubly periodic shear layer simulations using different lattice Boltzmann methods on a 256×256 grid with $Re = 10000$ and a numerical viscosity of $7.4 \times 10^{-4} \frac{\Delta x^2}{\Delta t}$.

being visually equivalent to MRT, FMLB, and the Cumulant method. The Entropic method results in increased numerical diffusion and checkerboard patterns. Increasing resolution seems to somewhat alleviate this issue.

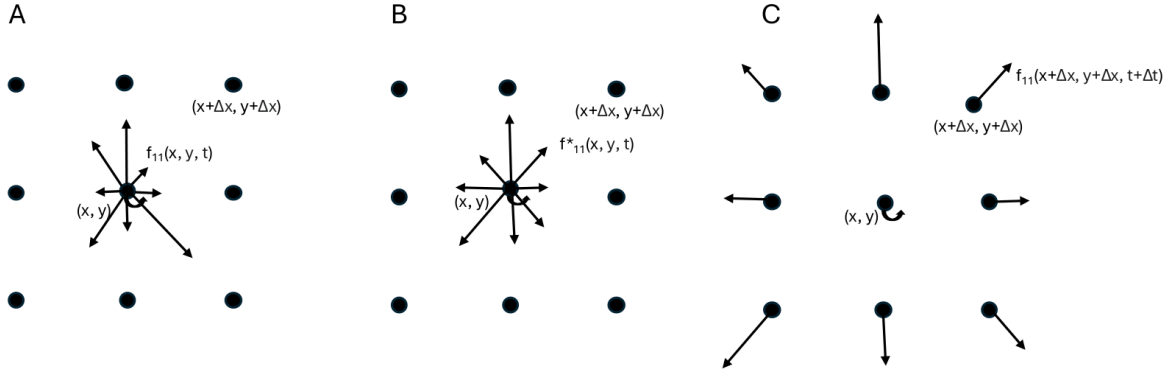


Figure 11: A schematic of a full time step in the MRT method is shown. (A) shows the incoming distributions at time t before collision. (B) displays the same distributions post-collision but before streaming. (C) shows the distributions post streaming at time $t + \Delta t$.

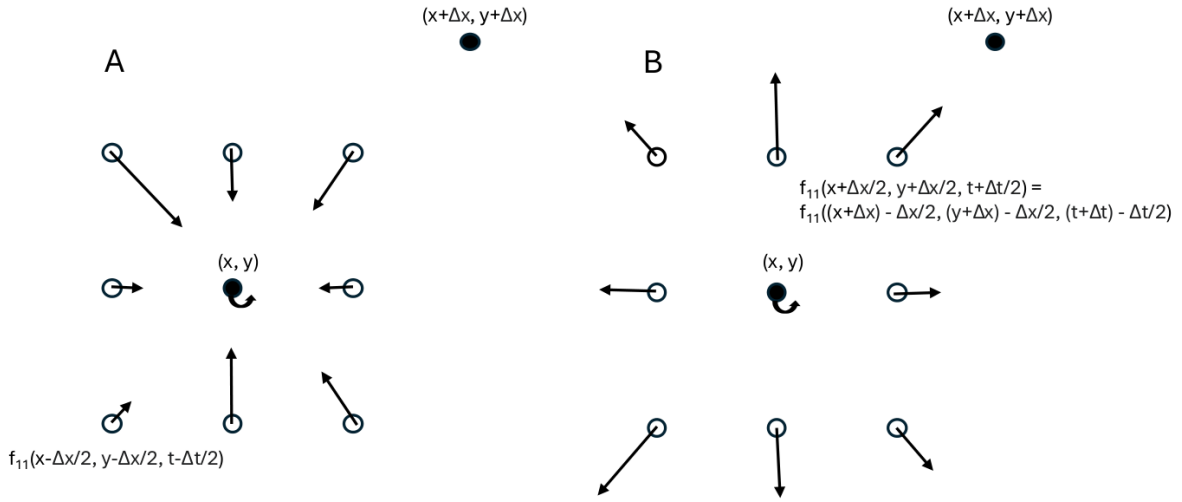


Figure 12: A schematic of a full time step in the FMLB method is shown filled dots indicate integer spatial steps, hollow dots indicate half steps. (A) shows the incoming distributions at $t - \Delta t/2$, (B) shows the outgoing distributions at $t + \Delta t/2$.

5 Filter Matrix and Multiple relaxation equivalence

At their core, both the FMLB and the MRT model start from the same idea, of performing collision in moment space instead of distribution space. While their derivation is quite different, it can be shown that both models produce the same velocity and density field. This equivalence is shown here for the first time. Equivalence is proven in two parts, first, the placing of distributions at either integer or half steps in MRT and FMLB respectively, is shown to result in the same algorithm, then

5.1 Equivalent streaming

the collision step is shown to modify distributions identically. Figs. 11 and 12 show a schematic of the time stepping algorithm of the MRT and FMLB method respectively. The arrows represent distributions, with their size indicating the amount of particles with the specific microscopic velocity. In the MRT method, the incoming distributions at a node: $f_i(x_\alpha, t)$, are modified

using the collision operator in Eq. 85 giving $f_i^*(x_\alpha, t)$ as the post collision distributions. As the final step these distributions are streamed to their neighbors with $f_i(x_\alpha + c_{i\alpha}\Delta t, t + \Delta t) = f_i^*(x_\alpha, t)$. Fig. 11 shows an example of this procedure with f_{11} the distribution moving in the North-East direction. The FMLB method condenses these operations to a single step: calculating $f_i(x_\alpha + c_{i\alpha}\Delta t/2, t + \Delta t/2)$ from $f_i(x_\alpha - c_{i\alpha}\Delta t/2, t - \Delta t/2)$. Fig. 12 shows an example. The distribution $f_{11}(x - \Delta x/2, y - \Delta x/2, t - \Delta t/2)$ is used in the FMLB collision procedure producing $f_{11}(x + \Delta x/2, y + \Delta x/2, t + \Delta t/2)$. This distribution is then interpreted as the incoming distribution for the node at $(x + \Delta x, y + \Delta x)$ at time step $t + \Delta t$. Note that even though the FMLB method defines distributions at half steps in time and space, the physical velocity and density field are still defined at the integer position [9]. This means that the physical fields are not effected by placing the distributions at either integer or half steps. Likewise, the transformation to and from moment space can be made equivalent by using the same transformation matrix in Eq. (75).

5.2 Equivalent collision

What remains to be proven is that the mapping:

$$m_k(x_\alpha - c_{i\alpha}\Delta t/2, t - \Delta t/2) \rightarrow m_k(x_\alpha + c_{i\alpha}\Delta t/2, t + \Delta t/2), \quad (103)$$

in the FMLB method is equivalent to:

$$m_k(x_\alpha, t) \rightarrow m_k(x_\alpha + c_{i\alpha}\Delta t, t + \Delta t), \quad (104)$$

using the MRT method. The proof provided here will be for the case of $\nu_B = \frac{2}{3}\rho\nu$ and no forces, with the general case being proven in A.1 and A.2. To write the MRT mapping, the equilibrium moments need to be found by transforming Eq. (33) into moment space using Eq. (75). Additionally, symmetry rules are used to find the moments of w_i . These rules can be found in Kruger et al. [1] and are given by:

$$\begin{aligned} \sum_i w_i &= 1, \\ \sum_i c_{i\alpha} w_i &= 0, \\ \sum_i c_{i\alpha} c_{i\beta} w_i &= c_s^2 \delta_{\alpha\beta}, \\ \sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} w_i &= 0, \\ \sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} w_i &= c_s^4 (\delta_{\alpha\beta} \delta_{\gamma\delta} + \delta_{\alpha\gamma} \delta_{\beta\delta} + \delta_{\alpha\delta} \delta_{\beta\gamma}), \\ \sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} c_{i\epsilon} w_i &= 0. \end{aligned} \quad (105)$$

With these rules, the equilibrium moments are given by:

$$m_k^{eq} = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_x u_x \\ \rho u_x u_y \\ \rho u_y u_y \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (106)$$

Using Eq. (85) the MRT mapping reads:

$$m_k^n = \begin{bmatrix} m_1^p \\ m_2^p \\ m_3^p \\ m_4^p + \Delta t \omega (\rho u_x u_x - m_4^p) \\ m_5^p + \Delta t \omega (\rho u_x u_y - m_5^p) \\ m_6^p + \Delta t \omega (\rho u_y u_y - m_6^p) \\ m_7^p + \Delta t \gamma_1 m_7^p \\ m_8^p + \Delta t \gamma_1 m_8^p \\ m_9^p + \Delta t \gamma_2 m_9^p \end{bmatrix}, \quad (107)$$

using the shorthand $m_k^n = m_k(x_\alpha + c_{i\alpha}\Delta t, t + \Delta t)$ and $m_k^p = m_k(x_\alpha, t)$. $\omega_4 = \omega_5 = \omega_6 = \omega$ is required for recovering the Navier-Stokes equations [5], also $\omega_7 = \omega_8 = \gamma_1$ and $\omega_9 = \gamma_2$. With ω_7 and ω_8 requiring the same relaxation rate by isotropy [5].

For the FMLB method, $m_k^+ = m_k(x_\alpha + c_{i\alpha}\Delta t/2, t + \Delta t/2)$ has to be written in terms of $m_k^- = m_k(x_\alpha - c_{i\alpha}\Delta t/2, t - \Delta t/2)$. This is done in Eq. (79). Using the above assumption on bulk viscosity, $G_1 = G$, and $G_2 = 0$ so that:

$$m_k^+ = \begin{bmatrix} m_1^- \\ m_2^- \\ m_3^- \\ \rho u_x u_x + G(m_4^- - \rho u_x u_x) \\ \rho u_x u_y + G(m_5^- - \rho u_x u_y) \\ \rho u_y u_y + G(m_6^- - \rho u_y u_y) \\ -\bar{\gamma}_1 m_7^- \\ -\bar{\gamma}_1 m_8^- \\ -\bar{\gamma}_2 m_9^- \end{bmatrix}. \quad (108)$$

Using the definition of G , this gives:

$$m_k^+ = \begin{bmatrix} m_1^- \\ m_2^- \\ m_3^- \\ \rho u_x u_x + \frac{-\frac{\nu}{c_s^2} + \frac{\Delta t}{2}}{-\frac{\nu}{c_s^2} + \frac{\Delta t}{2}} (m_4^- - \rho u_x u_x) \\ \rho u_x u_y + \frac{-\frac{\nu}{c_s^2} + \frac{\Delta t}{2}}{-\frac{\nu}{c_s^2} + \frac{\Delta t}{2}} (m_5^- - \rho u_x u_y) \\ \rho u_y u_y + \frac{-\frac{\nu}{c_s^2} + \frac{\Delta t}{2}}{-\frac{\nu}{c_s^2} + \frac{\Delta t}{2}} (m_6^- - \rho u_y u_y) \\ -\bar{\gamma}_1 m_7^- \\ -\bar{\gamma}_1 m_8^- \\ -\bar{\gamma}_2 m_9^- \end{bmatrix}, \quad (109)$$

which can finally be written as:

$$m_k^+ = \begin{bmatrix} m_1^- \\ m_2^- \\ m_3^- \\ m_4^- + \Delta t \omega (\rho u_x u_x - m_4^-) \\ m_5^- + \Delta t \omega (\rho u_x u_y - m_5^-) \\ m_6^- + \Delta t \omega (\rho u_y u_y - m_6^-) \\ -\bar{\gamma}_1 m_7^- \\ -\bar{\gamma}_1 m_8^- \\ -\bar{\gamma}_2 m_9^- \end{bmatrix}. \quad (110)$$

Eq. (107) can be compared to Eq. (110) showing that both mappings are equivalent given $\bar{\gamma}_1 = \Delta t \gamma_1 - 1$ and $\bar{\gamma}_2 = \Delta t \gamma_2 - 1$. This concludes the proof that the FMLB method is equivalent to the MRT method for a certain choice of the relaxation matrix.

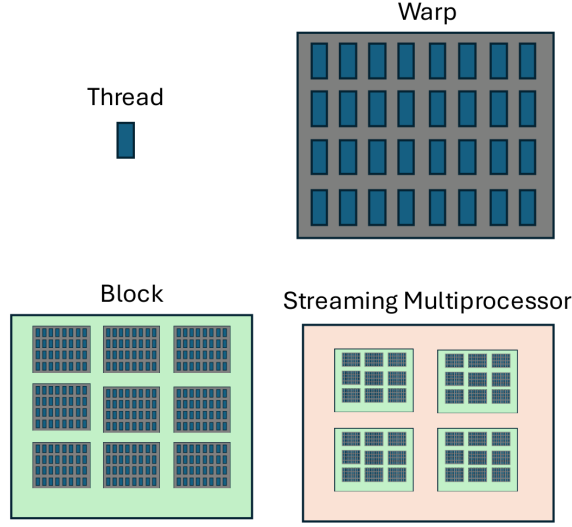


Figure 13: The hierarchy of GPU infrastructure from streaming multiprocessors operate on warps inside blocks which contain 32 threads per warp.

6 GPU optimization

All of the methods in this thesis are implemented in the Julia programming language [48]. Julia is a modern programming language focused on scientific codes. While having a relatively easy to read syntax, the language compiles to optimized binaries comparable to low level languages like C or Fortran. The Julia language has a large library of packages useful for scientific purposes. One of these packages is CUDA.jl [49] which wraps the CUDA API used for general graphics processing unit (GPU) programming. The GPU excels at handling data in parallel, which, when applicable, can improve computational efficiency by multiple orders of magnitude compared to the same algorithm running serially on a central processing unit (CPU). The infrastructure hierarchy of a GPU is shown in Fig. 13. Inside a GPU, each computational task is assigned to an individual thread. Threads are grouped into groups of 32 called a warp. Every thread inside a warp executes the same instructions at the same time. Warps are again grouped into blocks allowing some programmer defined functionality. Finally, blocks are allocated to one of a many streaming multiprocessors (SM). The SM is the one doing the actual computation, rapidly switching from warp to warp to achieve maximum performance. For example, when one warp is waiting on memory to load, the SM will quickly switch to another warp to do some other work while it waits for the previous warp to finish loading in data. GPU code runs inside so-called kernels, which can be thought of as GPU functions. A kernel is launched from the CPU specifying the number of blocks and threads per block, and is then executed by every thread in parallel. In LBM, each thread typically processes a single lattice node. Before a kernel can access any data, that data must first be allocated and transferred into the GPU memory by the CPU.

6.1 GPU architecture

LBM is well suited to GPU implementation as most of the computations can be done completely in parallel. To implement an LBM algorithm efficiently, it is important to understand how the GPU will handle data sent to it. A preliminary estimation of algorithmic efficiency can be derived by quantifying the floating-point operations, and counting the amount of bytes being moved to and from memory. A schematic of the following calculations is shown in Fig. 14. Any

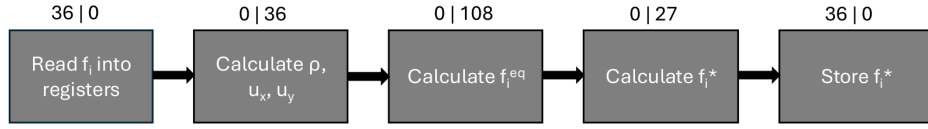


Figure 14: The fundamental computational steps of the SRT method. The numbers above the processes indicate memory moved and floating point operations performed in the format: bytes — operations. The sum of bytes moved is 72. The sum of operations performed is 171 rounded to 180 assuming a margin correlated to calculating indices for streaming.

LBM implementation will at least have to load and store all of the distributions each timestep. Using a D2Q9 lattice as reference, 9 loads and 9 stores are required. Using 32 bit floats for the distributions therefore results in moving a total of $(9 + 9) * \frac{32}{8} = 72$ bytes for each node and each timestep.

The amount of floating point operations varies quite heavily, depending on the implementation details and method used. An approximation follows: calculating ρ : 9 additions, calculating u_x, u_y : $2 * 9$ additions and $2 * 9$ multiplications = 36 operations, calculating f_i^{eq} : $9 * 4 = 36$ additions plus $9 * 8 = 72$ multiplications = 108 operations, calculating the post-collision distributions: $9 * 2 = 18$ additions and 9 multiplications = 27 operations, for a total of 180 floating point operations. The simulations in this thesis were performed on an NVIDIA RTX 3060 TI with a memory bandwidth of 448 GB/s and a peak flop rate of 16.2 TFLOPS. Based on just the memory traffic, it is possible to achieve $\frac{448 * 10^9}{72} = 6222$ million lattice updates per second (MLUPS). When taking only floating point operations into account, this comes out at $\frac{16.2 * 10^{12}}{180} = 90,000$ MLUPS. Based on this analysis it is found that the LBM algorithm is heavily memory bound, so that most of the time is spent on moving around the distributions inside GPU memory. A realistic implementation will not achieve the full 6222 MLUPS, and will only achieve a certain fraction based on the implementation details, as well as on the problem being solved. For example, a small grid will have a relatively high percentage of boundary nodes which require additional treatment, and can cause slow downs so that the resulting performance is much smaller than 6222 MLUPS. Likewise, simulations with few time steps will also achieve a reduced performance as start-up costs, like loading in constants, take up a relevant fraction of total simulation time. The cases where the highest performance is achieved, are therefore those with large grids, and many time steps, which is precisely the regime where high performance codes are most desirable.

It is essential to know where and when variables are stored in GPU memory, as these have the most impact on performance. GPU memory can be divided into five categories (a sixth, texture memory, is not used in LBM) each having different size, performance, and allowed access. From slowest to fastest:

- *Local memory* (despite its name) is as fast as global memory, and is physically in the same place as global memory called *VRAM*. Local memory is private to each thread, and acts as a spill-over when there are no more registers available in the thread. Local memory is best avoided.
- *Global memory* is the main memory in the GPU. Any thread can read and write to global memory. Global memory is the slowest type of memory, but also the largest, and is therefore ideally only used when no other locations are available, or when the size of the stored variables exceeds the capacity of other memory locations. Examples of data that should be stored in global memory are the distributions, and calculated macroscopic variables such as ρ and u_α , which should be sent back to the CPU.
- *Constant memory* is a small pool of memory which can be written to by the CPU, but

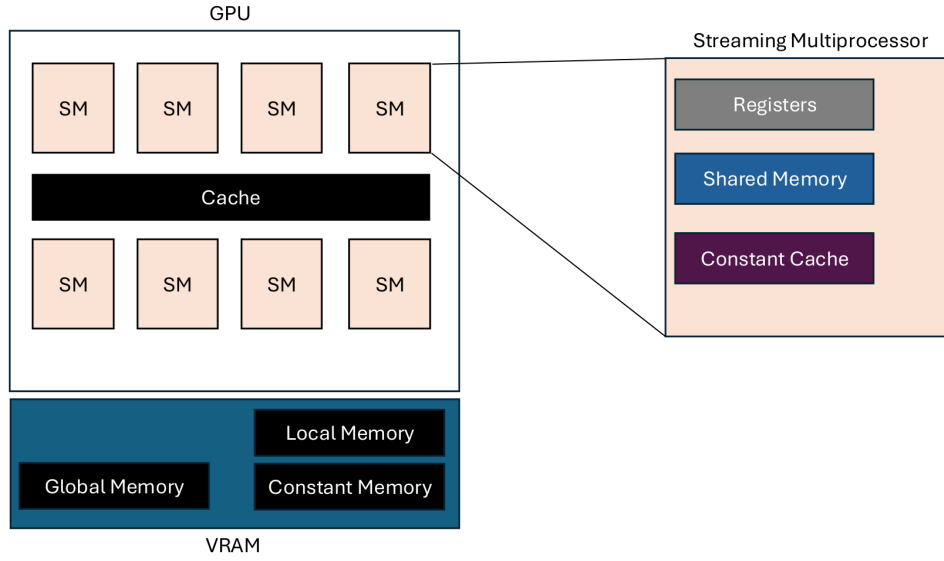


Figure 15: A diagram showing the physical location of different memory types inside a GPU. Global, local, and constant memory are physically outside the GPU, together in a space called *VRAM*. Data destined for registers or the constant cache is first moved into an on-GPU cache, then delivered to its destination inside the streaming multiprocessor. Shared memory is not populated this way, it is a scratch space explicitly written to and read from by threads within a block, rather than a cache destination for data from VRAM. Memory types not relevant to LBM simulations have been left out.

which is read-only to the GPU. Physically, constant memory sits inside global memory, but there is one big advantage. On the chip, a constant cache is present. When each thread within a warp reads the same address in constant memory simultaneously, the value is sent from the constant cache to each thread at the same time. Constant memory is very useful to LBM as small constants can be stored in it such as ω , simulation dimensions Nx, Ny , or even (more interestingly) lookup tables such as w_i , M_{ki} , and $c_{i\alpha}$.

- *Shared memory* is fast on-chip memory that is private to each block, and has the lifetime of the kernel. Shared memory is very useful when blocks can be defined in a way such that threads share data within a block, but not outside a block. Shared memory has to be allocated for each time a kernel is executed, and is therefore mainly used as a temporary scratch space. In LBM, shared memory is not as useful, as threads do not share any memory.
- *Registers* are the fastest type of memory. Registers are where the actual computations happen. Any value used in an operation always needs to be stored inside a register to do the computation. A diagram showing the physical location of these memory types is shown in Fig. 15 with Table. 2 summarizing the comparison between different memory types. Registers are also a limited resource. A thread can use a maximum of 255 registers, and a SM has a maximum of 65536 32-bit registers. Using more registers than available causes undesirable spillage to local memory. The compiler might also decide to free some registers by moving their values over to local memory to increase the number of threads well before these limits are reached.

A final consideration regarding memory access is *coalescing*. Memory moves from the global memory to its destination in cache lines of 128 contiguous bytes. When all threads in a warp request memory that is laid out inside global memory so that they fit into a single line, only a single line has to be sent. Alternatively, if the access pattern is spread across many cache

Table 2: Comparison of GPU memory types, ordered from slowest to fastest.

Memory type	Speed	Location	LBM use
Local	Slow	VRAM	Register spill-over
Global	Slowest	VRAM	f_i, ρ, u_α
Constant	Fast (cached)	VRAM + cache	$\omega, w_i, M_{ki}, c_{i\alpha}$
Shared	Fast	On-GPU	Rarely used
Registers	Fastest	On-GPU	Computation

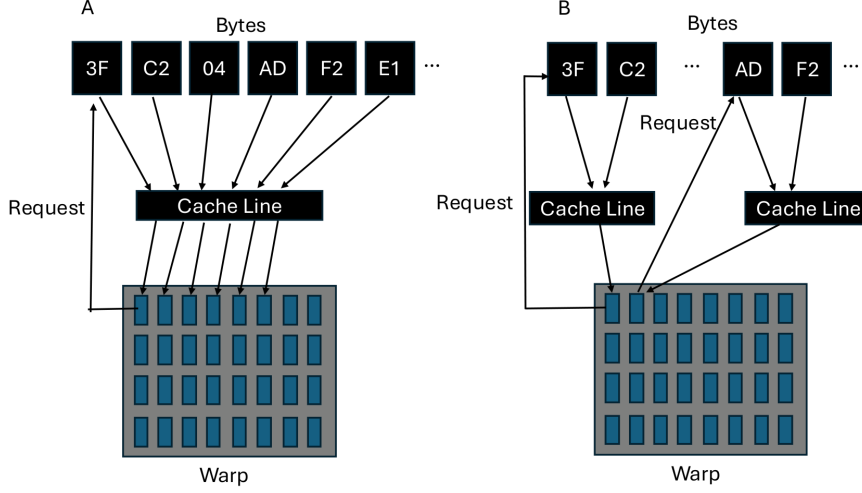


Figure 16: A schematic of cache lines in GPUs. When a thread requests the memory at a certain address, a line of 128 bytes, or 32 floats is sent starting from the requested address, and cached for quick access. If the next threads in the same warp request memory available in the cache as seen in (A), memory can be sent from the cache, avoiding memory to be moved from global memory. In (B), the second thread in the warp requires data which is not within 128 bytes of the data requested by the first thread, and was therefore not sent in the first cache line. The result is that a new cache line has to be sent from global memory.

lines, up to 32 different lines need to be moved from global memory which is much slower. A schematic of cache line behavior is shown in Fig. 16. Ensuring correct memory coalescing is vital for a fast LBM simulation.

6.2 General Implementation

With the information from the previous section, suitable decisions can be made to ensure a fast LBM code for different methods. Using all of the techniques, discussed here, a maximum efficiency of 5600 MLUPS is achieved resulting in at 90% of the theoretical maximum based on bandwidth alone. First, the choices are explained that are invariant to the LBM method used, then, method specific techniques are discussed.

6.2.1 Kernel fusing

A fast LBM code should avoid reads and writes to global memory where possible. The first and most impactful technique is kernel fusing which avoids unnecessary storing and reading operations in between collision and streaming. Fig. 17 shows a schematic of the following. A naive implementation defines two arrays in global memory: $f_{i,pre}$ and $f_{i,post}$ for the pre- and post-collision distributions respectively. The collision and streaming step are then split into two kernels, first the collision kernel reads from $f_{i,pre}$, then writes to $f_{i,post}$. Then, the streaming kernel reads from $f_{i,post}$, and writes to $f_{i,pre}$ on the neighboring nodes. This process requires

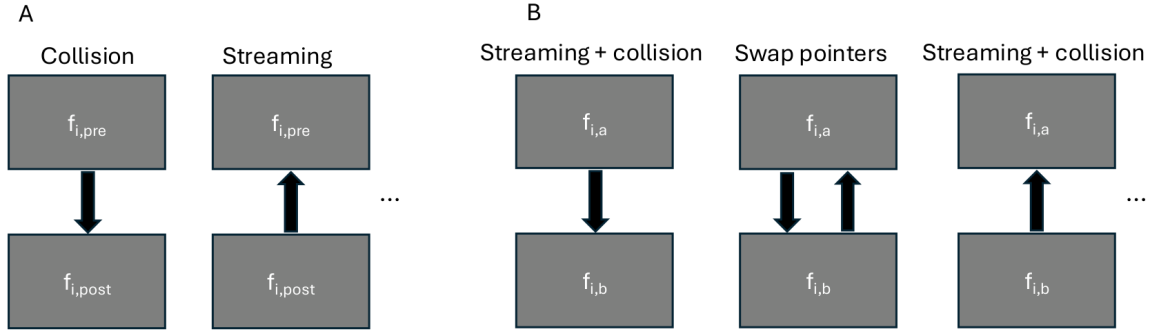


Figure 17: (A) shows an LBM implementation with split collision and streaming kernels. In the collision kernel data is read from $f_{i,pre}$, and stored in $f_{i,post}$. The streaming kernel then reads from $f_{i,post}$, and stores inside $f_{i,pre}$. In (B), data is read from $f_{i,a}$ after which streaming and collision are performed consecutively after which the resulting distributions are stored in $f_{i,b}$. The pointers for $f_{i,a}$, and $f_{i,b}$ are swapped so, in the next step, memory is read from $f_{i,b}$, and stored in $f_{i,a}$. The latter procedure reduces the amount of memory traffic by half by removing a store and load operations in between collision and streaming.

18 reads and 18 writes instead of 9 reads and writes. Fusing the kernels would mean both streaming and collision are done in the same kernel. Instead of allocating a pre- and post-collision distribution, two, more general distributions $f_{i,a}$ and $f_{i,b}$ are allocated. The order of streaming and collision are also reversed. For any lattice node, first streaming is done by reading from $f_{i,a}$ the neighboring distributions pointing to the lattice node. Now, those distributions undergo collision and are then written to $f_{i,b}$ at the lattice node. Finally, the pointers to $f_{i,a}$ and $f_{i,b}$ are swapped, and the process is repeated. Notice that $f_{i,a}$ and $f_{i,b}$ take turns in being either the pre- or post-collision distributions. This method reduces the amount of global memory traffic by half. Since LBM is memory bound, this yields a substantial performance improvement, up to twice as fast.

6.2.2 Sparse field saving

Often, macroscopic variables, like ρ and u_α , will be periodically copied from global memory to CPU memory for storage, and later, for analysis. Storing these variables into global memory each timestep is unnecessary because if they are not moved to the CPU, they will just be overwritten the next timestep. It is advantageous to implement two separate kernels, one which saves macroscopic variables, and one which does not. The saving kernel can then be launched only just before the CPU will actually read the data, with the other non-saving kernel being launched at all other timesteps. This method can save a couple of writes to global memory per node per timesteps at no extra cost compared to always saving.

6.2.3 Coalescing

The data structure used to store the distributions is very important for coalescing. If possible, requesting a value from the 32 threads inside a warp should result in only a single cache line. To achieve this, a 1D array of size $9 * Nx * Ny$ can be used to store the distributions. The distributions array should be ordered so that subsequent elements in the array correspond to the same distribution from subsequent nodes. An efficient code will then have all the threads inside a warp read the same distribution at the same time. With this method, only a single cache line is necessary to read the distribution for all threads in the warp. To efficiently interface with the 1D array, helper functions can be used that take the multiple indexed location of a node inside the simulation grid, and returns the corresponding one dimensional location of the corresponding data inside the array. This helper function usually has a form like $i + j * Nx + q * Nx * Ny$ with

i , j , and q corresponding to indices for x , y , and distribution respectively. Note that i is the fast changing index meaning that nodes horizontally next to each other will also have their data placed next to each other in the 1D array.

6.2.4 Constants

Constant memory should be used for small lookup tables like $c_{i\alpha}$, and w_i . These small arrays are the perfect use case for constant memory: they are small, read-only, and every thread inside a kernel reads from them at the same time. Putting these arrays in global memory would be unnecessarily expensive. Unfortunately, as of the time of writing, CUDA.jl does not allow directly declaring variables to be placed into constant memory. Ultimately, the CUDA compiler decides if any passed array will be put into constant memory or not. To check if any method works, the code can be compiled to SASS, which is the CUDA equivalent CPU assembly. If the compiler decided to put the arrays in constant memory, the SASS code will indicate this. A somewhat reliable method used in this thesis is to put the arrays into the static arrays from StaticArrays.jl.

6.2.5 Loop-unrolling

LBM algorithms often use sums over all of the distributions of a lattice node. In code, this sum can be expressed as a loop from 1 to Q , with Q being the number of distributions in the scheme (or from 0 to $Q-1$ if the language is 0-indexed). The compiler should know what the value of Q is at compile time. This allows for a process called loop unrolling [50] which reduces the number of computations required (like updating a counter). The easiest way to ensure Q is known at compile time is to sacrifice a little bit of generalizability, and just write the actual value of Q in the bounds of the loop instead.

Finally, switching from 64-bit to 32-bit floating point numbers substantially increases performance by halving the memory traffic, but reduces floating point precision.

6.3 Method specific implementation

6.3.1 Multiple relaxation time

The MRT method uses many more calculations than the SRT method. As an estimate, add the computations from the matrix multiplications with M_{ki} and M_{ik}^{-1} totaling $2*9*9*2 = 324$ floating point operations. The total cost is now approximately $180 + 324 = 504$ operations instead of 180 for SRT. With this new maximum speed, based on only operations equals $\frac{16.2*10^{12}}{504} \approx 32000$ MLUPS. Since the MRT does not require any extra movement of data from or to global memory, it can be seen that MRT is still memory bound. Although MRT requires approximately 2.8 times more floating-point operations than SRT, both methods are expected to run at comparable speed, as their performance is limited by the same memory bandwidth rather than by arithmetic throughput. In terms of memory management, some care is required. The transformation matrices total at 162 floats. Putting all of those floats into registers would reduce the amount of threads per SM, reducing efficiency. A nice way to get around this, is by baking the matrix coefficients directly into the instruction stream. This way, the matrices do not need to be stored in data memory, and are therefore essentially free. Again, the compiler will ultimately decide how to handle the matrices, but CUDA.jl does provide tools to check the number of registers used by a kernel. If the number of registers used in the MRT kernel is not much bigger than for SRT, this confirms the coefficients were successfully inlined. In the code for this thesis, this was achieved by putting the matrices into static arrays. Using these methods, the MRT kernel was only 2% slower than the SRT kernel.

6.3.2 Cumulant method

The cumulant method uses even more operations than the MRT method. In addition to a transformation to and from moments, there is also a transformation from moments to cumulants and back. A quick tally gives an extra ~ 70 operations on top of those in the MRT kernel. Using the same calculations as before this gives a maximum speed, based on only operations, of ≈ 28000 MLUPS. Just like for MRT, the cost of running the kernel is dominated by moving memory, meaning the cumulant method is only slightly slower than MRT.

6.3.3 Entropic method

The entropic method is the slowest of any of the methods used in this paper. Solving the non-linear equation for α is usually done through an iteration method, so that the exact amount of iterations are not known a priori. Using the upper bound from the essentially entropic method, the speed is greatly increased since α can be calculated with the iteration process. Still, the essentially entropic method requires calculating x, a, b , and c from Eq. (93) using sums over the distributions. Of special interest is the sum used to attain a , as this is a sum only over the i so that $x_i < 0$. This means that potentially, different lattice nodes within a warp will have different i to sum over causing warp divergence meaning all threads within the warp can not execute the same instructions causing slow downs. The entropic method implemented in this thesis is about 50% slower compared to SRT.

6.4 Thermal implementation

All of the above recommendations also apply for solving the temperature field. When solving for both the momentum and temperature at the same time using a double distribution function, and the Boussinesq approximation, data is interchanged between the momentum and the temperature solver. The momentum solver required knowing the temperature at each node, so that the bouyancy force can be calculated. The temperature solver required knowing the velocity at each node, so that the temperature can get advected. In principle, the momentum and temperature solver are very similar, so that the expected speed is approximately half.

Just like for collision and streaming, it might be tempting to fuse the kernels, updating both of momentum and temperature in the same kernel. This way, one would avoid having to store u_α or T in global memory for each node. In testing, this method proved to be slow, giving 1000 MLUPS when using SRT to solve both fields instead of the expected ~ 2800 MLUPS. The reason for this slowdown can be attributed to register usage. The amount of registers used in a fused momentum and temperature kernel causes a lower amount of warps to be active at the same time essentially reducing parallelism. Alternatively, the compiler might decide to store memory into local memory instead of registers, which are much slower. The solution to this problem is to split the kernels in 2. One kernel solves momentum, and one kernel solves temperature. The only extra step, is that u_α and T now do have to be stored in global memory to be accessible by both kernels. With this method, 2 new 32 bit numbers are loaded and stored in memory (16 bytes in total) for each lattice node, for each timestep. A new memory based maximum speed can be calculated as $\frac{448 \cdot 10^9}{2 \cdot 72 + 16} = 2800$ MLUPS. The actual reached speed comes in at about 2500 MLUPS, or again, about 90% of the memory based maximum.

7 Results

The models implemented in this thesis are now compared to known datasets from literature, analytical solutions, or empirical relations. The results for this thesis consists of 3 parts. First the core momentum solvers are tested on a Taylor-Green vortex benchmark. This benchmark has a known analytical solution to compare against which allows for direct measurements of model correctness. The second case consists of the well-known lid-driven cavity flow in a square domain. Contrary to Taylor-green vortices, lid-driven cavity flow has solid and moving wall boundary conditions. This benchmark can then be used to test boundary condition correctness and stability. The different models are compared to data by Ghia et al. [46]. The final case is given by Rayleigh-Bénard convection, which tests the coupling between momentum and temperature solvers by computing Nusselt numbers resulting from convection cells. The resulting Nusselt numbers are then compared to empirical relations from literature.

7.1 Taylor-Green vortex

The Taylor-Green vortex [51] is one of the few cases where an exact analytical solution to the Navier-Stokes equations is available. The Taylor-Green vortex consists of a sinusoidal initial condition together with periodic boundary conditions, where the velocity in every point decreases exponentially due to friction. The exact solution is given by:

$$\begin{aligned} u_x(x, y, t) &= u_0 \sin(kx) \cos(ky) e^{-2\nu k^2 t} \\ u_y(x, y, t) &= -u_0 \cos(kx) \sin(ky) e^{-2\nu k^2 t} \end{aligned} \quad (111)$$

with initial conditions:

$$\begin{aligned} u_x(x, y) &= u_0 \sin(kx) \cos(ky) \\ u_y(x, y) &= -u_0 \cos(kx) \sin(ky) \\ \rho(x, y) &= \rho_0 + \frac{\rho_0 u_0^2}{4} (\cos(2kx) + \cos(2ky)) \end{aligned} \quad (112)$$

and u_0 a reference velocity, $k = \frac{2\pi}{L}$ the wavelength, domainsize L , and ρ_0 the background density. The initial condition can be seen in Fig. 18 where the velocity magnitude is shown.

LBM simulations are now run with Eqs. (112) as initial conditions using the SRT, MRT, Entropic, and Cumulant model. All of the simulations are done on a 400x400 grid representing a 1x1 meter domain for 200 seconds of physical time using $u_0 = 1$ m/s with a Reynolds number of $Re = \frac{u_0 L}{\nu} = 1000$, a viscosity of $\nu = 0.011 \frac{\Delta x^2}{\Delta t}$, and at a numerical Mach number of $Ma = \frac{u_0 \frac{\Delta t}{c_s}}{\Delta x} = 0.05$. The relaxation rates used for the MRT and Cumulant method are $\gamma_1 = 1.2$, $\gamma_2 = 1.5$, and $\omega_b = 1.5$ consistent with Krüger et al. [1] and Lallemand et al. [5]. The velocity magnitudes are measured throughout the simulation and then compared to the analytical solution of Eq. (111). The RMS error is calculate using:

$$RMS_{err} = \sqrt{\frac{1}{N_x N_y} \sum_{x,y} (|\mathbf{u}_{ref}| - |\mathbf{u}_{sim}|)^2}, \quad (113)$$

for each time step with $\mathbf{u}_{ref}$ and $\mathbf{u}_{sim}$ the reference and simulated velocity field respectively. The resulting root mean square (RMS) error over time can be seen in Fig. 19. The MRT and Cumulant model both reach the correct final velocity field. The SRT model shows larger errors, with, at its peak, a root mean squared error of 0.6% of the reference velocity. After long enough

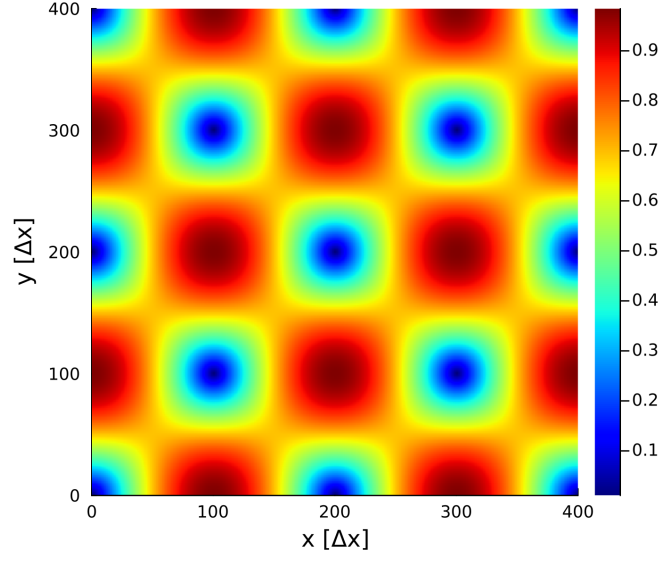


Figure 18: The initial condition of the Taylor-Green vortex showing velocity magnitude. Red indicates large velocities, blue indicates small velocities.

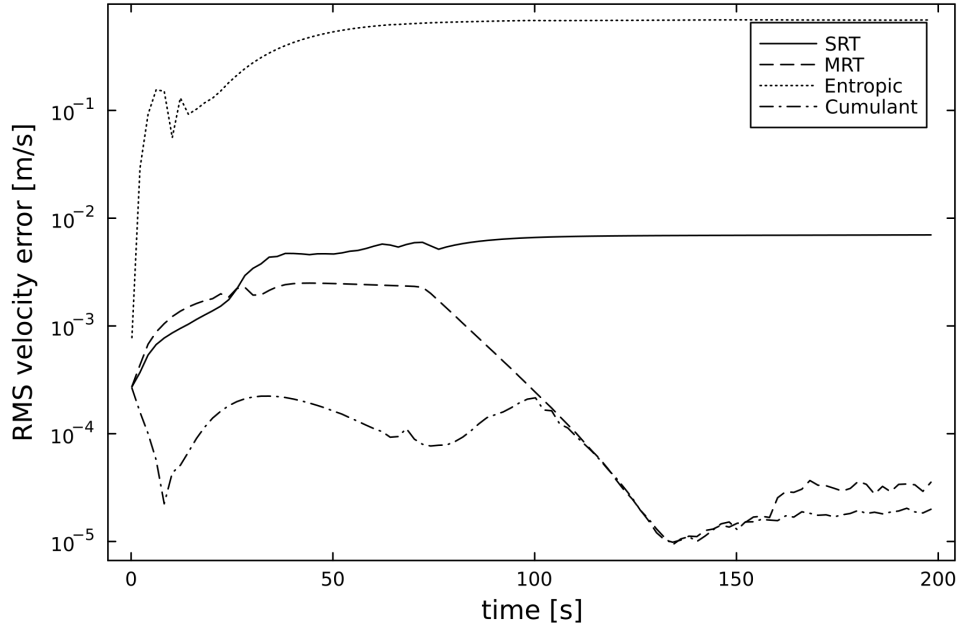


Figure 19: Root mean square error between the analytical solution of a Taylor-Green vortex, and simulations done with the single relaxation time, multiple relaxation time, the entropic, and the cumulant lattice Boltzmann method at a Reynolds number of 1000, a viscosity of $\nu = 0.11 \frac{\Delta x^2}{\Delta t}$, and a Mach number of 0.05.

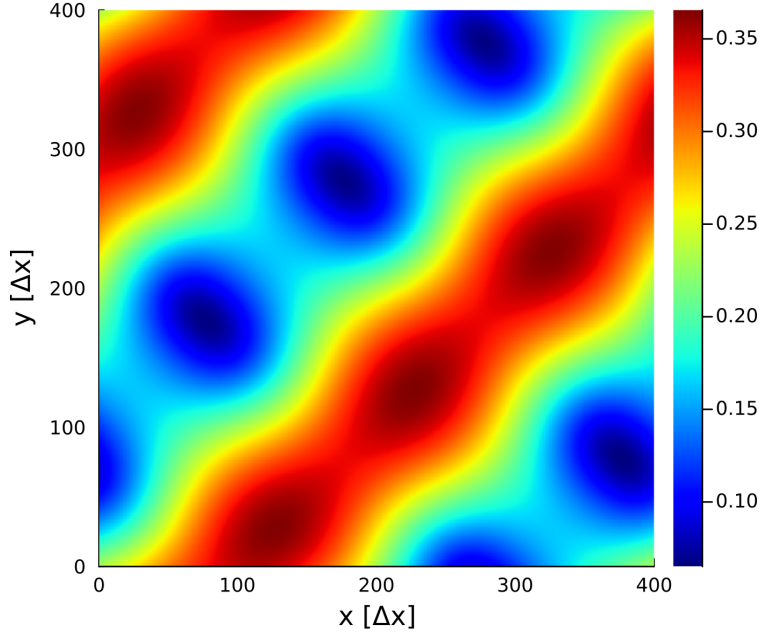


Figure 20: Taylor-Green vortex simulation using the Entropic lattice boltzmann method after 20 seconds of physical time. Instead of pure dissipation, the whole velocity field moves, and vortices are stretched which is unphysical.

time, the velocity field in the reference solution goes to zero as energy is dissipated. The SRT model does not converge to a zero-velocity field, instead a uniform velocity field is reached. The magnitude of the converged uniform velocities using the SRT method can be found using Eq. (113) where $\mathbf{u}_{ref} = 0$, and $\mathbf{u}_{sim} = const$ which results in $\mathbf{u}_{sim} = 0.06$. The errors in the Cumulant method can be seen to be the smallest throughout the entire simulation length. Just like for the SRT method, the Entropic method never converges to the correct, fully dissipated state. Instead, a state of uniform velocity of ≈ 0.6 m/s is reached. The correct Taylor-Green vortex solution shows no movement of the velocities, but purely dissipation. When looking at the simulation for the Entropic method, it can be seen that the whole velocity field starts to move, stretching the vortices along the way. This seems to be a general trend with the Entropic method when periodic boundaries are present. An example is seen in Fig. 20 showing the velocity field frozen at 20 seconds of physical time using the Entropic method.

7.2 Lid-driven cavity

Lid-driven cavity flow provides perhaps the most well known benchmark for Navier-Stokes solvers. The setup consists of a square domain with solid boundaries, with the top wall moving to the right with velocity U . A schematic of the Lid-driven cavity setup is shown in Fig. 2. The inclusion of no-slip and set velocity boundary conditions provide an extra layer of difficulty for the LBM methods. In this thesis, all of the methods use the half-way bounce back rule [1] for simulating the boundaries given by Eq. (114).

$$f_{\bar{i}}(x_{\alpha}, t + \Delta t) = f_i^*(x_{\alpha}, t) - 2w_i\rho\frac{c_{i\beta}U_{\beta}}{c_s^2}, \quad (114)$$

with $\bar{i}$ the velocity direction opposite to i . The initial conditions for lid-driven cavity flows are set by putting distributions to their equilibrium distribution with $\vec{u} = \vec{0}\frac{\Delta x}{\Delta t}$, and $\rho = 1\frac{kg}{m^3}$. The dataset of Ghia et al. [46] provides reference data that are commonly used for comparison in

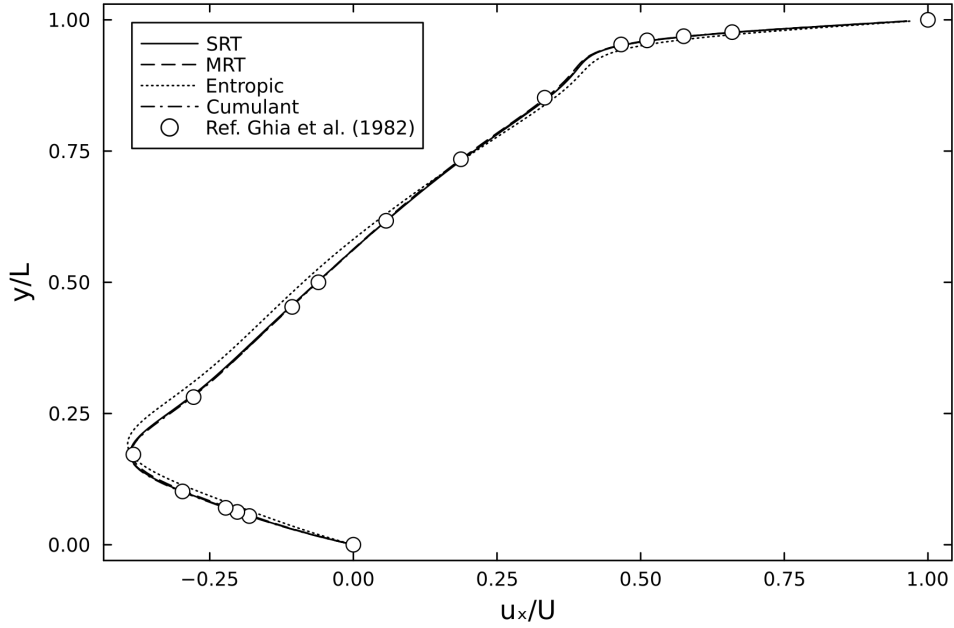


Figure 21: Horizontal center line velocity u_x of a lid-driven cavity flow problem at $Re = 1000$, numerical viscosity of $\nu = 5.8^{-3} \frac{\Delta x^2}{\Delta t}$, and lid velocity of $U = 0.05c_s$ using different LBM models on a 200x200 grid.

literature [9, 52]. Here, u_x on the vertical centerline is obtained in the steady state solution, and compared to the reference at a range of Reynolds numbers $Re \in \{1000, 5000\}$. The simulations are performed at a numerical Mach number of 0.05 with a numerical wall speed of $U = 0.05c_s$ starting with a grid of 200x200. The free input parameters of the three cases shown in the figures below are given in Table. 3. The centerlines for $Re = 1000$ are shown in Fig. 21. At $Re = 1000$, all methods show excellent agreement with the reference data, with no discernible difference between SRT, MRT, and the Cumulant method. At $Re = 5000$ in Fig. 22, again the MRT and Cumulant method agree well with the reference data. The SRT method is not present in Fig. 22 as its velocity field diverged. The Entropic method overshoots velocity near the edges of the domain. Increasing the resolution to a 400x400 grid as seen in Fig. 23, the Entropic method can now be seen to attain a closer fit to the reference. Using a 400x400 grid the SRT method becomes stable again, and shows agreement with the reference.

Table 3: Free input parameters for lid-driven cavity simulations using the SRT, MRT, Entropic, and Cumulant lattice Boltzmann method.

Grid size	Reynolds number	Viscosity $[\frac{\Delta x^2}{\Delta t}]$	Viscosity $[\frac{m^2}{s}]$
200x200	1000	5.8×10^{-3}	1.0×10^{-3}
200x200	5000	1.2×10^{-3}	2.0×10^{-4}
400x400	5000	2.3×10^{-3}	2.0×10^{-4}

7.3 Rayleigh-Bénard convection

Rayleigh-Bénard convection is used to compare thermal coupling of the different methods. The setup consists of a rectangular domain twice as wide as it is tall. The bottom and top wall are set to temperature T_h and T_c respectively. For the momentum solvers, the bottom and top wall correspond to no-slip boundary conditions. The side walls are periodic both for the momentum and temperature solver. The aspect ratio is given by $\Gamma = L/H = N_x/N_y = 2$. The initial

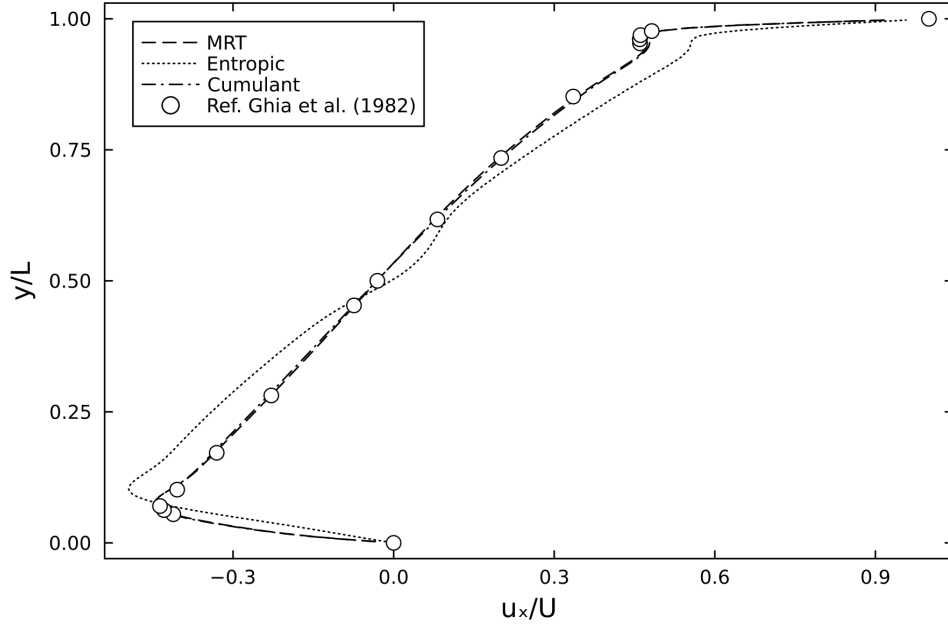


Figure 22: Horizontal center line velocity u_x of a lid-driven cavity flow problem at $Re = 5000$, numerical viscosity of $\nu = 1.2^{-3} \frac{\Delta x^2}{\Delta t}$, and lid velocity of $U = 0.05c_s$ using different LBM models on a 200x200 grid. The SRT method is not present in this figure as the velocity field diverged.

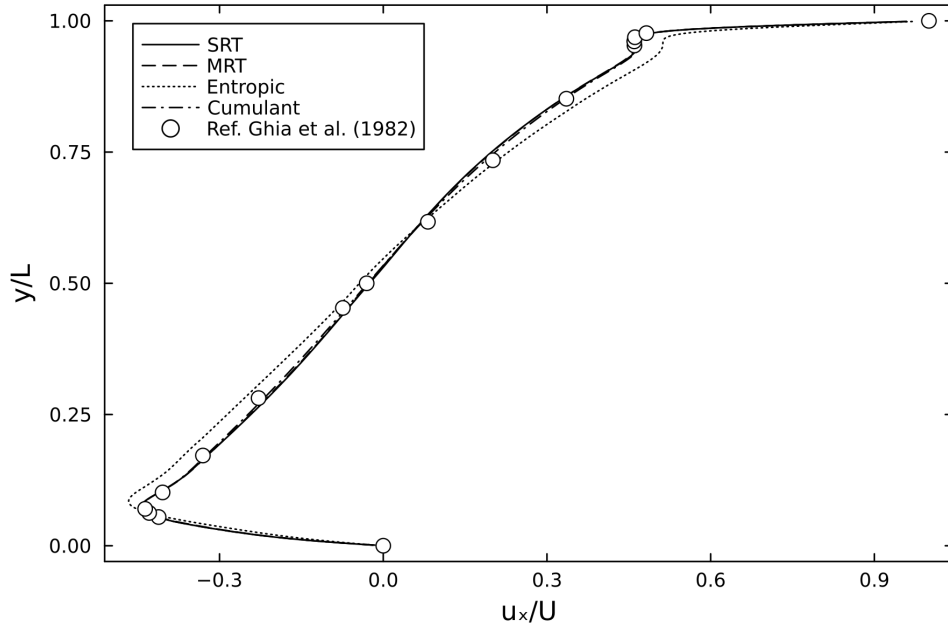


Figure 23: Horizontal center line velocity u_x of a lid-driven cavity flow problem at $Re = 5000$, numerical viscosity of $\nu = 2.3^{-3} \frac{\Delta x^2}{\Delta t}$, and lid velocity of $U = 0.05c_s$ using different LBM model on a 400x400 grids.



Figure 24: Initial and boundary conditions for Rayleigh-Bénard convection with initial temperature $T_0 = (T_c + T_h)/2$, and with a randomly perturbed initial velocity field using $u_\alpha(x_\beta, t = 0) = 0.1(R_\alpha - 0.5)\frac{\Delta x}{\Delta t}$.

temperature is set at $T_0 = (T_h + T_c)/2$. The initial velocity is perturbed triggering the growth of unstable modes helping convection cells forming by setting:

$$u_\alpha(x_\beta, t = 0) = 0.1(R_\alpha - 0.5)\frac{\Delta x}{\Delta t}, \quad (115)$$

with R_α a random number in the interval $[0, 1]$. A schematic of the initial and boundary conditions is shown in Fig. 24. The simulation grid has size 600x300, with a numerical Mach number of 0.05 based on the free-fall velocity [53] given by:

$$U_{ff} = \sqrt{\beta g \Delta T H}. \quad (116)$$

Thermal coupling is achieved using Eq. (9) with $g = -9.8 \frac{m}{s^2} = -28.3 \frac{\Delta x}{\Delta t^2}$, a thermal expansion coefficient of $\beta = 2.04 \frac{1}{K}$, and temperatures $T_c = 0$ K, $T_h = 1$ K, $T_0 = 0.5$ K. Note that β and $T - T_0$ are only used as a product with each other such that the resulting product is dimensionless. For each model, Nusselt numbers are calculated for $Ra \in \{10^3, 10^4, 10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}\}$ with $Pr = 1$. The Rayleigh numbers are varied by changing both ν and κ at the same rate to get the next value of Ra . An overview of the free input parameters for all the performed simulations is shown in Table. 4. The measurement process for the Nusselt numbers is as follows: After the

Table 4: Free input parameters of Rayleigh-Bénard convection simulations.

Rayleigh number	Viscosity = Thermal Diffusivity $[\frac{\Delta x^2}{\Delta t}]$	Viscosity = Thermal Diffusivity $[\frac{m^2}{s}]$
10^3	2.7×10^{-1}	1.6×10^{-4}
10^4	8.7×10^{-2}	5.0×10^{-5}
10^5	2.7×10^{-2}	1.6×10^{-5}
10^6	8.7×10^{-3}	5.0×10^{-6}
10^7	2.7×10^{-3}	1.6×10^{-6}
10^8	8.7×10^{-4}	5.0×10^{-7}
10^9	2.7×10^{-4}	1.6×10^{-7}
10^{10}	8.7×10^{-5}	1.6×10^{-8}

temperature field has visually settled into a steady formation, every 100 timesteps, the average Nusselt number is calculated using a discretization of Eq. (21):

$$Nu = 1 + \frac{H}{\kappa(T_h - T_c)} \frac{1}{N_x N_y} \sum_{x,y} u_y(x,y) T(x,y), \quad (117)$$

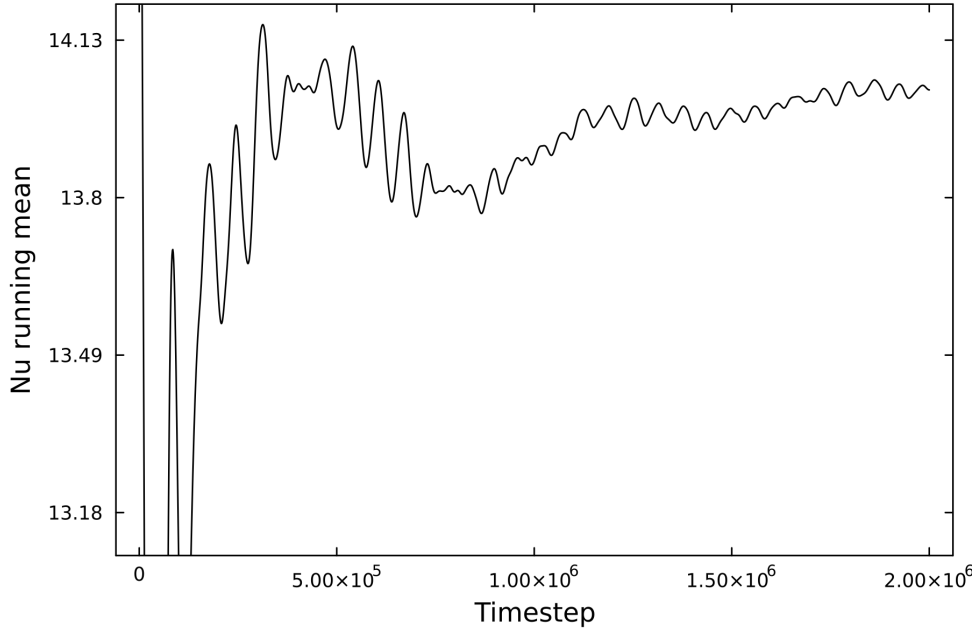


Figure 25: Example of convergence of average Nusselt number during Rayleigh-Bénard simulation at $Ra = 10^8$, $Pr = 1$. The running mean is calculated using $Nu_{mean}(T) = \frac{1}{T} \sum_{t=0}^T Nu(t)$.

which is used to calculate a running average Nusselt number. To evaluate when to stop the simulation, a lookback window of 10^6 timesteps is used. When every Nusselt number in the latest lookback window differs by not more than 0.1% of the window's mean, the simulation is stopped, and the running average Nusselt number is reported as the final Nusselt number. The size of the lookback window is chosen so that at least 5 oscillations are captured for every simulation. An example of the convergence of the average Nusselt number is seen in Fig. 25. For comparison to literature, the Rayleigh-Nusselt relation found by Johnston and Doering [26] given by:

$$Nu = 0.138Ra^{0.285} \quad \text{for } Pr = 1, \quad (118)$$

using the same setup is used. From their paper, the Johnston and Doering relation is seen to be valid from $10^6 < Ra < 10^{10}$, and underpredicts the Nusselt numbers at lower Rayleigh.

7.3.1 Nusselt number results

The results of simulations for the different models are shown in Fig. 26. At $Ra = 10^3$, all of the models correctly predict a pure diffusive state with $Nu = 1.0$. With $Ra \in \{10^4, 10^5, 10^6\}$, the models all predict a Nusselt number slightly bigger than the empirical relation, similar to the findings by Johnston and Doering [26]. With $Ra > 10^6$, the Entropic method did not converge within a reasonable amount of time and therefore no average Nusselt number is found. The reason for the failure of convergence with the Entropic method is seen by looking at the resulting temperature field over time, which shows the entire temperature field moving horizontally. An example is seen in Fig. 27. This is an unphysical result since the only external forces in the system are vertical, coming from the buoyancy force. With $Ra = 10^8$ and $Ra = 10^9$, SRT, MRT, and the Cumulant method all show excellent agreement with the reference. At $Ra = 10^{10}$, the temperatures in the SRT simulation diverge. The MRT and Cumulant method give a Nusselt number lower than the reference relation. An increase in Rayleigh number is accompanied by progressively finer relevant length scales. The size of the smallest relevant velocity scales, called

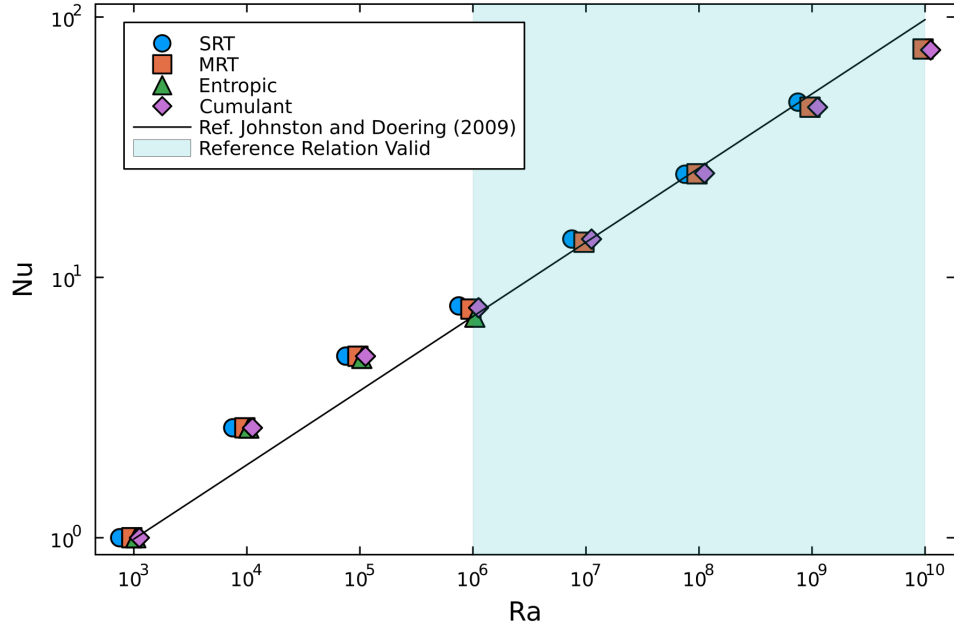


Figure 26: Nusselt number as a function of Rayleigh number at a Prandtl number of 1 for a Rayleigh-Bénard simulation on a 600x300 grid with viscosities and thermal diffusivities from $\nu = \kappa = 2.7 \times 10^{-1} \frac{\Delta x^2}{\Delta t}$ to $\nu = \kappa = 8.7 \times 10^{-5} \frac{\Delta x^2}{\Delta t}$ using different lattice Boltzmann method. A small horizontal offset is used for each point for visibility. The Entropic method does not converge for $Ra \geq 10^7$, and the SRT method diverges in its temperature field for $Ra = 10^{10}$. These data points were therefore excluded.

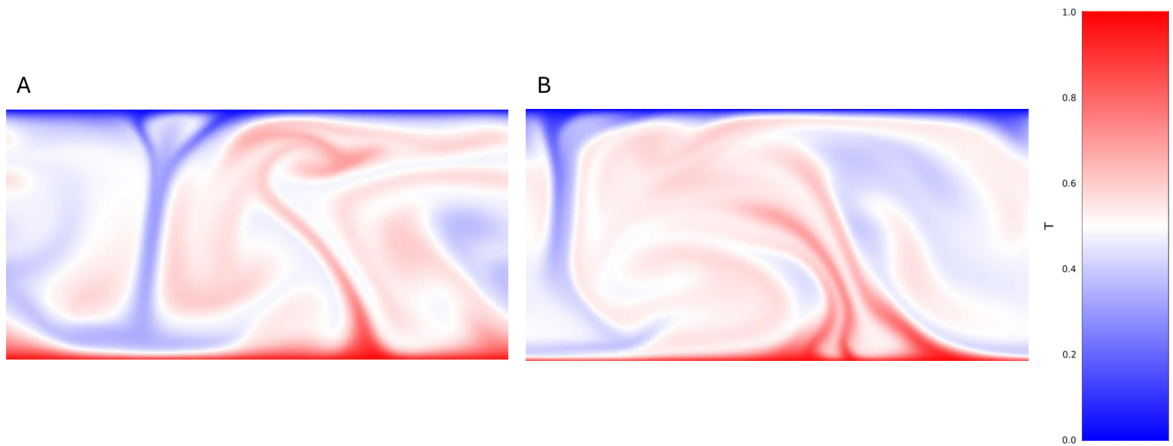


Figure 27: Temperature field of a Rayleigh-Bénard convection simulation using the Entropic lattice Boltzmann method on a 600x300 grid at $Ra = 10^7$, $Pr = 1$ with $\nu = \kappa = 2.7 \times 10^{-3}$. The entire temperature field is unphysically moving to the left as seen by comparing (A) at a prior time step with (B) a couple of time steps later.

the Kolmogorov lengthscale η can be estimated using dimensional analysis giving $\eta \sim (\nu^3/\epsilon)^{1/4}$ with ϵ the rate of dissipation of turbulent kinetic energy per unit mass using the scaling $\epsilon \sim U^3/H$ with U a characteristic velocity scale, and H a characteristic length scale. The length scale is given here by the distance between the hot and cold wall: 300 lattice nodes. A larger velocity will give a smaller Kolmogorov length scale. A conservative estimate of η uses the free fall velocity as the characteristic velocity. Using the acceleration from Boussinesq approximation. Substituting these relations gives:

$$\begin{aligned}\eta &\sim \nu^{3/4} H^{1/4} (\beta g \Delta T H)^{-3/8} \\ &= \nu^{3/4} (\beta g \Delta T H^3)^{-3/8} H \\ &= Ra^{-3/8} \nu^{3/8} \kappa^{-3/8} H \\ \eta/H &\sim \left(\frac{Ra}{Pr} \right)^{-3/8}.\end{aligned}\tag{119}$$

Similarly the smallest temperature length scale is given by the Batchelor scale [54]:

$$\begin{aligned}\eta_B &\sim \eta / \sqrt{Pr} \\ \eta_B/H &\sim Ra^{-3/8} Pr^{-1/8}.\end{aligned}\tag{120}$$

Eqs. (119, 120) can now be used to estimate for which Ra the simulation is fully-resolved by requiring $\eta > \Delta x/\pi$ and $\eta_B > \Delta x/\pi$ following the Grötzsch criterion [55]. Using $Pr = 1$ and $H = 300$, this gives a maximum resolved Rayleigh number of $Ra = 8.5 \cdot 10^7$. Since $Pr = 1$, the temperature based estimation gives the same Rayleigh limit. Notably, the LBM methods are able to recover Nusselt numbers close to the reference past 10^7 to approximately $Ra = 10^9$. This suggests that the free fall velocity only gives a loose upper bound on the required resolution.

7.3.2 Simulation time

The methods can furthermore be compared when looking at the number of time steps required for convergence. Fig. 28 shows the total time steps until convergence for the same simulations as in Fig. 26. In general, there is an upwards trend. With $Ra \leq 10^5$, the velocity and temperature field are almost completely static such that the Nusselt number does not significantly change over time. At $Ra = 10^6$ the required steps of the Entropic method increases sharply with the flow showing very slow convergence together with unphysical horizontal movement in the temperature and velocity field. MRT at $Ra = 10^7$ shows very slow convergence with long time scale behavior. To check for mistakes, the simulation is repeated which again results in a similar convergence at $\approx 2 \cdot 10^7$ steps. The outlier is not attributable to a physical transition since the other methods do not show such convergence at this Ra , and the repeat simulation rules out a one-off numerical accident. The most likely explanation is a numerical artefact specific to the MRT method at this parameter combination, but its origin remains unclear. Two methods requiring the same number of time steps does not mean that the simulations took the same amount of time. The wall-clock times for the different methods are shown in Fig. 29 which is consistent with Fig. 28. With SRT, MRT, and the Cumulant method all having similar performance, the differences in their simulation times are small other than from having different required convergence steps. The differences that are visible should also be regarded in the context that these simulations are performed on a consumer computer, often while other processes are running simultaneously potentially slowing down some simulations more than others. To conclude, a positive correlation is observed between Rayleigh number, and convergence time. The Entropic method fails to

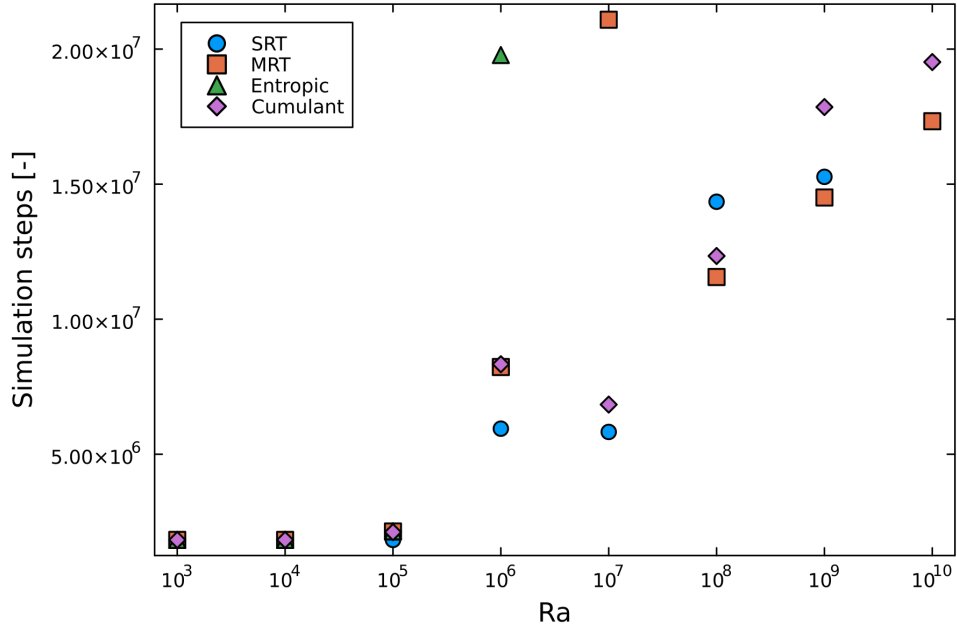


Figure 28: Number of time steps required for the average Nusselt number to converge in a Rayleigh-Bénard simulation using different LBM methods over a range of Rayleigh numbers.

converge at high Ra , and the SRT method diverges at $Ra = 10^{10}$. Comparing the methods with each other in terms of physical simulation time would require more testing, as background processes could potentially slow down some simulations more than others.

7.4 High Prandtl simulations

To simulate flows at Prandtl numbers larger than 1, the most suitable simulation method is selected based on the preceding results. In every test done in this thesis, the Cumulant method resulted in the most accurate and most stable simulations. The only exception concerns stability in the lid-driven cavity, where the Entropic method achieves superior stability at small viscosities. However because the Cumulant method shows substantially higher accuracy, it is selected for the high Prandtl number simulations. Following the Prandtl numbers usually present in molten salt reactors [24], the Prandtl number is varied from $Pr = 1$ to $Pr = 35$. For the Rayleigh number, a high Ra would mean larger grids for resolving all the scales, but a small Ra would mean the flow remains in the weakly convective or laminar regime, which are less physically interesting. For the simulations here, a Rayleigh number of $Ra = 10^7$ is chosen. Following the scaling analysis of the previous chapter, at $Pr = 1$, this would give a minimum resolution of $H \approx 400$, and at $Pr = 35$: $H \approx 650$ with the limiting scale now being the Batchelor scale, however, as seen in the previous chapter, results might still be accurate for coarser grids than what would be predicted based on the free-fall velocity. Here the simulations are performed at a 800×400 grid. To verify grid convergence, the $Pr = 35$ simulation, which would require the finest grid, is repeated using a grid of 1600×800 . Comparing the resulting Nusselt numbers, a difference of 2% is found which, for this thesis, is deemed sufficient. The free input parameters of the performed simulations is seen in Table. 5. The results for the Prandtl sweep can be seen in Fig. 30. Unlike for the Rayleigh sweep, there is no clear relation between Nu and Pr at $Ra = 10^7$. All of the Nusselt numbers are within 14.2 ± 0.4 of each other which is comparable to the 2% difference found in the grid refinement test. Nusselt numbers not strongly depending on Prandtl numbers is also found in the literature for example, Poel et al. [27] found similar results with free-slip boundary

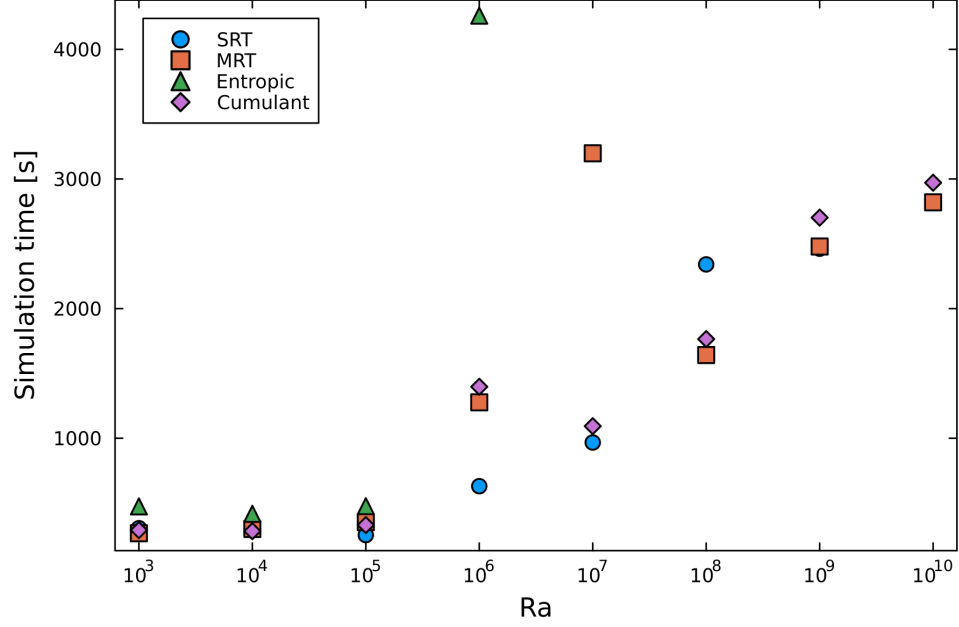


Figure 29: Wall-clock simulation time required for the average Nusselt number to converge in a Rayleigh-Bénard simulation using different LBM methods over a range of Rayleigh numbers.

Table 5: Free input parameters of Rayleigh-Bénard convection simulations with varying Prandtl number.

Prandtl number	Viscosity [$\frac{\Delta x^2}{\Delta t}$]	Viscosity [$\frac{m^2}{s}$]	Thermal diffusivity [$\frac{\Delta x^2}{\Delta t}$]	Thermal diffusivity [$\frac{m^2}{s}$]
1	3.7×10^{-1}	1.6×10^{-4}	3.7×10^{-1}	1.6×10^{-4}
2	5.2×10^{-1}	2.2×10^{-4}	2.6×10^{-1}	1.1×10^{-4}
3	6.3×10^{-1}	2.7×10^{-4}	2.1×10^{-1}	9.1×10^{-5}
4	7.3×10^{-1}	3.2×10^{-4}	1.8×10^{-1}	7.9×10^{-5}
5	8.2×10^{-1}	3.5×10^{-4}	1.6×10^{-1}	7.1×10^{-5}
10	1.2×10^0	5.0×10^{-4}	1.2×10^{-1}	5.0×10^{-5}
20	1.6×10^0	7.1×10^{-4}	8.2×10^{-2}	3.5×10^{-5}
35	2.2×10^0	9.4×10^{-4}	6.2×10^{-2}	2.7×10^{-5}

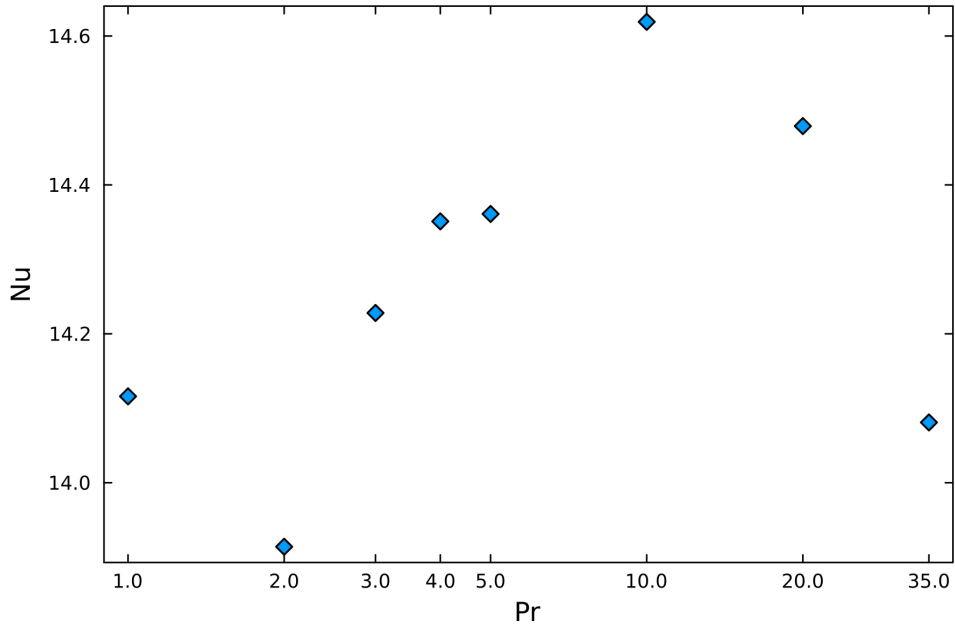


Figure 30: Nusselt numbers as a function of the Prandtl number at a Rayleigh number of 10^7 with viscosities from $\nu = 3.7 \times 10^{-1} \frac{\Delta x^2}{\Delta t}$ to $\nu = 2.2 \frac{\Delta x^2}{\Delta t}$, and thermal diffusivities from $\kappa = 3.7 \times 10^{-1} \frac{\Delta x^2}{\Delta t}$ to $\kappa = 6.2 \times 10^{-2}$ in Rayleigh-Bénard simulations using the Cumulant lattice Boltzmann method.

conditions instead of no-slip boundary conditions in the $Pr = 1$ to $Pr = 100$ range for $Ra = 10^8$. Qualitatively, the difference in temperature fields with different Prandtl numbers can be seen in Fig. 31 illustrating the finer thermal structures at higher Prandtl number despite comparable Nusselt numbers.

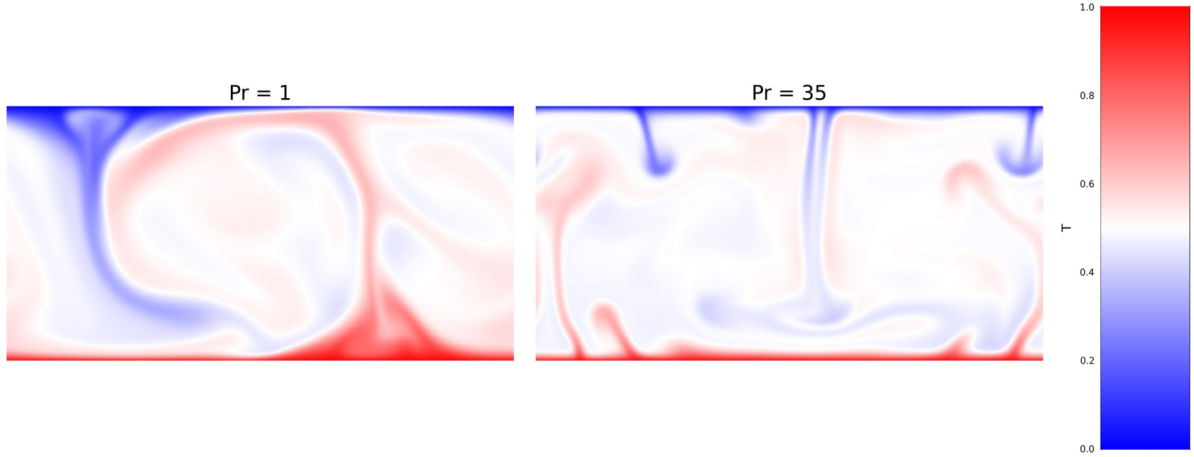


Figure 31: Temperature fields of Rayleigh-Bénard simulations on an 800×400 grid at $Ra = 10^7$ for $Pr = 1$ and $Pr = 35$ with viscosity and thermal diffusivity $\nu = \kappa = 3.7 \times 10^{-1} \frac{\Delta x^2}{\Delta t}$. The red color at the bottom is the hot wall temperature, the blue color at the top is the cold wall temperature.

8 Conclusion

8.1 Findings

This thesis investigated the performance and equivalence of lattice Boltzmann methods in GPU-accelerated simulations, with particular focus on their suitability for the high Prandtl number regime relevant to molten salt reactors. The methods investigated in this thesis include the single relaxation time (SRT), multiple relaxation time (MRT), filter matrix lattice Boltzmann (FMLB), matrix projected Boltzmann (MPB), the Entropic, and the Cumulant method.

8.1.1 Qualitative Comparison

First, the methods are tested on the isothermal benchmark of lid-driven cavity flow, and doubly periodic shear layers, where stability and accuracy are qualitatively compared. In the pre-study, the stability results show that SRT is the least stable method, followed by MPB. The MRT and FMLB method are more stable, and are comparable to each other. The Cumulant method demonstrates a further improvement in stability for low viscosity simulations. Finally, the Entropic method is stable for all the low viscosity tests done, and only shows instability at the very high viscosity limit due to catastrophic cancellation near equilibrium. In terms of accuracy the MRT, FMLB, and Cumulant method all show qualitatively very similar flow fields for the lid-driven cavity case, while MPB gives unphysical results for small viscosities with the difference between MPB and the other moment space methods being attributable to a lack of independent relaxation of bulk viscosity. Both SRT and the Entropic method demonstrate checkerboard-like numerical artefacts with increased intensity closer to the lid. For the doubly periodic shear layers, SRT produces unphysical additional vortices, and the Entropic method shows extra diffusion with both of these effects lessening with finer grids. The MRT, FMLB, MPB, and Cumulant method all give qualitatively similar results for the doubly periodic shear layers.

8.1.2 MRT and FMLB equivalence

Motivated by the similar results for the MRT and FMLB methods, a mathematical equivalence proof is presented by looking at the actual computational algorithm used for both methods. Even though MRT and FMLB are derived in different ways, it can be seen that MRT and FMLB are almost exactly mathematically equivalent. The only differences are a redefinition of the higher order relaxation rates, and a different treatment of body forces, however both of these could easily be adapted to both methods making them exactly mathematically equivalent. In addition, it is found that the MPB method can be fit into the same framework by setting higher order relaxation parameters to 1, and setting the bulk viscosity relaxation rate equal to that of the shear viscosity. It should be noted that the MPB framework is more general than the specific formulation analyzed here, and different choices within the framework could yield different behavior.

8.1.3 Rayleigh-Bénard convection varying Rayleigh

The SRT, MRT, Entropic, and Cumulant method were then used to simulate Rayleigh-Bénard convection across Rayleigh numbers ranging from 10^3 to 10^{10} at $Pr = 1$ measuring Nusselt numbers. The results were then compared to a $Nu(Ra, Pr)$ relation found by Johnston and Doering [26]. The MRT and Cumulant method show excellent agreement with the reference relation in the relevant Rayleigh range. SRT demonstrates similar accuracy however the simulation crashed at $Ra = 10^{10}$. The Entropic method achieved similar Nusselt numbers to the rest of the methods at low Ra , but, at higher Ra , resulted in unphysical bulk horizontal movement with very slow convergence.

8.1.4 Rayleigh-Bénard convection varying Prandtl

Next, the effect of higher Prandtl numbers on the heat transfer was tested by simulating Rayleigh-Bénard convection at $Ra = 10^7$ using $Pr = 1$ through $Pr = 35$. For these simulations, the Cumulant method was used for its demonstrated accuracy and stability compared to the other methods on the rest of the tests in this thesis. The resulting Nusselt numbers showed no clear dependency on Pr with Nusselt number staying within $Nu = 14.2 \pm 0.4$ over the entire Pr range despite clear qualitative differences in the temperature field at different Prandtl numbers. This suggests that heat transfer in the relevant MSR Prandtl number range is robust and not strongly sensitive to the exact value of Pr .

8.2 GPU optimization

In addition to mathematical and simulation results, the GPU implementation used for the different methods in this thesis is explained in great detail with techniques to improve performance. The simulations were done on a consumer RTX 3060 Ti GPU achieving $\approx 90\%$ of theoretical performance based on a memory bandwidth based maximum similar to the fastest LBM implementations publically available [3].

8.3 Research questions

Based on the findings, the research questions can now be answered:

- How do SRT, MRT, FMLB, and MPB compare in terms of stability on isothermal simulations?

In all tests done the SRT method was the most unstable, crashing at low viscosities. While

the MPB method does crash slightly less often than SRT, the resulting velocity field can be dominated by numerical artefacts as seen in the lid-driven cavity simulations. The MRT and FMLB method show very similar stability. Slight seemingly sporadic differences were only found when comparing lid-driven cavity flow at the edges of the border between stable and unstable simulations.

- How do the FMLB and MPB method compare mathematically to the MRT method?
Rewriting the FMLB and MPB method in the same form as MRT it is seen that the only differences are a definition of higher order relaxation parameters, and the specific forcing method. Both of these can also fit into the MRT method, after which the methods are equivalent.
- How do the SRT, MRT, Entropic, and Cumulant LBM compare when simulating thermal Rayleigh-Bénard flow in terms of stability and accuracy?
In the relevant Ra range, the SRT, MRT, and Cumulant method all show excellent agreement with reference data. With the simulations done for this thesis, instability was only encountered using SRT on very high Ra . While the Entropic method does not become unstable, unphysical behavior is demonstrated in the higher Ra simulations, with bulk horizontal movement of the temperature field. The MRT and Cumulant method perform very similarly being stable throughout the entire Ra range, and showing agreement with the reference data until $Ra = 10^9$ after which the resulting Nusselt number decreases compared to literature due to the smallest scales being unresolved.
- What is the effect of variations in Prandtl number from 1 to 35 in 2D Rayleigh-Bénard flow with no-slip bottom and top walls and periodic side-walls?
Using the Cumulant method for the Prandtl variation results, there is seemingly no significant effect of changing Pr from 1 through 35 on heat transfer at $Ra = 10^7$ with the resulting Nusselt number staying within $Nu = 14.2 \pm 0.4$

8.4 Limitations

The simulations from this thesis were done in 2D only. 2D flow has a fundamentally different turbulence structure compared to 3D meaning that generalization to 3D are limited especially at high Reynolds or high Rayleigh numbers. For the Rayleigh-Bénard convection simulations, periodic side-walls were used with a fixed aspect ratio of 2 even though Wang et al. [25] found aspect ratio to affect Nusselt numbers. In terms of parameter space, Pr is only varied from 1 to 35 where Nu did not change significantly. Results for $Pr < 1$ might be very different as might results for Pr larger than 35. The Pr variations were also only done for $Ra = 10^7$. It is unclear how heat transfer would be affected by Pr at different Rayleigh numbers. In terms of methods, the specific implementation of the Entropic method used in this thesis uses a lower order approximation for the collision step size. Higher order methods exist which could yield better accuracy.

8.5 Recommendations

Future work should perform three-dimensional simulations to obtain results applicable to real molten salt reactors. In terms of simulations, a larger range of Prandtl numbers could be simulated. For this thesis, these simulations took ~ 1 hour. On a more powerful GPU like the NVIDIA A100 which is used often in supercomputers, a speedup of $\sim 4x$ based on bandwidth would reduce this to only 15 minutes. With this increased performance it would be possible to simulate at finer grids, or in a larger parameter space. With the moment based methods,

the higher order relaxation rates could be tuned to see if accuracy is improved which was not done in this thesis. From a theoretical perspective, the analysis by Geier et al. [8] could provide a framework for analyzing thermal flows. With this method, it might be able to identify leading order errors just like for isothermal flow, or find LBM specific approximations for thermal coupling instead of using the Boussinesq approximation approach. A further avenue for future work is to compare the mathematical framework of the MPB method to the Cumulant method. Both methods avoid a Chapman-Enskog analysis by trying to fit moments (or cumulants) more directly to the desired equations. An overlap between these approaches may exist. Additionally, framework of the MPB method could be used to find LBM methods which are not equivalent to the MRT method. These could yield different results. Finally, the concepts underlying the Entropic and Cumulant methods could be combined, where relaxation rates are adaptively set to avoid numerical instability. This would require finding the space of relaxation rates for which the Lyapunov functional is constant, which may prove analytically nontrivial.

A Appendix

A.1 MRT and FMLB equivalence for general bulk viscosity

The equivalence between FMLB and MRT for general bulk viscosity follows from comparing Eq. (86) with the 4th to 6th rows of Eq. (79). First the relevant rows of Eq. (79) are written in the same form as Eq. (86) giving:

$$\begin{aligned} m_4^+ &= m_4^- + (1 - G_1)(\rho u_x u_x - m_4^-) + (-G_2)(\rho u_y u_y - m_6^-) \\ m_6^+ &= m_6^- + (1 - G_1)(\rho u_y u_y - m_6^-) + (-G_2)(\rho u_x u_x - m_4^-). \end{aligned} \quad (121)$$

Comparing to Eq. (86) it is seen that equivalence requires:

$$\begin{aligned} 1 - G_1 &= \Delta t \frac{\omega_b + \omega}{2} \\ G_2 &= \Delta t \frac{\omega - \omega_b}{2}. \end{aligned} \quad (122)$$

Using $\omega = \frac{1}{\frac{\nu}{c_s^2} + \frac{\Delta t}{2}}$ and $\omega_b = \frac{1}{\frac{\nu_B + \nu/3}{c_s^2} + \frac{\Delta t}{2}}$ this results in:

$$\begin{aligned} \Delta t \frac{\omega + \omega_b}{2} &= \frac{\Delta t}{2} \frac{\frac{\nu_B + \nu/3}{c_s^2} + \frac{\nu}{c_s^2} + \frac{\Delta t}{2}}{\frac{\nu \nu_B}{c_s^4} + \frac{\nu^2}{3c_s^4} + \frac{2\Delta t \nu}{3c_s^2} + \frac{\Delta t \nu_B}{2c_s^2} + \frac{\Delta t^2}{4}} \\ &= \frac{2c_s^2 \Delta t (\nu_B + \frac{4}{3}\nu + c_s^2 \Delta t)}{4\nu \nu_B + \frac{4}{3}\nu^2 + \frac{8}{3}c_s^2 \Delta t \nu + 2c_s^2 \Delta t \nu_B + c_s^2 \Delta t^2} \\ &= \frac{2c_s^2 \Delta t (\nu_B + \frac{4}{3}\nu + c_s^2 \Delta t)}{(2\nu + c_s^2 \Delta t)(2\nu_B + \frac{2}{3}\nu + c_s^2 \Delta t)}, \end{aligned} \quad (123)$$

and for the second coefficient:

$$\Delta t \frac{\omega - \omega_b}{2} = \frac{2c_s^2 \Delta t (\nu_B - \frac{2}{3}\nu)}{(2\nu + c_s^2 \Delta t)(2\nu_B + \frac{2}{3}\nu + c_s^2 \Delta t)}. \quad (124)$$

The same expressions now have to be found for the left hand side of Eq. (122) using:

$$\begin{aligned}
1 - G_1 &= \frac{B_1(1 - G)}{B_1 + B_2} \\
1 - G &= \frac{G^- - G^+}{G^-} \\
G^- - G^+ &= -2c_s^2 \Delta t \rho \\
A &\equiv 2\nu + c_s^2 \Delta t \\
G^- &= -\rho A \\
1 - G &= \frac{2c_s^2 \Delta t}{A} \\
B_1 &= -\rho(A + (B - 1)\nu) \\
B_1 + B_2 &= -\rho(A + 2(B - 1)\nu) \\
\frac{B_1}{B_1 + B_2} &= \frac{A + (B - 1)\nu}{A + 2(B - 1)\nu} \\
1 - G_1 &= \frac{2c_s^2 \Delta t}{A} \frac{A + (B - 1)\nu}{A + 2(B - 1)\nu} \\
(B - 1)\nu &= \nu_B - \frac{2}{3}\nu \\
1 - G_1 &= \frac{2c_s^2 \Delta t(\nu_B + \frac{4}{3}\nu + c_s^2 \Delta t)}{(2\nu + c_s^2 \Delta t)(2\nu_B + \frac{2}{3}\nu + c_s^2 \Delta t)}.
\end{aligned} \tag{125}$$

The same can be done for G_2 providing:

$$\begin{aligned}
G_2 &= \frac{(1 - G)B_2}{B_1 + B_2} \\
G_2 &= \frac{2c_s^2 \Delta t}{A} \frac{(B - 1)\nu}{A + 2(B - 1)\nu} \\
G_2 &= \frac{2c_s^2 \Delta t(\nu_B - \frac{2}{3}\nu)}{(2\nu + c_s^2 \Delta t)(2\nu_B + \frac{2}{3}\nu + c_s^2 \Delta t)}.
\end{aligned} \tag{126}$$

Q.E.D.

A.2 MRT and FMLB comparison for non-zero body forces

In FMLB, forcing is done in two places [9]. The velocities used in calculating equilibria moments is shifted by $u_{\alpha, new} = u_{\alpha} + \frac{\Delta t}{2\rho} F_{\alpha}$, and the outgoing 2nd and 3rd moments are shifted by $\Delta t F_{\alpha}$. This is similar to the moment transformed Guo forcing [34], except for the missing source term contribution to moments of higher order than the velocity moments as seen in Eq. (87). For the MRT method, multiple forcing treatments exist [1]. The FMLB forcing scheme could also be used in the MRT method after which, both methods are fully equivalent.

References

- [1] Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. *The lattice Boltzmann method*, volume 10. Springer, 2017.
- [2] Roberto Benzi, Sauro Succi, and Massimo Vergassola. The lattice boltzmann equation: theory and applications. *Physics Reports*, 222(3):145–197, 1992.
- [3] Moritz Lehmann. FluidX3D, August 2022.
- [4] Dominique d’Humières. Multiple-relaxation-time lattice boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.
- [5] Pierre Lallemand and Li-Shi Luo. Theory of the lattice boltzmann method: Dispersion, dissipation, isotropy, galilean invariance, and stability. Technical report, 2000.
- [6] JA Somers. Direct simulation of fluid flow with cellular automata and the lattice-boltzmann equation. *Applied Scientific Research*, 51(1):127–133, 1993.
- [7] Ilya V Karlin, Antonio Ferrante, and Hans Christian Öttinger. Perfect entropy functions of the lattice boltzmann method. *EPL (Europhysics Letters)*, 47(2):182–188, 1999.
- [8] Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. The cumulant lattice boltzmann equation in three dimensions: Theory and validation. *Computers & Mathematics with Applications*, 70(4):507–547, 2015.
- [9] Congshan Zhuo, Chengwen Zhong, and Jun Cao. Filter-matrix lattice boltzmann model for incompressible thermal flows. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 85(4):046703, 2012.
- [10] Joran Winden. Developing a liquid fraction incorporated wet-node boundary condition scheme for conjugate heat transfer in the filter-matrix lattice boltzmann method. 2026.
- [11] Twan Terpstra. Advancing towards lattice boltzmann simulations of molten salt fast reactor cores. 2025.
- [12] Jordi Reitsma. Simulating ice layer development in turbulent channel flow using a gpu-accelerated filter-matrix lattice boltzmann method. 2025.
- [13] Robin de Waal. Modelling a freeze plug with the filter matrix lattice boltzmann method to investigate effects of freeze plug design parameters on melting times. 2025.
- [14] Thomas Pijls. Developing a gpu-accelerated filter-matrix lattice boltzmann multiphysics tool for the transient thermal-hydraulics and neutronics of a molten salt fast reactor core. 2025.
- [15] Tom Entes. Development and integration of a gpu-accelerated lattice boltzmann simulation tool for thermal hydraulics with neutronics for multiphysics simulations in molten salt fast reactor cores. 2025.
- [16] Lotte Borstlap. Developing a filter-matrix lattice-boltzmann boundary method for conjugate heat transfer and phase change. 2024.
- [17] Pieter Spek. Gpu-accelerated largeeddy simulation of non-eutectic msfr salt freezing in turbulent channel flow. 2024.

- [18] Mees Wortelboek. Investigating gpu-accelerated double distribution function lattice boltzmann schemes for heat transfer and phase change in turbulent flows. 2023.
- [19] Thorben Besseling. Investigating adaptive mesh refinement criteria for a double-distribution fmlb scheme in melting and solidification processes. 2022.
- [20] Celeke Bus. Simulating the transient freezing in cooled non-eutectic molten salt channel flow. 2022.
- [21] Viktor Sundström. Deriving filter matrix methods for lattice boltzmann simulations. 2025.
- [22] Christophe Coreixas, Bastien Chopard, and Jonas Latt. Comprehensive comparison of collision models in the lattice boltzmann framework: Theoretical investigations. *Physical Review E*, 100(3):033305, 2019.
- [23] Jonas Latt and Bastien Chopard. Lattice boltzmann method with regularized pre-collision distribution functions. *Mathematics and Computers in Simulation*, 72(2-6):165–168, 2006.
- [24] Sheng Zhang. Mixed convective heat transfer of medium-prandtl-number fluids in horizontal circular tubes. *International Journal of Heat and Mass Transfer*, 190:122740, 2022.
- [25] Qi Wang, Kai Leong Chong, Richard JAM Stevens, Roberto Verzicco, and Detlef Lohse. From zonal flow to convection rolls in rayleigh–bénard convection with free-slip plates. *Journal of fluid mechanics*, 905:A21, 2020.
- [26] Hans Johnston and Charles R Doering. Comparison of turbulent thermal convection between conditions of constant temperature;? format?; and constant flux. *Physical review letters*, 102(6):064501, 2009.
- [27] Erwin P Van Der Poel, Richard JAM Stevens, and Detlef Lohse. Comparison between two- and three-dimensional rayleigh–bénard convection. *Journal of fluid mechanics*, 736:177–194, 2013.
- [28] 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. *Phys. Rev.*, 94:511–525, May 1954.
- [29] Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. Kinetic theory representation of hydrodynamics: a way beyond the navier–stokes equation. *Journal of Fluid Mechanics*, 550:413–441, 2006.
- [30] Xiaoyi He and Li-Shi Luo. Theory of the lattice boltzmann method: From the boltzmann equation to the lattice boltzmann equation. *Physical review E*, 56(6):6811, 1997.
- [31] Sauro Succi. *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford university press, 2001.
- [32] JGM Eggels and JA Somers. Numerical simulation of free convective flow using the lattice-boltzmann scheme. *International Journal of Heat and Fluid Flow*, 16(5):357–364, 1995.
- [33] Martin Rohde. The filter-matrix lattice-boltzmann method. 2025.
- [34] Zhaoli Guo and TS Zhao. Lattice boltzmann model for incompressible flows through porous media. *Physical review E*, 66(3):036304, 2002.

- [35] Congshan Zhuo and Chengwen Zhong. Les-based filter-matrix lattice boltzmann model for simulating fully developed turbulent channel flow. *International Journal of Computational Fluid Dynamics*, 30(7-10):543–553, 2016.
- [36] Congshan Zhuo, Chengwen Zhong, and Jun Cao. Filter-matrix lattice boltzmann simulation of lid-driven deep-cavity flows, part i—steady flows. *Computers & Mathematics with Applications*, 65(12):1863–1882, 2013.
- [37] Congshan Zhuo, Chengwen Zhong, and Jun Cao. Filter-matrix lattice boltzmann simulation of lid-driven deep-cavity flows, part ii—flow bifurcation. *Computers & Mathematics with Applications*, 65(12):1883–1893, 2013.
- [38] Congshan Zhuo and Chengwen Zhong. Filter-matrix lattice boltzmann model for microchannel gas flows. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 88(5):053311, 2013.
- [39] Martin Rohde, Drona Kandhai, Jos J Derksen, and Harry EA Van den Akker. A generic, mass conservative local grid refinement technique for lattice-boltzmann schemes. *International journal for numerical methods in fluids*, 51(4):439–468, 2006.
- [40] Dominique d’Humières. Generalized lattice boltzmann equations, rarefied gas dynamics: theory and simulations. *Progress in Astronautics and Aeronautics*, 159:450–458, 1992.
- [41] Seyed Ali Hosseini, Mohammad Atif, Santosh Ansumali, and Iliya V Karlin. Entropic lattice boltzmann methods: A review. *Computers & Fluids*, 259:105884, 2023.
- [42] Santosh Ansumali, Ilya V Karlin, and Hans Christian Öttinger. Minimal entropic kinetic models for hydrodynamics. *EPL (Europhysics Letters)*, 63(6):798–804, 2003.
- [43] Mohammad Atif, Praveen Kumar Kolluru, Chakradhar Thantapanally, and Santosh Ansumali. Essentially entropic lattice boltzmann model. *Physical review letters*, 119(24):240602, 2017.
- [44] Martin Geier and Andrea Pasquali. Fourth order galilean invariance for the lattice boltzmann method. *Computers & Fluids*, 166:139–151, 2018.
- [45] Jingsen Feng, Jing Leng, Jingchao Jiang, and Xu Chu. Entropic lattice boltzmann method for general anisotropic advection–diffusion. *arXiv preprint arXiv:2605.01774*, 2026.
- [46] UKNG Ghia, Kirti N Ghia, and CT Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of computational physics*, 48(3):387–411, 1982.
- [47] Christophe Coreixas, Gauthier Wissocq, Guillaume Puigt, Jean-François Boussuge, and Pierre Sagaut. Recursive regularization step for high-order lattice boltzmann methods. *arXiv preprint arXiv:1704.04413*, 2017.
- [48] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.
- [49] Tim Besard, Christophe Foket, and Bjorn De Sutter. Effective extensible programming: Unleashing Julia on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 2018.

- [50] Jung-Chang Huang and Tau Leng. Generalized loop-unrolling: a method for program speedup. In *Proceedings 1999 IEEE Symposium on Application-Specific Systems and Software Engineering and Technology. ASSET'99 (Cat. No. PR00122)*, pages 244–248. IEEE, 1999.
- [51] Geoffrey Ingram Taylor and Albert Edward Green. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, 158(895):499–521, 1937.
- [52] H Nemati, M Farhadi, K Sedighi, E Fattahi, and AAR Darzi. Lattice boltzmann simulation of nanofluid in lid-driven cavity. *International Communications in Heat and Mass Transfer*, 37(10):1528–1534, 2010.
- [53] Francesca Chillà and Joerg Schumacher. New perspectives in turbulent rayleigh-bénard convection. *The European Physical Journal E*, 35(7):58, 2012.
- [54] Olga Shishkina, Richard JAM Stevens, Siegfried Grossmann, and Detlef Lohse. Boundary layer structure in turbulent thermal convection and its consequences for the required numerical resolution. *New journal of Physics*, 12(7):075022, 2010.
- [55] Günther Grötzbach. Spatial resolution requirements for direct numerical simulation of the rayleigh-bénard convection. *Journal of computational physics*, 49(2):241–264, 1983.