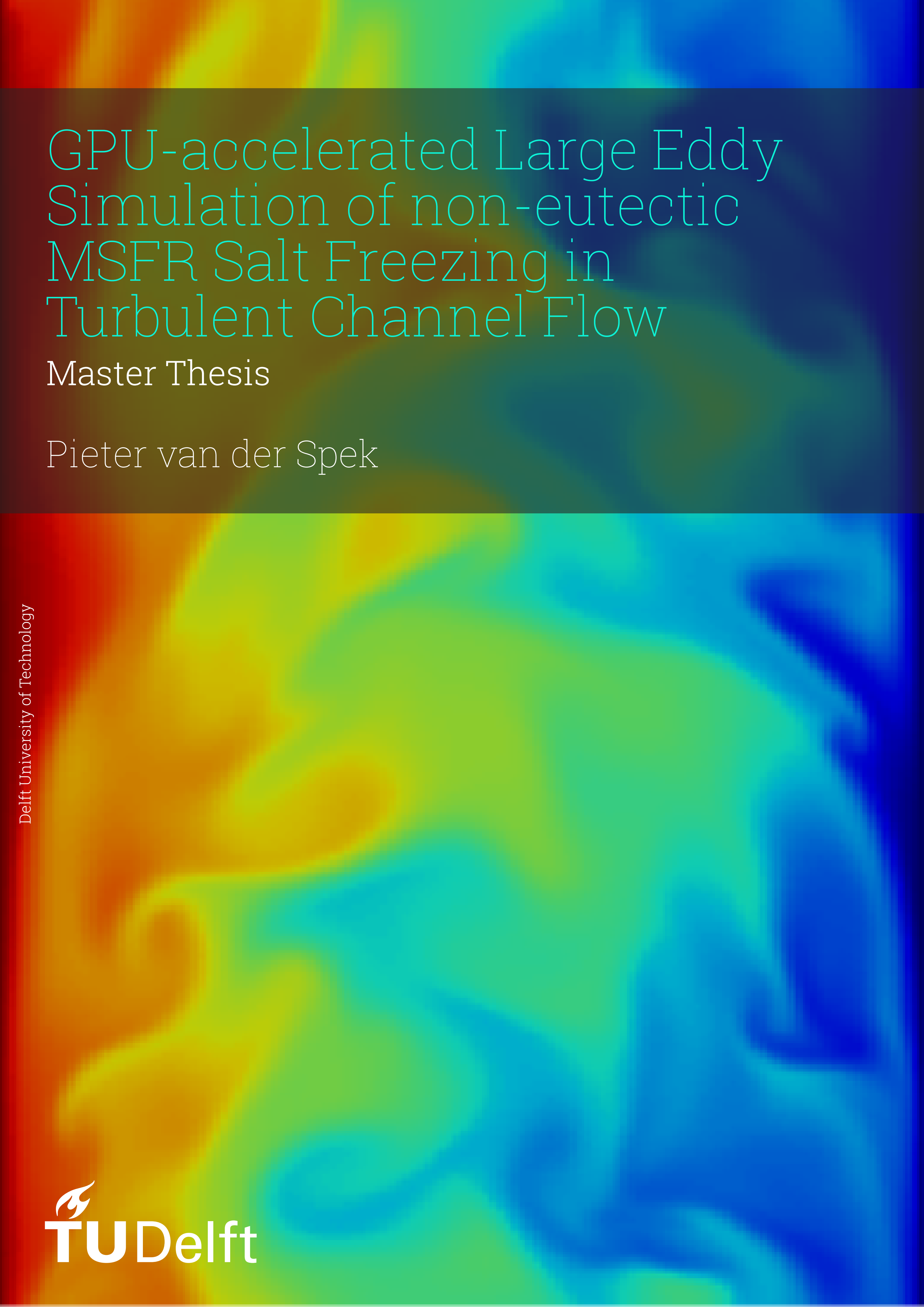


GPU-accelerated Large Eddy Simulation of non-eutectic MSFR Salt Freezing in Turbulent Channel Flow

Master Thesis

Pieter van der Spek

Delft University of Technology

TUDelft

GPU-accelerated Large Eddy Simulation of non-eutectic MSFR Salt Freezing in Turbulent Channel Flow

Master Thesis

by

Pieter van der Spek

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday February 28, 2024 at 10:00 AM

Student number:	4681959	
Project duration:	April 24, 2023 - February 28, 2024	
Thesis committee:	Dr. Ir. M. Rohde,	TU Delft, supervisor
	Dr. Ir. D. Lathouwers,	TU Delft
	Dr. Ir. J. Peeters,	TU Delft

TU Delft

Highlights

The most significant contributions of this thesis are listed below.

1. This work is the first to successfully implement a GPU-accelerated implementation of the filter-matrix lattice Boltzmann method using a large eddy simulation (LES).

2. This work is the first to combine the filter-matrix lattice Boltzmann method with the WALE sub-grid scale model for LES.

3. This work presents a novel hierarchical local grid refinement technique that aims to overcome non-physical operations introduced in existing methods.

4. This work presents an enthalpy transformation procedure in the LBM framework that successfully suppresses thermal fluctuations and increases numerical stability. The procedure resolves challenges encountered in previous research.

5. This work is the first to use an adapted immersed boundary method, derived from Noble and Torczynski's original formulation, to prevent instability during phase change in 3D turbulent channel flows within the LBM framework.

6. This work is the first to successfully simulate phase change in turbulent channel flows using the filter-matrix lattice Boltzmann method, both for eutectic and non-eutectic fluids.

Abstract

The need for continuous carbon-free energy supply makes nuclear energy a valuable addition to the energy mix. The Generation IV International Forum (GIF) made a selection of the most promising next-generation reactor concepts that excel in safety, waste management, resource utilization, and proliferation resistance. One such concept is the Molten Salt Fast Reactor (MSFR) and much research is being conducted to evaluate its safety aspects. One potential risk inherent to MSFRs involves salt freezing inside the heat exchangers, which can lead to degraded heat transfer and structural damage. To further examine this risk, this thesis aims to develop a numerical model that can simulate salt freezing inside cooled 3D turbulent channel flow.

To this end, a GPU-accelerated double distribution filter-matrix lattice Boltzmann (DDF-FMLB) model was implemented. A Large Eddy Simulation (LES) has been applied to model the small unresolved turbulent structures, using a wall-adapting local eddy viscosity (WALE) model. Because the center of the channel requires less resolution than the near-wall region, local grid refinement was implemented to save computational cost. Two techniques were investigated: an existing local grid refinement technique and a novel approach. For laminar channel flow, it was found that the existing technique was both time-convergent and second-order grid convergent towards the analytical Poiseuille solution. The novel technique did not exhibit such convergence, leading to the implementation of the existing technique in further simulations.

Turbulent simulations were performed for $Re_\tau = 180$ and $Re_\tau = 395$ flows, both with a Direct Numerical Simulation (DNS) and a Large Eddy Simulation (LES). Both DNS and LES simulations were in close agreement with benchmark data and showed similar performance. The LES simulations yielded an improved approximation of the mean stream-wise velocity, a slight overshoot of RMS fluctuation, and no change in Reynolds stress. The use of local grid refinement led to ~ 40% more computational efficiency for $Re_\tau = 180$, and ~ 240% for $Re_\tau = 395$. To achieve a more efficient GPU implementation, advanced collision and propagation kernels are recommended.

In addition, the implemented DDF-FMLBM WALE-LES was suitable for simulating turbulent channel flow coupled with heat transfer. A set of transformation rules for temperature and enthalpy were introduced to mitigate fluctuations and instabilities in thermal simulations and proved effective. Both DNS and LES accurately predicted turbulent statistics, with LES outperforming DNS in predicting root-mean-square temperature fluctuations at the channel center, where slight under-predictions were observed compared to reference studies.

To model phase change for both eutectic and non-eutectic fluids, the immersed boundary method was adopted. This method introduced instability in the turbulent velocity field, which was resolved by implementing an adapted immersed boundary method. Simulations of steady-state ice layer formation in a stationary channel yielded accurate results with minor deviations from analytical solutions. Freezing simulations of eutectic turbulent channel flows were also conducted and benchmarked against analytical solutions using Gnielinski's Nusselt correlation. Results indicated thicker ice layers than expected, corresponding to a moderate over-prediction of the Nusselt number by Gnielinski. Similar deviations were observed in reference experiments, suggesting acceptable limits of accuracy. Moreover, simulations involving non-eutectic fluids produced slightly thicker ice layers compared to eutectic simulations, attributed to higher heat conductivity in the mushy region.

In future research on freezing in turbulent channel flows, in- and outflow boundary conditions are recommended to allow for better comparison with reference studies. Furthermore, challenges remain regarding the stability of high-Prandtl number turbulent flows in the lattice Boltzmann framework. Similar to most studies, this thesis restricts itself to low Prandtl numbers ($Pr = 0.71$), while molten salts typically have values $Pr \geq 7.5$. Recently, techniques have been proposed to simulate high-Prandtl turbulent flows, which are recommended for future implementation.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
MSFR	Molten Salt Fast Reactor
LBM	Lattice Boltzmann Method
FMLBM	Filter-matrix lattice Boltzmann method
LES	Large eddy simulation
DNS	Direct Numerical Simulation
DDF	Double-distribution function
LGR	Local grid refinement
WALE	Wall adapting local eddy viscosity
RANS	Reynolds-averaged Navier-Stokes
SGS	Sub-grid scale
GPU	Graphical Processing Unit
LBGK	Lattice Bathnagar-Gross-Krook
MRT	Multi-relaxation time
BC	Boundary condition
bpg	Blocks per grid
tpb	Threads per block

Symbols

Symbol	Definition	Unit
α	Thermal diffusivity	[m^2/s]
$\alpha^{\pm}$	Momentum post-/pre-collision vector	[-]
α_t	Eddy thermal diffusivity	[m^2/s]
$\beta^{\pm}$	Thermal post-/pre-collision vector	[-]
BW	Memory bandwidth	[GB/s]
C_p	Specific heat capacity	[J/kg K]
C_S	Smagorinsky constant	[-]
C_w	WALE model constant	[-]
Δ	Cut-off length	[m]
δ_ν	Viscous length scale	[m]
d	Ice thickness	[-]
E_{ki}	Filter matrix	[-]
$\epsilon_{\text{diff},L2}$	L2 difference	[-]
$\epsilon_{\text{err},L2}$	L2 error norm	[-]
f_i	Discrete distribution function of velocity direction c_i	[-]
f_i^{eq}	Equilibrium distribution function	[-]
f_l	Liquid fraction	[-]
g	Body force	[m/s^2]
g_i	Thermal distribution function of velocity direction c_i	[-]
H	Channel half-height or total enthalpy	[m] or [J/kg]
H_l	Liquidus total enthalpy	[J/kg]
H_s	Solidus total enthalpy	[J/kg]

Symbol	Definition	Unit
h	Sensible enthalpy or heat transfer coefficient	[J/kg] or [W/m^{2}K]
h_i	Interface enthalpy	[J/kg]
h_l	Liquidus sensible enthalpy	[J/kg]
h_L	Lower wall enthalpy	[J/kg]
h_s	Solidus sensible enthalpy	[J/kg]
h_U	Upper wall enthalpy	[J/kg]
L	Latent heat or channel length	[J/kg] or [m]
ℓ	Characteristic eddy scale	[m]
μ	Dynamic viscosity	[Pa s]
$MLUPS$	Million lattice updates per second	[1/s]
N	Number of ... (with subscript)	[-]
Nu	Nusselt number	[-]
ν	Kinematic viscosity	[m^2/s]
ν_t	Eddy-viscosity	[m^2/s]
Ω	Collision operator	[-]
Ω_i^s	Modified collision operator in immersed boundary method	[-]
p	Pressure	[Pa]
Pr	Prandtl number or turbulent Prandtl number	[-]
Pr_t	Turbulent Prandtl number	[-]
Re_m	Bulk Reynolds number	[-]
Re_τ	Shear Reynolds number	[-]
r	Refinement factor	[-]
S_{ij}	Strain-rate tensor	[1/s]
σ_{ij}	Stress tensor	[Pa]
T	Temperature or friction temperature	[K]
T_f	Friction temperature	[K]
T_i	Interface temperature	[K]
T_l	Liquidus temperature	[K]
T_L	Lower wall temperature	[K]
T_s	Solidus temperature	[K]
T_U	Upper wall temperature	[K]
T^+	Non-dimensional temperature	[-]
t	Time	[s]
t^+	Non-dimensional time	[-]
τ	Relaxation time or total shear stress	[s] or [Pa]
τ_{ij}^R	Residual-stress tensor or sub-grid stress tensor	[Pa]
τ_w	Wall shear stress	[Pa]
u^+	Non-dimensional velocity	[-]
u, v, w	x, y, z velocity components	[m/s]
u_τ	Wall shear velocity	[m/s]
u_m	Bulk velocity	[m/s]
w_i	Weight of velocity direction c_i	[-]
y^+	Non-dimensional wall-distance	[-]
ξ	Microscopic particle velocity or friction factor	[m/s] or [-]

1 Introduction

Over the past 70 years, atmospheric CO_2 levels have increased by more than 30% due to rising global energy consumption, and they are expected to increase by another 15% up to 2050 [16, 29]. To reduce climate change, focus has shifted towards carbon-free energy sources, such as wind, hydro, or solar power. Although these are important sources of renewable energy, they are subject to intermittent production, meaning their availability is not constant. Nuclear reactors are a valuable addition as they can provide large supplies of carbon-free energy continuously. Nuclear power is currently responsible for 10% of global electricity production and this share is expected to rise to 14% by 2050, according to IAEA forecasts [2]. Given the increasing electricity demand, this corresponds to an approximate tripling of current installed nuclear capacity [3].

Nuclear reactors generate thermal energy through fission chain reactions, requiring relatively small amounts of fuel compared to traditional fossil-fuel combustion. The majority of the thermal energy produced by these reactors is quickly converted into usable heat, which can be further transformed into electrical energy. The most common method involves using the generated thermal energy to produce high-pressure gas, subsequently driving a turbine to generate electricity. Various reactor designs have been proposed, with initial developments focusing on graphite or heavy-water moderated systems.

The most widely used types of reactors are Pressurized Water Reactors (PWRs) and Boiling Water Reactors (BWRs). PWRs operate with two water loops: the primary loop, which is pressurized to prevent boiling, removes thermal energy from the core, after which hot pressurized water is directed to a heat exchanger, where the secondary-loop water is transformed into high-temperature, high-pressure steam, driving a turbine for electricity generation. In contrast, Boiling Water Reactors (BWRs) allow cooling water to boil while passing through the core. The resulting steam goes directly to the turbine, and the low-pressure steam leaving the turbine is condensed, then pumped back to the reactor. This single-loop system eliminates the need for steam generators and other expensive equipment found in PWRs [95].

While these reactor types are widely used, their designs have shortcomings in terms of safety, waste management, resource utilization, and proliferation resistance [40]. Extensive research is being conducted on so-called Generation IV reactors that aim to overcome those limitations. Six reactor technologies have been selected for further research and development by the Generation IV International Forum[1] (GIF) [54, 36]. One of these is the Molten Salt Reactor (MSR), a family of liquid-fueled fission reactor concepts using a fluid molten salt mixture as fuel [5].

1.1. The Molten Salt Fast Reactor

The MSFR is based on the MSR concept and thus contains a circulating liquid salt that acts both as the coolant and the fuel. The MSR concept is not new and there have already been significant experimental studies with operational MSRs. The Molten Salt Reactor Experiment operated without trouble from 1966 to 1969 using a liquid-fluoride based fuel and a graphite-moderated thermal neutron spectrum [5]. The MSFR is still in the conceptual stage. It adopts a fast neutron spectrum, thereby eliminating the need for a moderator. The reactor has a large negative reactivity feedback coefficient,

[1]The leading organization for multinational collaboration on nuclear system research and development.

and is able to operate in a Thorium fuel cycle. The fast spectrum of the MSFR enables better breeding capabilities and good trans-uranic isotope (TRU) burning, leading to extended resource utilization and waste minimization. The negative feedback coefficient arises from a combination of the Doppler effect and density variations of the molten salt. The result is a passive reactivity control mechanism that is balanced between power generation in the reactor salt and heat removal at the heat exchangers [32]. The MSFR fuel circuit also incorporates a passive salt draining system for planned shutdowns or in case of incidents, preventing excessive temperature increases in the core.

1.1.1. Design

The reference MSFR is a 3000 MWth reactor that uses a molten binary fluoride salt, primarily composed of 77.5% lithium fluoride [51]. The total fuel salt volume amounts to 18 m^3 and is operated at a maximum fuel salt temperature of 750°C. A schematic view of the fuel circuit is given in Fig. 1.1 [101, 75]. Note that there is a continuous flow of salt from the reactor core (green) through the heat exchangers which are located circumferentially around the vessel. Although the flow is primarily driven by a set of pumps, the MSFR could be designed in a way that natural circulation is enabled in the case of a power outage [32]. The annular breeding blanket, shown in red, is filled with fertile salt and has the purpose of capturing neutrons leaking from the core to produce extra fuel [95]. The circuit also involves a unit that injects gas bubbles near the pumps for the removal of non-soluble fission products like noble metals. At the same time, the circulating liquid salt makes it possible to continuously reprocess a small portion of it without the need to shut down the reactor. Lastly, at the bottom of the reactor, the fuel salt can be drained to dedicated reservoirs using a freeze plug if excessive temperatures are reached.

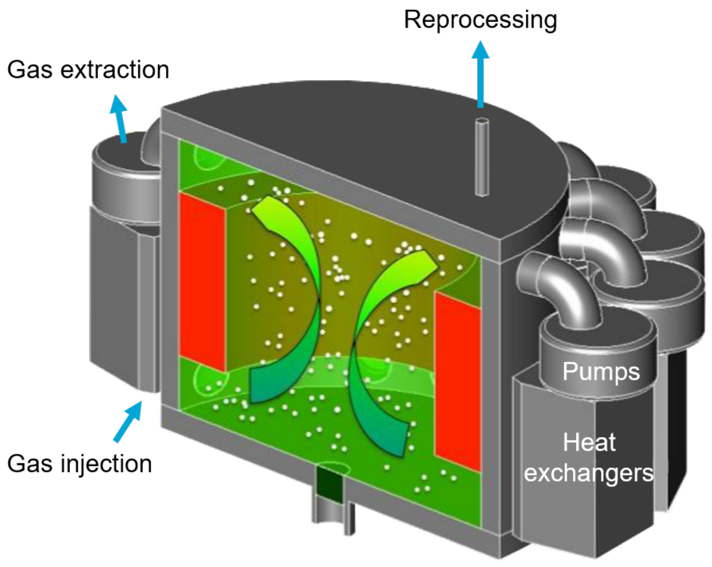


Figure 1.1: Schematic design of the MSFR core. The core (green) is surrounded by an annular breeding blanket (red). Molten salt is pumped through the heat exchangers which are located circumferentially around the vessel. [101]

1.1.2. Complications

The use of molten salts inside the reactor core enables many useful safety features, such as passive safety mechanisms, continuous reprocessing, and waste management. However, some drawbacks of using molten salts are related to the weakening of structural materials as a result of corrosion and irradiation embrittlement [63]. Another major issue originates from the relatively high melting points of the applied fuel salts, which makes molten salts prone to solidification when cooled down.

The same issue has been encountered in the solar industry, where molten salts are often used as a heat transfer fluid [97], resulting in the need for advanced solidification protection measures. Freezing can be dangerous primarily for two reasons: (1) it can cause damage to the equipment through phase change-induced volumetric expansion; and (2) it can clog the pipes and obstruct fluid circulation, which is necessary for heat removal. Both issues are even more challenging for molten salt reactors as the melting points of typical fluoride and chloride salts (approximately 450°C) are much higher compared to nitrate-based salts (approximately 150°C) used in solar energy applications [59].

By modeling the flow and heat characteristics of molten salts under different circumstances, design choices can be made to mitigate potential freezing risks. Such circumstances can generally be divided into the type of flow regime and the type of thermal conditions present in the reactor tubes. The flow regime is dictated by the Reynolds number, which is a quantity that relates the inertial forces to the viscous forces present in a fluid. When the Reynolds number is high, inertial forces are dominant and a flow is said to be turbulent; when the number is low, viscous forces dominate and a flow tends towards a laminar regime. Thermal conditions depend on both the cooling power inside the heat exchanger and the thermal properties of the salt. By balancing these characteristics carefully, the risk of solidification can effectively be reduced.

1.1.3. Research Focus

In this work, a numerical simulation model is developed to examine salt freezing in the heat exchanger leading to degraded heat transfer. The model simulates a fluid flowing through a 3D channel in a turbulent regime, while one of the walls is being cooled. The fluid dynamics is modeled using the lattice-Boltzmann method, which is a popular approach that describes the evolution of particle distributions on a discrete grid [56]. Furthermore, simulating detailed turbulent motion comes at a great computational cost. Therefore, simulations will be accelerated using a 'large eddy simulation' for the modeling of small turbulent scales and a Graphical Processing Unit will be leveraged to allow for parallel computations.

This thesis is structured as follows. In the remainder of this chapter, an introduction will be given to the MSFR concept (Sec. 1.1), a brief overview of recent work on computational fluid dynamics is presented (Sec. 1.2), and the research goals are discussed (Sec. 1.3). Chapter 2 gives an introduction to the theoretical concepts that will be used throughout this study. Subsequently, the adopted numerical methods are discussed in Chapter 3. The different grids that are used in the different simulations will be benchmarked in Chapter 4. In Chapter 5, the obtained turbulent flow results will be discussed and, lastly, the results of the performed freezing simulations will be presented in Chapter 6.

1.2. Previous Research

Relevant studies conducted in the past are presented below and will be utilized as a starting point for this research. Studies on the Filter Matrix LB method, LES, thermal LB, and phase change are listed (non-exhaustive).

Filter Matrix LB

- Somers, [98], summarized theoretical background of the FMLB method. They implemented LES in a 3D FMLB scheme with a derivative of the Smagorinsky SGS model.
- Eggels and Somers, [27], made a 2D numerical simulation of free laminar convective flow using the filter matrix lattice-Boltzmann scheme. In their model the subgrid-scale stress tensor τ was set to zero, corresponding to laminar flow.
- Rohde et al., [92], proposed a new hierarchical grid refinement technique that was tested for laminar and turbulent flows in the FMLB framework. Turbulent flows were modeled using a DNS.
- Zhuo and Zhong, [129], made an LES-based filter-matrix lattice-Boltzmann model for simulating fully developed turbulent channel flow. The Vreman-SGS model was adopted for the eddy viscosity calculation as proposed by [111].

Large Eddy Simulation in LB

- Meyers and Sagaut, [74], did a theoretical analysis of the model coefficients for the Smagorinsky model and two variational multi-scale variants of the Smagorinsky model. They proposed modifications to the different models that follow the behaviour of the 'exact' coefficients better.
- Mehta et al., [72], did an LES simulation using the Smagorinsky Model in combination with energy-conserving schemes. These EC schemes make sure that dissipation arises only from viscous effects, rather than spurious dissipations from discretization in space and time as well. The Smagorinsky model is explained briefly in this paper.
- Lévêque et al., [62], proposed a shear-improved Smagorinsky model that seeks to improve the overly dissipative nature near walls by subtracting the magnitude of the mean shear from the instantaneous strain-rate tensor when calculating the eddy viscosity.

- Nguyen et al., [78], very recently made a compressible LB-LES simulation of a developed jet leaving a tube and striking a flat heated plate. The LB model uses a hybrid formulation for mass, momentum, and energy conservation and a Hybrid Recursive Regularized (HRR) collision operator. The eddy viscosity was calculated following [62].
- Zhuo and Zhong, [129], see Ch. 1.2.

Thermal LB

- Zhuo et al., [131], introduced a D2Q9 filter-matrix lattice Boltzmann method and extended it to include incompressible thermal flows using a double-distribution function framework. The paper contains some derivation of the FMLB model, as well as the temperature equations for LB in 2D.
- Ren et al., [90], modeled 3D turbulent channel flow using a double-distribution function in the LBM-MRT framework. Additionally, a large eddy simulation was applied using the dynamic Vreman model.

Phase Change in LB

- Huang et al., [49], developed a solid-liquid phase LB method without the use of iteration steps or large groups of linear equations, yielding higher efficiency than iterative methods. The moving phase interface was treated by the immersed moving boundary scheme. The flow was laminar and a 2D9Q scheme has been adopted. The energy conservation equation is expressed in terms of the total enthalpy $H = C_pT + f_lL$, instead of ”sensible enthalpy” C_pT. Consequently, the temperature equilibrium distribution function is modified for the model to correspond to the correct macroscopic equations.
- Zhao et al., [126], established a 2D model composed of a liquid zone, a mushy zone, and a solid zone based on the improved enthalpy-porosity method. No LBM has been applied.
- Chakraborty and Chatterjee, [14], developed an LB model with a solid-liquid phase transition process. The phase-change modelling is enthalpy-based and a modified enthalpy-porosity method has been applied, in conjunction with an enthalpy-updating closure scheme.

Work that previous students in the affiliated research group ”Transport Phenomena in Nuclear Applications” performed is:

- Van Winden, [108], constructed a lattice Boltzmann finite difference model with phase change. He recommended the use of the double-distribution function LB model, because of its superior accuracy and stability compared to present hybrid LB models.
- Bus, [10], made a simulation of transient freezing in cooled non-eutectic molten salt laminar channel flow. The DDF FMLB was applied with a double distribution function. Phase change was modeled using the immersed boundary method and the enthalpy-porosity method.
- Besseling, [109], implemented Adaptive Mesh Refinement (AMR) in a DDF LB model of (1) natural convection of air in a square cavity, and (2) melting of gallium in a square cavity. Adaptive mesh refinement was investigated based on velocity, vorticity, and shear rate. The latter two methods showed the best accordance with the fully fine grid.
- Wortelboer, [119], conducted a DNS of 3D turbulent channel flow using the FMLBM. An extension with a double-distribution function was made to investigate thermal flows. Furthermore, the simulation was combined with a GPU implementation in Python.
- Van Bemmelen, [105], conducted a DNS of 3D turbulent channel flow using the FMLBM. It was applied to model the behavior inside semi-solid flow batteries. Furthermore, the simulation was combined with a GPU implementation in Python.

1.3. Research Goal

In this thesis, an answer will be formulated to the following set of research questions:

1. How can turbulent channel flow accurately be modeled using a large eddy simulation (LES) in the filter-matrix lattice Boltzmann (FMLB) framework?
2. How can the solidification of a non-eutectic fluid be modeled in a turbulent channel flow using an LES in the FMLB framework?

3. Which boundary conditions and input parameters should be chosen to arrive at a stable and accurate simulation of turbulent channel flow and phase change?
4. How does such a model compare to reference cases?

2 Theory

This chapter provides an overview of the theoretical concepts that underlie the techniques and models used in this thesis. Sec. 2.1 covers the fundamentals of fluid dynamics by discussing some of its governing equations and the concept of kinetic theory. Subsequently, the Lattice Boltzmann method is presented in Sec. 2.2, which is a numerical technique frequently used in fluid dynamics modeling. In sec. 2.3, important thermodynamical concepts are discussed that describe the behavior of multi-phase fluids. Then, turbulent flows will be described and the way they can be approximated using a turbulence model in Sec. 2.4 and 2.5. Lastly, the hardware components and working principles of a Graphical Processing Unit will be discussed in Sec. 2.6.

2.1. Fluid Dynamics

Before turbulent flow and heat transfer characteristics in 3D channels can be studied, it is necessary to understand some of the key concepts in fluid dynamics. In this section, a description of the governing principles of fluid dynamics will be given in the form of two conservation laws. Subsequently, an introduction to kinetic theory is provided, which is a convenient way of describing fluids for different modeling approaches.

2.1.1. Conservation Laws

The governing equations of fluid flow are a mathematical representation of physical conservation laws. The balancing equations for mass and momentum will be given in the following.

Conservation of Mass

A central principle in fluid dynamics is that no mass in a given system can be created or destroyed. This is known as the conservation of mass and leads to the continuity equation:

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}) = 0. \tag{2.1}$$

This equation states that the change of mass inside a region with density ρ is equal to the mass inflow into that region [88].

Conservation of Momentum

A second central principle is the conservation of momentum. It implies that the change in linear momentum of any control volume is equal to the total force exerted on that volume. By using that the total force is the sum of surface and body forces, one can arrive at Cauchy's equation of motion

$$\nabla \cdot \overline{\overline{\sigma}} + \rho \boldsymbol{f} = \rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right), \tag{2.2}$$

where the surface force is represented by stress tensor $\overline{\overline{\sigma}}$ and the body force by $\boldsymbol{f}$ [ms^{-2}] [80]. For incompressible flow and a Newtonian fluid (i.e., viscosity is not a function of shear rate), the components of $\overline{\overline{\sigma}}$ can be expressed in terms of dynamic viscosity μ, pressure p, and velocity u_i as

$$\sigma_{ij} = -p\delta_{ij} + 2\mu \left(S_{ij} - \frac{1}{3} S_{kk}\delta_{ij} \right), \tag{2.3}$$

where S_{ij} is the strain-rate tensor, given by

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right). \tag{2.4}$$

Note the use of Einstein notation in the above definitions.

Substituting Eq. 2.3 in Eq. 2.2 leads to the equation for conservation of momentum in an incompressible flow [88]:

$$\mu \nabla^2 \boldsymbol{u} - \nabla p + \rho \boldsymbol{f} = \rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \right). \tag{2.5}$$

The above equation and Eq. 2.1 are known as the Navier-Stokes equations for incompressible flows and constant dynamic viscosity. Together with an equation of state that relates the pressure and density of a system [46], the system of equations is closed and describes the time evolution of any fluid with the previously assumed characteristics.

Due to the non-linearity of Eq. 2.5, it is often extremely difficult or even impossible to find exact solutions to the Navier-Stokes equations [89]. However, there exist situations where the equations simplify to the extent that finding an analytical solution becomes possible. One such situation is the body force-driven steady-state parallel flow of incompressible fluid between parallel plates. Denoting the 1D body force with g, Eq. 2.5 simplifies to

$$\mu \frac{d^2 u}{dy^2} = -\rho g. \tag{2.6}$$

After assuming no-slip boundaries at $y = 0$ and $y = 2H$ with $u_y(H) = 0$, the obtained solution is called Poiseuille flow and can be expressed as

$$u(y) = \frac{g}{\nu} \left(Hy - \frac{y^2}{2} \right) \tag{2.7}$$

The kinematic viscosity ν that is used in Eq. 2.7 is related to the dynamic viscosity μ as

$$\nu = \frac{\mu}{\rho}. \tag{2.8}$$

Due to the availability of an analytical benchmark solution, Poiseuille flow is a widely used flow case for validating flow simulations.

2.1.2. Reynolds Number

The Reynolds number is used to characterize the flow regime and describes the ratio between inertial and viscous forces present in a flow [80]. It is a non-dimensional number that emerges after non-dimensionalization of the Navier-Stokes equations and is defined as

$$Re = \frac{UL}{\nu}, \tag{2.9}$$

where U is the characteristic velocity of the flow, L is the characteristic length scale, and ν is the kinematic viscosity. The choice of U and L depends on the type of Reynolds number that is used. In channel flows, it is common to define the bulk Reynolds number Re_m and the shear Reynolds number Re_τ. Their formal definitions will be given in 2.4.2. The laminar regime generally corresponds to $Re_m < 2300$, while the fully turbulent regime corresponds to $Re_m > 2700$. However, the exact boundaries depend on the flow conditions, such as smoothness of the walls and the present geometry. In between these boundaries there exists a transitional regime in which the flow is considered neither fully turbulent nor laminar. There will be a more extensive description of turbulence in Sec. 2.4.

2.1.3. Kinetic Theory

Fluids can generally be modeled on three different scales: the miscroscopic, mesoscopic and macroscopic scale. Physically, fluids are large sets of atoms or molecules that collide with one another and follow random trajectories. The microscopic description aims to track all these individual particle motions, which quickly comes at an enormous computational cost when moving to macroscopic situations. An alternative approach is to consider fluids as being continuously distributed throughout the domain, describing their macroscopic behavior in terms of density, velocity, and temperature. Such an approach attempts to solve the equations of fluid dynamics directly, applying generic numerical methods such as finite difference or finite volume methods [56].

The mesoscopic scale is an intermediate scale in which fluids are represented by distributions or collections of molecules that interact with one another and are tracked in time. One class of mesoscopic methods solves so-called kinetic equations with some numerical scheme, such as finite differences, the gas kinetic scheme, or the Lattice Boltzmann method [43]. Kinetic equations are conservation equations that describe how probability density functions of stochastic systems evolve in time [128]. The Boltzmann equation is one such equation and will be discussed in the following.

Boltzmann Equation

Kinetic theory describes fluids through the use of a probability density function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ that depends on position x, microscopic particle velocity $\boldsymbol{\xi}$, and time t. This function determines the probability $f d\boldsymbol{x} d\boldsymbol{\xi}$ that a particle is located within the infinitesimal volume $(\boldsymbol{x}, \boldsymbol{x} + d\boldsymbol{x})$ with velocity $(\boldsymbol{\xi}, \boldsymbol{\xi} + d\boldsymbol{\xi})$. The rate of change of the particle density function df/dt can be decomposed as

$$\frac{df}{dt} = \left(\frac{\partial f}{\partial t}\right)\frac{dt}{dt} + \left(\frac{\partial f}{\partial x_i}\right)\frac{dx_i}{dt} + \left(\frac{\partial f}{\partial \xi_i}\right)\frac{d\xi_i}{dt} \tag{2.10}$$

Looking at the terms from left-to-right, we have $dt/dt = 1$, the particle velocity $dx_i/dt = \xi_i$, and the specific body force $d\xi_i/dt = g_i$ which has units $[F/\rho]$ =N/kg. We use $df/dt \equiv \Omega(f)$ on the left-hand side, apply vector notation, and arrive at the Boltzmann equation

$$\frac{\partial f(\boldsymbol{x}, \boldsymbol{\xi}, t)}{\partial t} + \boldsymbol{\xi} \cdot \nabla f(\boldsymbol{x}, \boldsymbol{\xi}, t) + \nabla_{\boldsymbol{\xi}} f(\boldsymbol{x}, \boldsymbol{\xi}, t) \cdot \boldsymbol{g} = \Omega(f). \tag{2.11}$$

This equation describes the transportation of f and has the form of an advection equation. The first two terms describe the advection with velocity ξ and the third term describes the influence of forces on particle velocities. The term Ω represents the rate of change in f due to binary molecular collisions and is known as the collision operator. Its formal definition is a double integral over velocity space that considers all outcomes of two-particle collisions for a choice of intermolecular forces. The macroscopic quantities can be obtained from the distribution function through the following integrals:

$$\rho = \int f d^3\boldsymbol{\xi}, \tag{2.12}$$

$$\rho u_i = \int \xi_i f d^3\boldsymbol{\xi}. \tag{2.13}$$

These quantities are also referred to as the moments of f.

The Navier-Stokes equations can be retrieved from the continuous Boltzmann equation by applying a so-called Chapman-Enskog analysis [56]. This analysis applies a decomposition of the distribution function into an equilibrium and a non-equilibrium part. The equilibrium distribution is the state a system relaxes to when it is left alone for a sufficiently long time and the non-equilibrium part is the perturbation from this state [43]. When $f = f^{eq}$, the collision takes no net effects, giving $\Omega(f^{eq}) = 0$. For an ideal gas, the equilibrium distribution is given by the Maxwell-Boltzmann distribution

$$f^{eq}(\boldsymbol{x}, \boldsymbol{\xi}, t) = \rho \left(\frac{1}{2\pi RT}\right)^{3/2} e^{-|\boldsymbol{\xi}|^2/(2R_G T)}. \tag{2.14}$$

2.2. Lattice Boltzmann Method

The Lattice Boltzmann Method (LBM) is a relatively new numerical technique that solves a simplified version of the Boltzmann equation on a discrete grid [91]. The groundwork of this method lies in the discretization of the Boltzmann equation that leads to the formulation of the lattice Boltzmann equation (LBE), which will be discussed in Sec. 2.2.1. The stream-and-collide algorithm that is a central concept in LBM simulations is explained in Sec. 2.2.2 and the most general collision operator, the BGK operator, is dicussed in Sec. 2.2.3.

2.2.1. Discretizing the Boltzmann Equation

We will now derive the lattice Boltzmann equation by discretizing the Boltzmann equation. To this end, consider the force-free Boltzmann equation

$$\frac{\partial f(\boldsymbol{x}, \boldsymbol{\xi}, t)}{\partial t} + \boldsymbol{\xi} \cdot \nabla f(\boldsymbol{x}, \boldsymbol{\xi}, t) = \Omega(f). \tag{2.15}$$

As a starting point, one can discretize the velocity space $\boldsymbol{\xi}$ into a finite set of velocities $\{c_i\}$ and define corresponding weights w_i [43]. Subsequently, a discrete distribution function $f_i(\boldsymbol{x}, t) = \omega_i f(\boldsymbol{x}, \boldsymbol{c}_i, t)$ can be introduced. This function satisfies the following equation:

$$\frac{\partial f_i}{\partial t} + \boldsymbol{c}_i \cdot \nabla f_i = \Omega(f), \tag{2.16}$$

Density and momentum can now be obtained from the discrete distribution function as

$$\rho = \sum_i f_i, \quad \rho \boldsymbol{u} = \sum_i \boldsymbol{c}_i f_i. \tag{2.17}$$

Integrating Eq. 2.16 from t to $t + \Delta t$ along the characteristic line $\boldsymbol{x}(s) = \boldsymbol{x} + \boldsymbol{c}_i s$ using the method of characteristics [45], and assuming a constant collision term over this interval, yields

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = \Delta t \Omega(f). \tag{2.18}$$

This equation is known as the lattice Boltzmann equation. The population $f_i(\boldsymbol{x}, t)$ now represents the particle number density for velocity direction $\boldsymbol{c}_i$ at grid node $\boldsymbol{x}$ and time t. Collision operator Ω_i now relaxes the system towards the discretized equilibrium distribution, which is given by

$$f_i^{eq}(\boldsymbol{x}, t) = w_i \rho \left(1 + \frac{\boldsymbol{u} \cdot \boldsymbol{c_i}}{c_s^2} + \frac{(\boldsymbol{u} \cdot \boldsymbol{c_i})^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2}\right), \tag{2.19}$$

where c_s represents the lattice speed of sound and w_i are weights specific to the used velocity set [56]. Velocity sets come in different forms and can be classified based on the number of dimensions d and the number of discrete velocities q, using the notation DdQq. Typically, a higher amount of velocities leads to a higher accuracy, but also a greater computational cost. Common velocity sets used in three dimensions are D3Q15, D3Q19 and D3Q27 [56].

2.2.2. Stream-and-collide Algorithm

When taking a closer look at the terms in the lattice Boltzmann equation (Eq. 2.18), two separate processes can be observed:

1. The first process is a collision,

$$f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x}, t), \tag{2.20}$$

   where f_i^* denotes the post-collision distribution. The pre-collision distributions are redistributed, based on their current values and the present macroscopic variables
2. The second process is streaming (or propagation)

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) = f_i^*(\boldsymbol{x}, t), \tag{2.21}$$

   where post-collision distribution f_i^* is streamed to a neighboring site in the lattice.

This illustrates the straightforwardness of the LB algorithm. Each time step consists of a collision step and a consecutive propagation step, which are repeated at every node in the lattice for the remainder of the simulation.

2.2.3. BGK Collision Operator

The collision operator Ω_i can be chosen in various ways. The simplest and most widely applied form is the Bhatnagar-Gross-Krook (BGK) operator,

$$\Omega_{BGK} = -\frac{1}{\tau}(f_i - f_i^{eq}). \tag{2.22}$$

This operator assumes the relaxation of populations f_i to settle towards equilibrium f_i^{eq} at a rate that is determined by the relaxation time τ. Despite its wide application in various fields, the BGK operator is reported to suffer from stability issues due to non-physical terms that arise from the discretization of the Boltzmann equation. Numerous solutions have been proposed to alleviate this problem. These can generally be categorized into changing the numerical discretization, the collision scheme, or both of them [17].

2.3. Thermodynamics

Because heat transfer will be studied in fluids that are subject to freezing, a thermodynamical description of fluids will now be given that is suitable for situations with phase change. To this end, the conservation of total enthalpy will be discussed in Sec. 2.3.1, and a description of the concepts behind phase change is given in Sec. 2.3.2.

2.3.1. Conservation of Enthalpy

Apart from mass and momentum conservation in Sec. 2.1.1, one can also define an equation that balances the sensible enthalpy $h = C_p T$ in a confined system and results from the conservation of energy. Assuming no viscous dissipation and shockwaves, constant density, constant specific heat capacity, and applying Fourier's law, the sensible enthalpy equation for a specific phase ϕ can be written as

$$\rho^{\phi}\frac{\partial h^{\phi}}{\partial t} + \rho u_i \frac{\partial h^{\phi}}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\alpha^{\phi}\frac{\partial h^{\phi}}{\partial x_i}\right) + q''', \tag{2.23}$$

where q''' is a local heat source term that represents the addition or release of energy due to melting or solidification and α is the thermal diffusivity [11, 110]. Similar to the Navier-Stokes equation, Eq. 2.23 contains convective and diffusive terms.

One can define a non-dimensional number that expresses the ratio between momentum diffusivity and thermal diffusivity. It is known as the Prandtl number Pr and is defined as

$$Pr = \frac{\nu}{\alpha}. \tag{2.24}$$

A high Prandtl number $Pr \gg 1$ leads to a situation where diffusion of momentum is dominant over diffusion of heat. At low Prandtl number $Pr \ll 1$, the opposite is true [52].

2.3.2. Phase Change

When a liquid changes to a solid state, there is no instant transition. This is because the change in molecular structure is associated with a change in energy that is given by the latent heat of fusion L of phase change [110]. A measure of energy that takes into account this latent heat contribution is the total enthalpy H, which can be expressed as

$$H^{\phi} = h^{\phi} + f_l^{\phi} L. \tag{2.25}$$

The quantity f_L^{ϕ} is the liquid fraction, which expresses how close a medium is to the solid or liquid phase. A value $f_l^{\phi} = 1$ represents a fully liquid phase, while 0 represents a fully solid phase. When $0 < f_l < 1$, the substance is said to be in the so-called mushy region. This is an intermediate phase in which the growing solid phase forms a porous matrix through which the liquid can flow [118]. Convection is still possible in a mushy region, but the fluid has different properties than in the fully liquid phase. The relevant phases are now the liquid, solid, and mushy phases, $\phi = \{l, s, m\}$.

Upon substituting $h^{\phi} = H^{\phi} - f_l^{\phi} L$ into Eq. 2.23, and assuming negligible advection of latent heat such that $\boldsymbol{u} \cdot \nabla H^{\phi} \approx \boldsymbol{u} \cdot \nabla h^{\phi}$, one obtains the following total enthalpy balance equation:

$$\frac{\partial H^{\phi}}{\partial t} + u_i \frac{\partial h^{\phi}}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\alpha^{\phi}\frac{\partial h_i^{\phi}}{\partial x_i}\right). \tag{2.26}$$

Note that q''' does not appear in this equation anymore, because it has canceled with the latent heat contribution in the transient term.

The transition from liquid to mushy takes place at the liquidus total enthalpy H_l and the transition from mushy to solid takes place at the solidus total enthalpy H_s. The liquid fraction can be defined in terms of H, H_l and H_s,

$$f_l = \begin{cases} 0 & H < H_s \\ \frac{H-H_s}{H_l-H_s} & H_s \leqslant H \leqslant H_l \\ 1 & H > H_l. \end{cases} \tag{2.27}$$

Similar to H_s and H_l, it is also possible to define a solidus and liquidus temperature. These are the temperatures at which a substance enters the solid or liquid phase, respectively. Materials that have equal solidus and liquidus temperatures $T_s = T_l$ are called eutectic materials. On the other hand, non-eutectic materials have different temperatures $T_s < T_l$, causing the latent heat evolution during phase change to take place over a range of temperatures [110]. Following [49], this leads to the following expression of temperature T in terms of total enthalpy H:

$$T = \begin{cases} H/C_{p,s} & H < H_s \\ T_s + \frac{H-H_s}{H_l-H_s}\left(T_l - T_s\right) & H_s \leqslant H \leqslant H_l \\ T_l + \left(H - H_l\right)/C_{p,l} & H > H_l \end{cases}. \tag{2.28}$$

Typically, molten salts are non-eutectic mixtures and, therefore, their phase change trajectory is bound by a temperature range rather than a single temperature.

Thermal variables such as the specific heat capacity C_p often have different values in different phases. This complicates the definition of the sensible enthalpy, because the definition $h = C_pT$ does not anymore guarantee a unique temperature for one value of h. To this end, a more universal definition of h can be used that is valid in all phases:

$$h = \begin{cases} C_{p,s}T & T < T_s \\ h_s + C_{p,m}T & T_s \leqslant T \leqslant T_l \\ h_l + C_{p,l}T & T > T_l \end{cases}. \tag{2.29}$$

Here, C_p, m is the specific heat capacity in the mushy region and h_s and h_l are the solidus and liquidus sensible enthalpies, respectively. They are given by

$$h_s = C_{p,s}T, \tag{2.30}$$

$$h_l = h_s + C_{p,m}(T_l - T_s). \tag{2.31}$$

2.4. Turbulent Flows

In a turbulent flow, the velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$ exhibits seemingly random behavior. When multiple realizations of the same flow experiment are performed, one obtains different outcomes for the instantaneous flow field, as opposed to a laminar field in which the flow is constantly aligned. Paradoxically, $\boldsymbol{u}$ is fully deterministic in a turbulent flow and the field is not completely random. However, turbulent flow fields just have an inherent sensitivity to only the slightest perturbations. Any flow has unavoidable perturbations in initial conditions, boundary conditions, and material properties, which cause it to diverge from its unperturbed (or differently perturbed) state [86].

The statistical description of turbulent flows is discussed in Sec. 2.4.1. Subsequently, the concepts and quantities that are used to classify channel flows will be described in Sec. 2.4.2 and 2.4.3. Sec. 2.4.4 takes a more qualitative standpoint and gives an understanding of the different scales of motion in turbulent flows.

2.4.1. Turbulent Statistics

A turbulent variable, such as the flow field $\boldsymbol{u}$, can be decomposed into its mean $\overline{\boldsymbol{u}}$ and the fluctuation $\boldsymbol{u'}$, which is referred to as a Reynolds decomposition. The same can be done for any turbulent variable, such as temperature, leading to the following relations:

$$\begin{aligned} \boldsymbol{u} &= \overline{\boldsymbol{u}} + \boldsymbol{u}', \\ T &= \overline{T} + T'. \end{aligned} \tag{2.32}$$

Mean Quantities

The mean of a quantity is defined as the ensemble average over many realizations N of the same experiment, such that

$$\overline{\boldsymbol{u}}(\boldsymbol{x}, t) = \frac{1}{N} \sum_{\alpha=0}^{N} \boldsymbol{u}^{(\alpha)}(\boldsymbol{x}, t), \tag{2.33}$$

where the index α labels the realization of the experiment.

In flow simulations, it is often computationally too expensive to perform the same experiment a large number of times. Therefore, it is common to take a time average $\overline{\boldsymbol{u}}^T$ of only one experiment over an interval $[0, T]$, which is defined as

$$\overline{\boldsymbol{u}}^T(\boldsymbol{x}, t) = \int_0^T \boldsymbol{u}(\boldsymbol{x}, t) dt. \tag{2.34}$$

Similarly, it is also possible to take a spatial average $\overline{\boldsymbol{u}}^L$ over an interval $[0, L]$. In the case of 3D turbulent flow between parallel plates that span the x-z plane, the spatial average of the velocity field at position y would be defined as

$$\overline{\boldsymbol{u}}^L(y, t) = \int_0^{L_x} \int_0^{L_z} \boldsymbol{u}(\boldsymbol{x}, t) dx dz. \tag{2.35}$$

A process is called stationary if it is statistically invariant to translations in time and homogeneous if it is statistically invariant to translations in space. For such processes, it holds that

$$\lim_{T \to \infty} \overline{\boldsymbol{u}}^T = \overline{\boldsymbol{u}}, \tag{2.36}$$

$$\lim_{L \to \infty} \overline{\boldsymbol{u}}^L = \overline{\boldsymbol{u}}. \tag{2.37}$$

Developed turbulent flow between parallel plates is both stationary and homogeneous in the stream- and span-wise directions. It is therefore suitable to take time and spatial averages to determine the means of the flow variables.

However, in flow simulations, there is often no continuous flow field that can be integrated over. Instead, the velocity is defined on discretely spaced points $\boldsymbol{x}_i$ in space and at discrete time steps t_i. It is then convenient to define $\overline{\boldsymbol{u}}^T$ and $\overline{\boldsymbol{u}}^L$ in terms of a discrete sum over time and space, respectively [80]:

$$\overline{\boldsymbol{u}}^T(\boldsymbol{x}, t) = \frac{1}{N} \sum_{i=0}^{N} \boldsymbol{u}(\boldsymbol{x}, t_i), \tag{2.38}$$

$$\overline{\boldsymbol{u}}^L(\boldsymbol{x}, t) = \frac{1}{N} \sum_{i=0}^{N} \boldsymbol{u}(\boldsymbol{x}_i, t). \tag{2.39}$$

The selection of N should ensure a sufficiently large number of uncorrelated points and snapshots of the flow field. This is necessary to guarantee the convergence of Eqs. 2.38 and 2.39 to the ensemble average.

Fluctuating Quantities

The fluctuation of a scalar field, such as $u' = u - \overline{u}$ or $T' = T - \overline{T}$, is often described in terms of its root mean square (RMS) value, e.g.,

$$u_{rms} = \sqrt{\overline{u'u'}}, \tag{2.40}$$

$$T_{rms} = \sqrt{\overline{T'T'}}. \tag{2.41}$$

which is a widely used metric to benchmark the statistics of turbulent flows. The RMS velocity fluctuation u_{rms} is proportional to the turbulent kinetic energy k, which in a 3D channel with velocity components u_i, is given by

$$k = \frac{1}{2}\overline{u_i'u_i'}. \tag{2.42}$$

When Eq. 2.42 is normalized by the squared mean velocity, one obtains the turbulent intensity i [80]:

$$i = \frac{k}{\|\overline{\boldsymbol{u}}\|^2}. \tag{2.43}$$

Reynolds Stress and Turbulent Heat Flux

A statistic that is often used to describe the turbulent channel flows is the Reynolds stress. It is denoted by $-\rho\overline{u_i'u_j'}$ and represents the stress contribution from turbulent fluctuations. In Sec. 2.5.1, it will be explained how the Reynolds stress emerges from the Reynolds-averaged Navier-Stokes equations.

The analogous quantity in the thermal description is the turbulent heat flux $-\overline{u_j'T'}$, which represents an additional heat flux due to turbulent fluctuations. Similar to the Reynolds stress, it is a turbulent statistic that is commonly used in the description of turbulent channel flows. A more extensive description is given in Sec. 2.5.1.

2.4.2. Channel Flows

This research is focused on describing the behavior of turbulent flows passing through a channel. To describe the characteristics of such a flow, a rectangular duct of height $h = 2H$ will be considered in the remainder of this chapter, as sketched in Fig. 2.1. The duct is infinitely long and assumed to be very wide ($b/H \gg 1$), such that it can be approximated as a set of infinite parallel planes. The flow propagates in the x-direction and the mean velocity varies in the y-direction, assuming zero velocity at the boundaries. The bottom and top walls are positioned at $y = 0$ and $y = 2H$, respectively, with the mid-plane being $y = H$. The three velocity components are denoted by $(u, v, w) = (u_1, u_2, u_3)$.

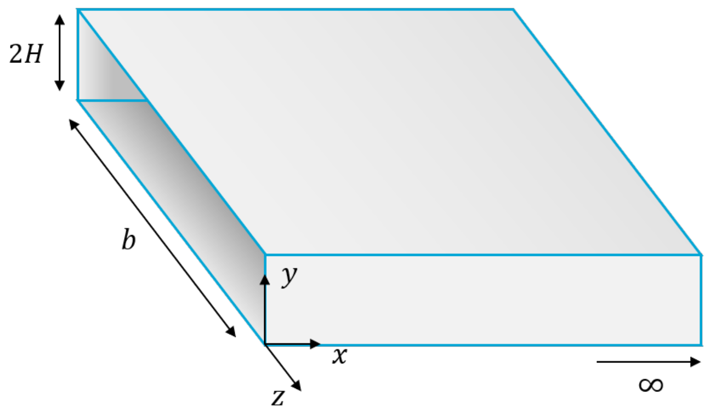


Figure 2.1: A rectangular duct with half-height H and depth $b \gg H$.

As was mentioned in Sec. 2.1.2, a definition of the Reynolds number that is commonly used to describe flow regimes in channel flows is the bulk Reynolds number Re_m,

$$Re_m \equiv \frac{\overline{u}_m 2H}{\nu}. \tag{2.44}$$

Here, the length scale is given by the full-channel height $2H$ and the characteristic velocity is given by the bulk mean velocity $\overline{u}_m$. The latter is defined as

$$\overline{u}_m = \frac{1}{2H}\int_0^{2H} \overline{u} dy. \tag{2.45}$$

Wall Units

We will now dive further into a convenient and universal way to classify and describe turbulent channel flows, following the descriptions in [86, 90]. To this end, we introduce the total shear stress $\tau(y)$ in the channel, which is defined as

$$\tau(y) = \rho\nu\frac{d\overline{u}}{dy} - \rho\overline{u'v'}. \tag{2.46}$$

The first term is the viscous stress and the second term is the so-called Reynolds stress, which is the stress contribution from turbulent fluctuations. The origin and physical description of Reynolds stress will be discussed in Sec. 2.5.1.

Close to the walls, the Reynolds stress should vanish due to the no-slip condition $u_w = 0$. Thus, the wall shear stress at $y = 0$ is exclusively determined by viscous forces:

$$\tau_w \equiv \rho\nu\left(\frac{d\overline{u}}{dx}\right)_{y=0}. \tag{2.47}$$

It is convenient to define appropriate velocity and length scales in the near-wall region that allow for a more universal description of channel flows, based on viscosity ν and wall shear stress τ_w. These are the friction velocity,

$$u_\tau \equiv \sqrt{\frac{\tau_w}{\rho}}, \tag{2.48}$$

and the viscous length scale,

$$\delta_\nu \equiv \nu\sqrt{\frac{\rho}{\tau_w}} = \frac{\nu}{u_\tau}. \tag{2.49}$$

With these viscous scales, it is possible to define the friction Reynolds number, which is a widely used universal turbulence measure in channel flow experiments,

$$Re_\tau \equiv \frac{u_\tau H}{\nu} = \frac{H}{\delta_\nu}. \tag{2.50}$$

The non-dimensional quantity that is used to express relative distance to the wall, is denoted by y^+ and is measured in viscous lengths δ_ν,

$$y^+ \equiv \frac{y}{\delta_\nu} = \frac{y}{\delta_\nu} = \frac{u_\tau y}{\nu}. \tag{2.51}$$

Similarly, a non-dimensional velocity u^+ and time t^+ can be defined as

$$u^+ \equiv \frac{\overline{u}}{u_\tau}, \tag{2.52}$$

$$t^+ \equiv \frac{u_\tau}{H}t. \tag{2.53}$$

This framework can also be extended to the thermal description of channel flows. The non-dimensional temperature T^+ is obtained after a normalization

$$T^+ \equiv \frac{T}{T_f}, \tag{2.54}$$

where T_f is the friction temperature, which is defined as

$$T_f \equiv -\frac{\alpha}{u_\tau}\left(\frac{\partial\overline{T}}{\partial y}\right)_{y=0}. \tag{2.55}$$

The friction temperature depends on thermal diffusivity α and mean temperature gradient $\partial \overline{T}/\partial y$ at the wall.

These quantities are referred to as wall units and provide useful non-dimensional descriptions of any type of channel flow. An example of such a universal description will be given in the following section.

2.4.3. Wall Flow Regions

Wall-bounded turbulent flows exhibit universal mean streaming profiles within specified ranges of y^+. According to Prandtl's postulation, the mean velocity near the wall is exclusively influenced by the viscous scales, particularly at sufficiently high Reynolds numbers. Consequently, the mean velocity is solely dependent on the distance to the wall, leading to the so-called law of the wall:

$$u^+ = f_w(y^+), \tag{2.56}$$

where the function $f_w(y^+)$ is universal for Reynolds numbers that are well beyond the transitional regime (Re $\gtrsim 3000$) [86]. The form of $f_w(y^+)$ can be determined for several ranges of y^+, as will be shown in the following.

The Viscous Sublayer

The viscous sublayer is the region that is nearest to the wall and is located at $y^+ < 5$. In this region, the no-slip condition at the wall yields a value $u^+(0) = 0$. The derivative at the wall can be obtained from Eq. 2.47 after normalizing with δ_ν. This leads to

$$\frac{du^+}{dy}(0) = 1. \tag{2.57}$$

After applying the definition of the Taylor-series expansion to Eq. 2.57 for small values y^+, and neglecting higher order terms $\mathcal{O}(y^{+4})$, one obtains the following average velocity profile in the viscous sublayer [86]:

$$u^+(y^+) = y^+. \tag{2.58}$$

The Log-law Region

The log-law region is located at $y^+ > 30,\ y/H < 0.3$ and is also a region for which $f_w(y^+)$ can be well estimated. As the distance to the wall becomes greater, the relative contribution of viscous stress to the total shear stress in Eq. 2.46 becomes negligibly small compared to the contribution of Reynolds stress, leading to

$$\tau(y) \approx -\rho\overline{u'v'}. \tag{2.59}$$

It has also been experimentally observed that the shear stress is approximately constant near the wall, such that $\tau(y) \approx \tau_w$. Recalling the definition of the friction velocity in Eq. 2.48, one then obtains

$$u_\tau^2 \approx -\overline{u'v'}. \tag{2.60}$$

The Reynolds stress $-\overline{u'v'}$ can be modeled using Prandtl's mixing length theory [50], which estimates the Reynolds shear stress as

$$-\overline{u'v'} = \ell^2 \left|\frac{\partial \overline{u}}{\partial y}\right| \frac{\partial \overline{u}}{\partial y}, \tag{2.61}$$

where the characteristic size ℓ of eddies scales with distance from the wall according to

$$\ell = ky. \tag{2.62}$$

Here, k is the Von Kármán constant with a value $k \approx 0.4$. The combination of Eqs. 2.60 and 2.61 leads to a solution for the mean velocity that can be written as

$$u^+ = \frac{1}{k}\ln y^+ + B, \tag{2.63}$$

where B is a constant with value $B \approx 5.2$ that depends on the Reynolds number that is used [80]. For high Reynolds numbers (e.g., $Re_\tau = 395$), the value is closer to 5, while for low Reynolds numbers the value is closer to 5.5 (e.g., $Re_\tau = 180$) [55].

As mentioned earlier, the log-law region stretches until $y/H = 0.3$. However, even at the center of the channel, where the arguments leading to Eq. 2.63 no longer apply ($\tau(y) \neq$ constant), deviations from the log law are still quite small [86].

2.4.4. Scales of Turbulent Motion

Turbulent flow consists of eddies, which are vortex-like turbulent structures with varying length scales ℓ. The largest scale eddies are characterized by length scale ℓ_0, velocity u_0 and timescale $\tau(\ell_0) \equiv \ell_0/u(\ell)$. They are comparable in size with the flow domain, meaning

$$\ell_0 \approx H. \tag{2.64}$$

The large eddies are unstable and break up into smaller eddies, transferring their kinetic energy. The smaller eddies undergo a similar break-up process, which is repeated until the smallest length scale η is reached. At this scale, kinetic energy is dissipated into heat, resulting from viscous forces that dominate over inertial forces. Thus, energy is transported from large to small scales, or equivalently, from small to large wavenumbers κ, which can be defined as

$$\kappa = 2\pi/\ell. \tag{2.65}$$

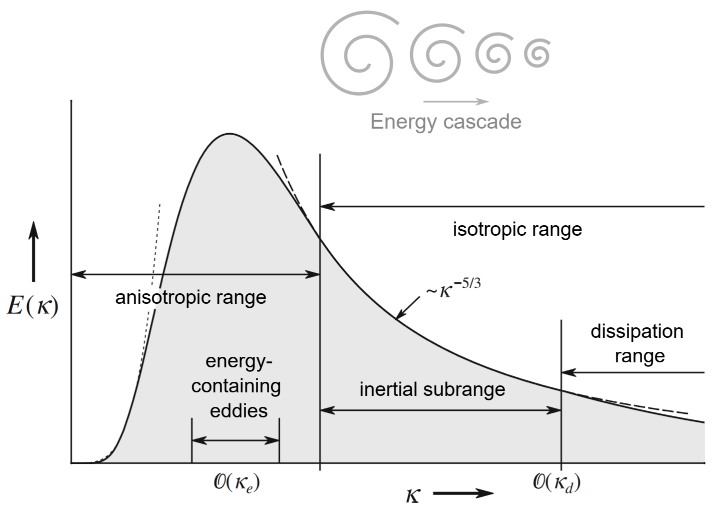


Figure 2.2

The largest-scale eddies draw their energy directly from the mean motion, leading to a strong directional preference of their flow statistics. Consequently, these large turbulent structures are said to be highly anisotropic. Upon transitioning to considerably smaller scales $\ell \ll \ell_0$, provided a sufficiently high Reynolds number, the turbulent structures lose their directional preference and become isotropic. This is known as Kolmogorov's hypothesis of local isotropy and the applicable range of scales is referred to as the isotropic range [86]. This range can be categorized into the dissipation range and the inertial subrange. As the names suggest, energy is dissipated in the dissipation range, while inertial forces dominate in the inertial subrange. A schematic overview of the different turbulent scales and corresponding energy spectra $E(\kappa)$ is given in Fig. 2.4. It can be seen that most energy is contained in large-scale eddies with wavenumbers $\kappa \approx \kappa_e = 2\pi/H$. These are the large-scale eddies that are on the order of the domain size. In the inertial subrange, the energy contained in the eddies gradually decays with $\kappa^{-5/3}$. It is also the range where energy is transferred to successively smaller scales, following the energy cascade. The dissipation range is approached when wavenumbers reach a value $\kappa \approx \kappa_d = O(2\pi/\eta)$. [80, 71]

According to Kolmogorov's first similarity hypothesis, the small-scale motions in the dissipation range take a universal form that is exclusively determined by viscosity ν and the energy dissipation rate ϵ. These two parameters can be used to derive unique length, velocity, and time scales for the smallest, dissipative eddies. Respectively,

$$\eta \equiv \left(\frac{\nu^3}{\epsilon}\right)^{1/4}, \quad u_\eta \equiv (\epsilon\nu)^{1/4} \quad \tau_\eta \equiv (\epsilon\nu)^{1/4}. \tag{2.66}$$

The energy dissipation rate ϵ is approximately equal to the production of energy at large scales. Because the latter scales as $u_0^2/\tau_0 = u_0^3/\ell_0$ [86], the ratios between the smallest and the largest scales can be expressed as

$$\frac{\eta}{\ell_0} \sim Re^{-3/4}, \quad \frac{u_\eta}{u_0} \sim Re^{-1/4}, \quad \frac{\tau_\eta}{\tau_0} \sim Re^{-1/2}. \tag{2.67}$$

It can be concluded from the above relations that the scales of the smallest eddies decrease with increasing Reynolds number. Consequently, when it is desired to simulate all scales of a high-Re flow (i.e., Direct Numerical Simulation), it is necessary to have a high grid resolution. This comes at a large computational cost, so there exists great interest in modeling approaches that do not need to resolve all turbulent scales. One such approach will be introduced in the next section.

2.5. Turbulence Modeling

Turbulence models are methods that aim to include the effects of turbulence in simulations of fluid flows. It is possible to directly solve the governing equations without the use of any turbulence model. This approach is known as a Direct Numerical Simulation (DNS) and it requires a very fine grid that captures the full range of turbulent motion. Especially for high Reynolds numbers, when turbulent scales become smaller, this approach is not feasible as computation times quickly become too large. An alternative method lies in the Reynolds-Averaged Navier-Stokes (RANS) approach, which effectively models all turbulent scales. A third method is the Large Eddy Simulation (LES), which takes an intermediate approach and models only the small turbulent motions. The latter two approaches will be discussed in Sec. 2.5.1 and 2.5.2, respectively.

2.5.1. RANS Approach

The RANS approach relies on a Reynolds decomposition $u = \overline{u} + u'$ that was discussed in Sec. 2.4.1. This decomposition can be inserted in the equations of motion of Sec. 2.1.1, i.e., the continuity equation (Eq. 2.1) and the Navier-Stokes equation (Eq. 2.5). Subsequent averaging of the continuity equation for incompressible flows and applying the so-called ”Reynolds conditions” [80] for the averaging operators, yields

$$\frac{\partial \overline{u_i}}{\partial x_i} = 0, \tag{2.68}$$

$$\frac{\partial u_i'}{\partial x_i} = 0. \tag{2.69}$$

This implies that both the average and fluctuating velocity fields are divergence-free. A similar procedure can be performed for the Navier-Stokes equation to arrive at the Reynolds-Averaged Navier-Stokes equations (RANS):

$$\rho\frac{\partial \overline{u}_i}{\partial t} + \rho\frac{\partial \overline{u}_i\overline{u}_j}{\partial x_j} = \rho\overline{f}_i + \frac{\partial}{\partial x_j}\left[\overline{p}\delta_{ij} + \mu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) - \rho\overline{u_i'u_j'}\right], \tag{2.70}$$

where $-\rho\overline{u_i'u_j'}$ is referred to as the Reynolds stress tensor[80]. This tensor is composed of nine Reynolds stresses for each combination of i and j, of which only six are unique. The term ’stress’ is being used here, first of all, because the dimensions of $\rho\overline{u_i'u_j'}$ are force per unit area. Secondly, the last two terms in Eq. 2.70 can be grouped, such that $-\rho\overline{u_i'u_j'}$ becomes the turbulent equivalent of the viscous term $\mu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)$ [61]. Physically, Reynolds stress can be interpreted as the transfer of momentum in the i-direction, due to turbulent fluctuations in the j-direction. If $i = j$, the stress components are called normal stresses, while for $i \neq j$, they are called shear stresses.

Analogously, one can apply a Reynolds-averaging approach to the total enthalpy balancing equation (Eq. 2.26). The resulting Reynolds-averaged total enthalpy equation yields

$$\frac{\partial \overline{H}}{\partial t} + \overline{u_j}\frac{\partial \overline{h}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\alpha \frac{\partial \overline{h}}{\partial x_j} - C_p \overline{u_j' T'}\right], \tag{2.71}$$

where $-\overline{u_j' T'}$ is referred to as the turbulent heat flux [61]. It can be seen as the thermal equivalent of the Reynolds stress.

The Reynolds stress and turbulent heat flux are generally unknown quantities as turbulent structures are hard to predict, in contrast to the averaged quantities. A central topic in the theory of turbulence is to predict these unknown quantities by relating them to the known, averaged quantities, and so, get a mathematically closed problem [80].

Eddy-Viscosity Assumption

One way to relate Reynolds stress to the averaged quantities is through an eddy-viscosity assumption. This approach was first proposed by Boussinesq in 1877 [9], and it is therefore also referred to as the Boussinesq hypothesis. Analogous to the viscous stress term in Eq. 2.70, he suggested expressing the Reynolds stress in terms of the mean velocity gradients and a factor of proportionality, known as the eddy-viscosity $\mu_t = \rho\nu_t$. This led to the representation of the Reynolds stress as

$$-\rho\overline{u_i' u_j'} = \mu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) - \frac{2}{3}\rho k \delta_{ij}, \tag{2.72}$$

where k is the turbulent kinetic energy [71]. In Boussinesq's initial proposal, the term $-\frac{2}{3}\rho k \delta_{ij}$ was not included, but it has been added to retrieve the turbulent kinetic energy when taking the sum of normal components $-\rho u_i' u_i'$. For each normal component (i.e., $i = j$), the first term on the right-hand side of Eq. 2.72 becomes zero for incompressible flows. This leaves only $-\frac{2}{3}\rho k$. The sum of normal components becomes

$$-\rho\overline{u_i' u_i'} = -2\rho k, \tag{2.73}$$

which returns the definition of k in Eq. 2.42 [61]. Note the use of Einstein summation convention in Eq. 2.73.

By introducing the Boussinesq assumption, the six independent unknowns of the Reynolds stress tensor have been reduced to only one unknown in the form of μ_t. A common approach to solving turbulent flow problems is to choose a turbulence model that calculates μ_t and then use the eddy-viscosity assumption to obtain the Reynolds stresses. From this point, one can solve the RANS equations.

Eddy-Diffusivity Approximation

It is possible to relate the turbulent heat flux to the averaged quantities in a similar manner as was done for the Reynolds stress in the eddy-viscosity assumption. This is known as the eddy-diffusivity assumption, which expresses the turbulent heat flux as

$$-\overline{u_j' T} = \frac{\nu_t}{Pr_t}\frac{\partial \overline{T}}{\partial x_j}. \tag{2.74}$$

The pre-factor ν_t/Pr_t is known as the eddy thermal diffusivity α_t and it is related to the eddy-viscosity through the turbulent Prandtl number Pr_t,

$$\alpha_t \equiv \frac{\nu_t}{Pr_t}. \tag{2.75}$$

The turbulent Prandtl number is a model constant (or function) that needs explicit prescription. The result is a mathematically closed set of equations for both the mean velocity and mean temperature.

Shortcomings

A shortcoming of the RANS approach is that one solves only for the averaged quantities, while all scales of instantaneous motion are modeled by a turbulence model. This leads to a significant loss of information, which is often necessary to accurately describe turbulent flows. Also, the Boussinesq approximation introduces some inaccuracies as it falsely assumes a linear relationship between the Reynolds stress and the mean flow strain rate $\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)$. This linear relation is not valid in various applications, especially in regions with sudden changes in $\overline{S}_{ij}$, strong streamline curvature, three-dimensional structures, or anisotropic flow conditions [117].

Numerous efforts have been undertaken to overcome the limitations of linear eddy-viscosity models by developing more sophisticated Reynolds stress modeling approaches. However, these approaches have a lack of robustness that restricts them to a small portion of practical turbulent flows, and no turbulence model can accurately describe the flow physics in all circumstances [121].

2.5.2. Large-Eddy Simulation

Large-eddy simulations (LES) provide a compromise between the turbulence modeling approach in RANS and the computationally expensive, all-scale resolving approach of DNS. This technique divides the complete turbulent field into large-scale or ”resolved” eddies and small-scale or ”sub-grid” eddies. The resolved eddies are computed directly through the equations of motion while a single sub-grid scale model represents the combined effect of all sub-grid eddies [38].

Filtering Approach

Resolving motions only up to a cut-off length Δ is the equivalent of introducing a high-pass spatial filter or a low-pass wavenumber filter to the scalar fields. The resolved part $\widetilde{\phi}(\boldsymbol{x}, t)$ of a scalar field $\phi(\boldsymbol{x}, t)$ (e.g., velocity) is defined by the relation

$$\widetilde{\phi}(\mathbf{x}, t) = \int_{-\infty}^{\infty} G(\boldsymbol{\xi}, \boldsymbol{x})\, \phi(\boldsymbol{x} - \boldsymbol{\xi})\, d\boldsymbol{\xi}, \tag{2.76}$$

where the convolution kernel G is a property of the adopted filter, and is associated with cut-off length Δ and cut-off time scale τ_c [35, 86]. The filtering operation can also be described in terms of the wavenumber $\boldsymbol{k}$ and frequency ω by switching to Fourier space. To this end, the Fourier spectrum $\hat{\phi}(\boldsymbol{k}, \omega)$ of $\phi(\boldsymbol{x}, t)$ is multiplied by the transfer function $\hat{G}(\boldsymbol{k}, \omega)$ of the kernel $G(\boldsymbol{x}, t)$:

$$\widetilde{\hat{\phi}}(\boldsymbol{k}, \omega) = \hat{G}(\boldsymbol{k}, \omega)\hat{\phi}(\boldsymbol{k}, \omega), \tag{2.77}$$

where the spatial cut-off length Δ is now associated to the cut-off wavenumber k_c and the cut-off time scale τ_c to the cut-off frequency ω_c. The energy spectrum is also cut off at this scale due to the filtering procedure, meaning that the energy of structures with $k > k_c$ is not included in the resolved flow field. This is schematically shown in Fig. 2.3.

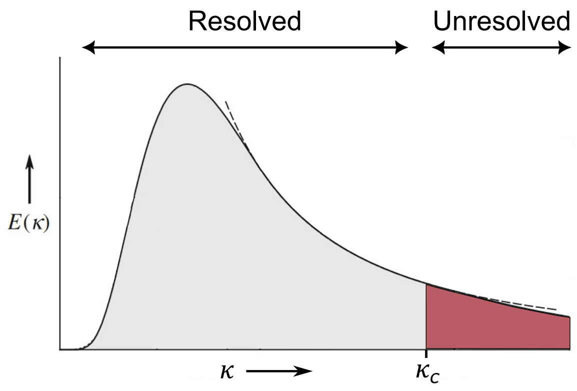


Figure 2.3: The energy spectrum in LES; eddies up to cut-off wavenumber κ_c are resolved. The unresolved eddies are modeled using a sub-grid scale model.

LES Governing Equations

To arrive at the governing equations for the filtered fields, the filters are required to be consistent, linear, and commuting with differentiation [35]. Respectively, this leads to conditions

$$\widetilde{a} = a, \quad a = \text{constant}, \tag{2.78}$$

$$\widetilde{\phi + \psi} = \widetilde{\phi} + \widetilde{\psi}, \tag{2.79}$$

$$\widetilde{\frac{\partial \phi}{\partial s}} = \frac{\partial \widetilde{\phi}}{\partial s}, \quad s = \boldsymbol{x}, t. \tag{2.80}$$

Similar to the Reynolds averaging that was performed in Sec. 2.5.1, one can apply a filter with the above characteristics to the incompressible continuity and Navier-Stokes equations (Eq. 2.1 and 2.5). The resulting filtered continuity equation is given by

$$\frac{\partial \widetilde{u}_i}{\partial x_i} = 0, \tag{2.81}$$

and the filtered Navier-Stokes momentum equation by

$$\rho \frac{\partial \widetilde{u}_i}{\partial t} + \rho \frac{\partial \widetilde{u_i u_j}}{\partial x_j} = \rho f_i + \frac{\partial}{\partial x_j} \left[\widetilde{p} \delta_{ij} + \mu \frac{\partial \widetilde{u}_i}{\partial x_j} \right]. \tag{2.82}$$

The nonlinear term $\widetilde{u_i u_j}$ is the filtered product of two non-filtered variables, which is not a usable result. Similar to the Reynolds decomposition, it is possible to decompose a scalar quantity ϕ into a filtered part $\widetilde{\phi}$ and an unresolved part ϕ''. Following the Leonard decomposition [60], the nonlinear term can be expressed as

$$\rho \widetilde{u_i u_j} = \rho \widetilde{u}_i \widetilde{u}_j + \rho \tau_{ij}^R, \tag{2.83}$$

where $\rho \tau_{ij}^R$ is the residual-stress tensor or sub-grid stress tensor, which is analogous to the Reynolds stress tensor and captures the influence of the sub-filter scales to filtered momentum transfer [93]. From the residual-stress tensor, one can define the residual kinetic energy as

$$k_r \equiv \frac{\tau_{ii}^R}{2}, \tag{2.84}$$

which is the energy contained in the sub-grid scales [86].

Eqs. 2.83 and 2.85 can be combined, leading to the following form of the filtered Navier-Stokes momentum equation [93]:

$$\rho \frac{\partial \widetilde{u}_i}{\partial t} + \rho \frac{\partial \widetilde{u}_i \widetilde{u}_j}{\partial x_j} = \rho f_i + \frac{\partial}{\partial x_j} \left[\widetilde{p} \delta_{ij} + \mu \frac{\partial \widetilde{u}_i}{\partial x_j} - \rho \tau_{ij}^R \right]. \tag{2.85}$$

In a similar manner, one can apply a filter to the total enthalpy balance (Eq. 2.26) to obtain [93]

$$\frac{\partial \widetilde{H}}{\partial t} + \frac{\partial}{\partial x_i} (\widetilde{u}_i C_p \widetilde{T}) = \frac{\partial}{\partial x_i} \left[\alpha \frac{\partial}{\partial x_i} C_p \widetilde{T} - \widetilde{q}_i \right], \tag{2.86}$$

where $-\widetilde{q}_i$ is the sub-grid heat flux that needs to be modeled using a SGS model. Modeling the sub-grid stress tensor τ_{ij}^R and the sub-grid heat flux $\widetilde{q}_i$ with a so-called sub-grid scale (SGS) model represents a central topic in the LES approach.

Eddy-Viscosity and Diffusivity SGS model

The largest and most commonly used class of LES models for τ_{ij}^R are the Eddy-Viscosity Models, which model the residual-stress tensor as [86]

$$-\rho \tau_{ij}^R = \mu_t \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) - \frac{1}{3} \rho \tau_{kk}^R \delta_{ij}, \tag{2.87}$$

where μ_t is the eddy-viscosity. The term $-\frac{1}{3}\rho \tau_{kk}^R \delta_{ij}$ ensures that the residual kinetic energy (Eq. 2.84) is retrieved when taking the sum of normal components $-\rho \tau_{ii}^R$.

For the modeling of the sub-grid heat flux, one can take an analogous approach to the eddy thermal diffusivity approximation that was introduced for RANS in Sec. 2.5.1. Similar to the approximation of the turbulent heat flux, one can approximate the sub-grid heat flux as

$$-\widetilde{q}_i = \alpha_t \frac{\partial \overline{T}}{\partial x_j}, \tag{2.88}$$

where α_t is the eddy thermal diffusivity [120]. It is defined in terms of the eddy-viscosity $\nu_t = \mu/\rho$ and the turbulent Prandtl number Pr_t, according to

$$\alpha_t = \frac{\nu_t}{Pr_t}. \tag{2.89}$$

Sec. 3.5 dives deeper into the different possible approaches for modeling ν_t and discusses some commonly used values for Pr_t.

Implicit Filtering

The spatial filtering approach that has been discussed, removes the small turbulent scales below cut-off length Δ. Usually, the governing equations are not filtered using an explicit filter G as in Eq. 2.76 [8]. Instead, the grid is assumed to behave as an implicit filter that only resolves scales above the cut-off length, which equals the grid spacing. The velocity field defined on the grid is then equivalent to the filtered velocity u_i that appears in Eq. 2.87.

In general, high-quality LES resolves at least 80% of the turbulent kinetic energy in isotropic turbulence [127]. While it is important to choose a grid that has enough resolution, the optimal grid is difficult to determine as the turbulence energy spectrum varies throughout the domain. As was mentioned in Sec. 2.4.4, more turbulent energy is concentrated at higher length scales. Regions with large turbulent scales can therefore have a larger grid spacing than regions where the small scales are dominant. Near no-slip walls, turbulent structures become smaller. For accurate simulations, it is therefore necessary to have high resolution near the walls, which limits the application of LES.

2.6. Parallel Programming on the GPU

We shift our focus to the implementation of high-performance, parallelized computing, now that the relevant physics have been discussed. This is an important topic because the 3D turbulent flow simulations that will be performed in this research require an enormous computational effort. Although the application of LES allows for slightly less resolution in the center of the grid, there is still a need for high resolution near the walls. The Lattice Boltzmann Method (LBM) tracks the collision and propagation of particle distributions locally on each grid cell. This intrinsically makes LBM a promising candidate for parallel implementation on High-Performance Computing (HPC) architectures, including the Graphical Processing Unit (GPU) [103].

In the present work, the GPU NVIDIA implementation will be used, which is compatible with the Compute Unified Device Architecture (CUDA) technology [85]. This technology offers a development environment that enables communication between the CPU and a GPU, such that GPU-accelerated applications can be created and deployed [20]. The CUDA framework was originally developed for C-based languages, but it can also be accessed using the Numba library in Python.

Sec. 2.6.1 dives deeper into the concept of parallel programming and the CUDA framework. In Sec. 2.6.2, the working principles of the CUDA kernel are discussed and the link to the GPU hardware is explained. Lastly, in Sec. 2.6.3, the different memories that are relevant to the GPU will be described.

2.6.1. Parallel Programming

The GPU can break a repetitive task down into much smaller components that can all be finished in parallel due to its large number of cores. This enables GPU-accelerated LBM to reach hundred-fold speed-ups, compared to traditional CPU implementations of LBM. In the CUDA framework, the standard workflow is to define variables and functions on the CPU, which are then transferred to the GPU memory. The GPU then executes the computations, caching data locally for increased performance. When the computations on the GPU are finished, data is then transferred back to the CPU. The transfer of data from device to host and vice versa is a slow process and is therefore required to be minimized. This process can be summarized as:

1. Allocate GPU memory
2. Transfer input variables from CPU memory to GPU memory.
3. Run computations on the GPU
4. Transfer back the results to CPU memory.

Although the parallelization of computations greatly increases computational efficiency, there are some difficulties associated with programming on a CUDA-enabled GPU. The most important ones are:

1. **Low individual performance**: Because there are so many small cores in a GPU, the individual cores are not as powerful as CPU cores. An individual GPU core performs its tasks significantly slower than an individual CPU core. The capabilities of a GPU are therefore only leveraged when a high GPU utilization is achieved, meaning that many GPU cores can perform computations simultaneously.
2. **Limited Functionalities**: The CUDA framework, especially in the Numba environment of Python, offers much fewer functionalities on the GPU than what is common in CPU applications. Almost all built-in functions, modules, and functionalities are not available in a GPU environment. This includes print statements and a large amount of specified error messaging. This makes debugging on a GPU much harder than it is on a CPU and it is therefore extremely important to write robust code that can be tested in small chunks. The limited built-in functions and modules force one to build all their functionalities from scratch. Only some basic basic data frameworks and atomic operations are available.
3. **Raise Conditions**: Many cores read and write data in parallel and the order in which they execute their tasks is randomly determined. It is therefore possible that some cores write to or access the same data in the wrong order, resulting in so-called raise conditions. The program's correctness then relies on the order of execution.

2.6.2. CUDA Kernel and Hardware Perspective

In the CUDA framework, a kernel is a function that can be assigned to the GPU. It is executed a K number of times by K different CUDA threads, which are single execution units that can run in parallel [84]. A thread block is a programming abstraction that signifies a collection of threads and the collection of all thread blocks forms the kernel grid. Threads, blocks and grids are essentially software abstractions that are used to write kernels in the CUDA framework. The programmer must define the amount of threads and blocks before the kernel is executed, choosing the right balance between amounts of threads and blocks for optimal performance. This balance is based on memory constraints and the maximum amount of threads in one block, which is 1024 in current GPUs. All threads in the grid can be synchronized after a kernel has been executed. This is realized with a specfic statement (e.g., numba.cuda.synchronize in the Numba library) that must be reached by all threads before anyone can proceed. It is also possible to synchronize threads during the execution of a kernel, but, in general, only within one specific block.

In a hardware perspective, threads are grouped into so-called warps, which is a set of 32 threads that execute the same instruction. Each warp is executed by one core. Because the GPU executes threads in groups of 32, thread blocks should be initialized in a multiple of 32 threads. If this is not done, the GPU will allocate the remaining threads in the warp anyhow, while they are not assigned to execute any instruction through CUDA. This leads to reduced performance.

The GPU hardware contains several Streaming Multiprocessors or SMs, which are general purpose processors that can execute a maximum amount of thread blocks in parallel [19]. The maximum number of blocks that can be executed simultaneously is determined by the amount of SMs and the resources needed per block [44]. As soon as a thread block has completed execution on an SM, the next thread block in the queue is assigned for execution. Because the SM only schedules a new block when all threads in a block have finished execution, it is important to have an equally distributed computational load across the threads in a block [84]. If one thread takes significantly longer to execute than other threads in the same block, all threads in the block remain on the SM until the last thread is finished. The collection of SMs comprises the whole GPU unit.

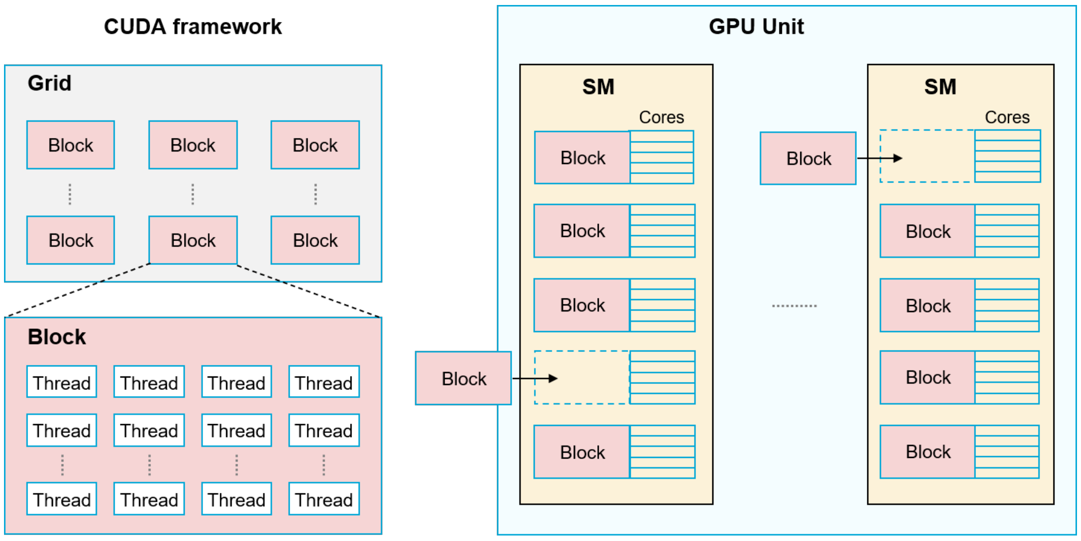


Figure 2.4: Overview of the CUDA framework and GPU hardware. The CUDA framework abstracts the GPU framework in threads, blocks, and a grid. The GPU Unit consists of Streaming Multiprocessors containing a set of cores that can execute warps in a block. New blocks are scheduled, when cores become available in the SM.

2.6.3. Memory Hierarchy

There are several different memory spaces available for the GPU, all with different size, application and performance [84, 18]. They are summarized below:

1. **Global Memory**: The main memory store of the GPU from/to which all threads and the CPU can read and write. The storage capacity is large, but it is slow to read from and write to. The data is stored for the entire lifetime of the simulation.
2. **Constant Memory**: A small memory store that can be read by all threads, but can only written to by the CPU before launching a kernel. It can be very fast if all running threads access the exact same address. It is also stored for the entire lifetime of the simulation. In the Numba library of Python [82], global variables are automatically converted to arrays in the constant memory, which makes the explicit allocation of constant memory unnecessary.
3. **Shared Memory**: Very fast and relatively small memory that is shared among the threads in a block. It makes communication between threads in a block possible and exists only for the lifetime of a block, meaning that it is deleted after kernel execution.
4. **Registers**: The smallest and fastest available memory on the GPU, which is used to store local variables that are declared within a kernel. If the maximum available amount of registers is exceeded, the excess data is stored in the slower local memory. This is known as register spilling. It exists for the lifetime of a thread.
5. **Local Memory**: Slow type of memory that is part of the main memory of the GPU (same location as the global memory). It is used when threads run out of registers during the execution of a kernel. It exists for the lifetime of a thread.

3 Numerical Method

This chapter describes the numerical techniques and implementations that will be applied in the LES-LBM modeling of 3D turbulent heat transfer in conjunction with phase change. Sec. 3.1 discusses the implementation of the highly stable Filter-Matrix Lattice Boltzmann method, both for the velocity and thermal fields. A description of the applied phase change model is given in Sec. 3.3. An overview of the relevant boundary conditions is given in Sec. 3.4. Sec. 3.5 delves into the WALE sub-grid-scale model that will be applied to model the eddy viscosity and thermal diffusivity. Subsequently, two possible approaches to implementing a locally refined grid are discussed in Sec. 3.6. The implementation of existing turbulent data sets to initialize the turbulent simulations is discussed in Sec. 3.7. Lastly, the applied approach in GPU parallelization is treated in Sec. 3.8.

3.1. Filter-Matrix Lattice Boltzmann Method

It was discussed in Sec. 2.2 that the Lattice Boltzmann method can be used to model the dynamics of a flow on discrete grids. The collision operator Ω dictates the evolution of the distribution function f during the collision step and can be chosen in various ways. It was discussed that the simple BGK approach, the lattice Bathnagar-Gross-Krook (LBGK) model, suffers from numerical instability, which happens especially at low viscosity [15]. This is the result of non-physical terms that arise due to discretization errors. An alternative approach, superior over the widely used LBGK, is the multi-relaxation time (MRT) model, proposed by d'Humeriers [77]. This approach has gained increased attention over the past decades [131]. However, it has been reported that the MRT model experiences difficulty in finding the right relaxation rates to achieve the required accuracy and stability [68]. An alternative scheme was proposed in 1993 by Eggels and Somers [98], the filter-matrix lattice Boltzmann method (FMLBM). Although slightly less familiar in the LB community, this approach achieves high stability by filtering out the non-physical terms that arise during the discretization of the Boltzmann equation.

Following [98], the lattice-Boltzmann equation is rewritten to a staggered formulation, in which position and time are shifted by half a grid spacing $\boldsymbol{c}_i\Delta t/2$ and time step $\Delta t/2$, respectively:

$$f_i\left(\boldsymbol{x}+\frac{c_i\Delta t}{2}, t+\frac{\Delta t}{2}\right) = f_i\left(\boldsymbol{x}-\frac{c_i\Delta t}{2}, t-\frac{\Delta t}{2}\right) + \Omega_i(f) \tag{3.1}$$

Subsequently, a Taylor expansion is applied around $f_i(\boldsymbol{x},t)$, leading to

$$f_i\left(\boldsymbol{x}\pm\frac{c_i\Delta t}{2}, t\pm\frac{\Delta t}{2}\right) = f_i(\boldsymbol{x},t) \pm \frac{\Delta t}{2}c_i\cdot\nabla f_i(\boldsymbol{x},t) \pm \frac{\Delta t}{2}\partial_t f_i(\boldsymbol{x},t) + \mathcal{O}\left(\Delta t^2\right), \tag{3.2}$$

which can also be written in terms of the collision operator by filling the above expression for f_i into the staggered LBE (Eq. 3.1),

$$f_i\left(\boldsymbol{x}\pm\frac{c_i\Delta t}{2}, t\pm\frac{\Delta t}{2}\right) = f_i(\boldsymbol{x},t) + \frac{\Delta t}{2}\Omega\left(f_i\right) + \mathcal{O}(\Delta t^2). \tag{3.3}$$

One can express f_i in terms of an equilibrium part f_i^{eq} and a non-equilibrium part f_i^{neq} according to Chapman-Enskog analysis [56]. The equilibrium part f_i^{eq} is given by the discretized equilibrium distribution function in Eq. 2.19. The non-equilibrium part can be expressed in terms of physical quantities by requiring that the Navier-Stokes equations are retrieved from Eq. 3.1 [129]. As a result, the distribution function f_i can be approximated as

$$f_i(\boldsymbol{x},t) = \rho\omega_i \left[1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{1}{2}\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{c_s^4} - \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{c_s^2}\right) - \nu\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{\nabla})(\boldsymbol{c}_i \cdot \boldsymbol{u})}{c_s^4} - \frac{\boldsymbol{\nabla}\cdot\boldsymbol{u}}{c_s^2}\right)\right], \quad i = 0, ..., N \tag{3.4}$$

where N is the amount of velocities in the adopted scheme and ω_i are the corresponding weights [131, 129].

The expression for f_i in Eq. 3.4 can be substituted in Eq. 3.2 and after recalling the definition of Ω_i in the LBE (Eq. 2.18), one can approximate Ω_i as

$$\Omega_i(f) = \frac{\rho w_i}{c_s^2}\left((\boldsymbol{c_i}\cdot\nabla)(\boldsymbol{c_i}\cdot\boldsymbol{u}) - c_s^2\nabla\cdot\boldsymbol{u} + \boldsymbol{c_i}\cdot\boldsymbol{g}\right), \tag{3.5}$$

where g represents the specific body force. By combining Eqs. 3.3, 3.4 and 3.5, we obtain

$$\begin{aligned} f_i\left(\boldsymbol{x} \pm \frac{\boldsymbol{c}_i\delta t}{2}, t \pm \frac{\delta t}{2}\right) =& \rho\omega_i \left[1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{1}{2}\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{c_s^4} - \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{c_s^2}\right)\right. \\ & - \nu\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{\nabla})(\boldsymbol{c}_i \cdot \boldsymbol{u})}{c_s^4} - \frac{\boldsymbol{\nabla}\cdot\boldsymbol{u}}{c_s^2}\right) \\ & \left. + \Delta t\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{\nabla})(\boldsymbol{c}_i \cdot \boldsymbol{u})}{c_s^2} - \boldsymbol{\nabla}\cdot\boldsymbol{u} + \frac{\boldsymbol{c}_i\cdot\boldsymbol{g}}{c_s^2}\right)\right]. \end{aligned} \tag{3.6}$$

This expression can be written as a matrix multiplication by introducing a reversible matrix E_{ki} and a solution vector $\alpha^{\pm}$, such that

$$f_i\left(\boldsymbol{x} \pm \frac{\boldsymbol{c}_i\Delta t}{2}, t \pm \frac{\Delta t}{2}\right) = \sum_k w_i E_{ik}\alpha_k^{\pm}(\boldsymbol{x}), \tag{3.7}$$

where E_{ki} is defined to satisfy the reversibility condition $\Omega_i E_{ik} = E_{ki}^{-1}$. With this condition, it is possible to inversely express the solution vector $\alpha^{\pm}$ as

$$\alpha_k^{\pm}(\boldsymbol{x},t) = \sum_k E_{ki} f_i\left(\boldsymbol{x} \pm \frac{\boldsymbol{c}_i\Delta t}{2}, t \pm \frac{\Delta t}{2}\right). \tag{3.8}$$

The form of E_{ki} and $\alpha^{\pm}$ depends on the chosen velocity scheme.

3.1.1. D3Q19 Filter Matrix Method

It was mentioned in Sec. 2.2 that the most common choices for velocity sets in 3D flows are D3Q15, D3Q19, and D3Q27 schemes. The D3Q19 velocity set is by far the most common scheme in LB simulations as it provides a balance between reasonable computational cost and stability. It has been applied in many 3D turbulent channel flow LES-LBM simulations with satisfying results [129, 78, 114] and it is therefore also the chosen velocity scheme for the current work. The 19-speed filter matrix E_{ki} is defined as

$$\begin{aligned} E_{ki} =& \left[1, c_{ix}, c_{iy}, c_{iz}, 3c_{ix}^2 - 1, 3c_{iy}^2 - 1, 3c_{iz}^2 - 1,\right. \\ & 3c_{iy}c_{iz}, 3c_{ix}c_{iz}, 3c_{ix}c_{iy}, 3c_{ix}\left(c_{iy}^2 - c_{iz}^2\right), 3c_{iy}\left(c_{iz}^2 - c_{ix}^2\right), \\ & 3c_{iz}\left(c_{ix}^2 - c_{iy}^2\right), c_{ix}\left(3c_{iy}^2 + 3c_{iz}^2 - 2\right), c_{iy}\left(3c_{ix}^2 + 3c_{iz}^2 - 2\right), \\ & c_{iz}\left(3c_{ix}^2 + 3c_{iy}^2 - 2\right), 3\left(2c_{ix}^2 - c_{iy}^2 - c_{iz}^2\right)\left(|c_i|^2 - \frac{3}{2}\right), \\ & \left. 3\left(c_{iy}^2 - c_{iz}^2\right)\left(|c_i|^2 - \frac{3}{2}\right), 3|c_i|^2\left(|c_i|^2 - 2\right) + 1\right]^{\mathsf{T}}, \end{aligned} \tag{3.9}$$

and the corresponding solution vector is given by

$$\alpha_k^{\pm} = \begin{bmatrix} \rho \\ \rho u_x \pm \Delta t F_x/2 \\ \rho u_y \pm \Delta t F_y/2 \\ \rho u_z \pm \Delta t F_z/2 \\ 3\rho u_x^2 + \rho\left(-6v \pm \Delta t\right)\partial_x u_x + \rho v \boldsymbol{\nabla}\cdot\boldsymbol{u} \\ 3\rho u_y^2 + \rho\left(-6v \pm \Delta t\right)\partial_y u_y + \rho v \boldsymbol{\nabla}\cdot\boldsymbol{u} \\ 3\rho u_z^2 + \rho\left(-6v \pm \Delta t\right)\partial_z u_z + \rho v \boldsymbol{\nabla}\cdot\boldsymbol{u} \\ 3\rho u_y u_z + \rho\left(-3v \pm 0.5\Delta t\right)\left(\partial_y u_z + \partial_z u_y\right) \\ 3\rho u_x u_z + \rho\left(-3v \pm 0.5\Delta t\right)\left(\partial_x u_z + \partial_z u_x\right) \\ 3\rho u_x u_y + \rho\left(-3v \pm 0.5\Delta t\right)\left(\partial_x u_y + \partial_y u_x\right) \\ -0.8, k = 10, \ldots, 15 \\ -0.95, k = 16, 17, 18 \end{bmatrix}, \tag{3.10}$$

for $k = 0, 1, \ldots, 18$. The terms $a_{10-15}^{\pm}$ and $a_{16-18}^{\pm}$ are third and fourth-order terms that relate to the non-physical contributions. These terms arise from discretization errors that were mentioned at the start of the section. The filter-matrix LB method elegantly dampens these undesired contributions using factors $\gamma_1 = -0.8$ and $\gamma_2 = -0.95$, which increases numerical stability of the simulation. These factors have no specific physical meaning and are also sometimes taken to be zero [131].

3.2. Thermal Lattice Boltzmann Method

In general, there are three ways in which heat transfer can be included into the LB mechanism: the hybrid method, the multi-speed (MS) method and the double distribution function (DDF) method. The hybrid method decouples the computation of flow and temperature, with the latter being solved using traditional CFD methods such as finite-difference techniques. This approach compromises the simplicity of the Lattice Boltzmann Method (LBM) [58]. The multi-speed method, as described in [4], introduces additional particle velocities that represent temperature or energy. However, despite the increased complexity from the expanded velocity set, the multi-speed approach suffers from numerical instability, a limited temperature range, and a fixed Prandtl number [48]. The double distribution method does not experience these drawbacks as thermal distributions g are tracked using an adapted version of the classic lattice Boltzmann method. The thermodynamics can therefore be evaluated independently while leveraging the simplicity of classic LBM.

3.2.1. Double Distribution Function

In the previous section, it has been pointed out that requiring the retrieval of the Navier-Stokes equations from the LBE leads to an expression of f_i in terms of physical flow variables (Eq. 3.6). A similar approach can be applied to the thermal case. The thermal lattice Boltzmann method is associated with a thermal distribution function $g_i(\boldsymbol{x}, t)$, which represents the temperature contribution of velocity population i in a specific point in space and at a specific time step [49]. The corresponding LBE is given as

$$g_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) - g_i(\boldsymbol{x}, t) = \Delta t \Omega_i(g), \tag{3.11}$$

and the equilibrium distribution is given by

$$g_i^{eq} = \omega_i T\left[1 + \frac{\boldsymbol{c}_i\cdot\boldsymbol{u}}{c_s^2} + \frac{\left(\boldsymbol{c}_i\cdot\boldsymbol{u}\right)^2}{2c_s^4} - \frac{u^2}{2c_s^2}\right]. \tag{3.12}$$

The macroscopic temperature is calculated as

$$T(\boldsymbol{x}, t) = \sum_{i=0} g_i(\boldsymbol{x}, t). \tag{3.13}$$

The algorithm works precisely the same as in regular LBM; thermal distributions collide at local lattice sites, after which they stream to neighboring sites. Again, there are several options for the choice of collision operator, such as the BGK, MRT, or filter-matrix approach. Thermal distributions g_i are influenced by the flow distribution functions f_i since the collision operator $\Omega(g)$ contains velocity terms. This accounts for the effect of thermal convection. The flow distribution function f_i can also be influenced by g_i through the force term that appears in $\Omega(f)$. A thermal effect that can be a source of momentum is, for example, buoyancy. Because the present work aims to describe a driven turbulent channel flow, buoyancy effects are expected to have minimal influence over the large inertial forces that are already present in the flow. Buoyancy is therefore chosen to be disregarded.

3.2.2. Enthalpy Distribution

Because the goal of this research is to describe the thermal behavior of fluids undergoing a phase change, it is necessary to determine the temperature of both the fluid and the solid. Because the energy of two phases is distinguished by a difference in latent heat, it becomes more convenient to track total enthalpy rather than temperature alone. The methodology of [49] will be followed to treat the latent-heat source term. This approach avoids iteration steps or solving a group of linear equations for the liquid fraction calculation, which yields higher computational efficiency. We modify the temperature distribution functions in such a way that they represent total enthalpy H, which is defined by

$$H = h + f_l L \tag{3.14}$$

with sensible enthalpy h, temperature T, liquid fraction f_l, and latent heat L. Correspondingly, the equilibrium distribution function g_i^{eq} is modified to

$$g_i^{\text{eq}} = \begin{cases} Lf_l + \omega_i h & i = 0 \\ \omega_i h \left[1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2}\right] & i \neq 0 \end{cases}, \tag{3.15}$$

where higher order terms $\mathcal{O}(u^2)$ have been dropped for simplicity. Following [131], one can define the distribution function g_i in the incompressible limit as

$$g_i = \begin{cases} Lf_l + \omega_i h \left(1 - \frac{u^2}{2c_s^2}\right) & i = 0 \\ \omega_i \left[h + h\frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} - \alpha \frac{\boldsymbol{c}_i \nabla h}{c_s^2}\right] & i \neq 0 \end{cases}. \tag{3.16}$$

It can be seen that the latent term Lf_l has been added to the zero-velocity component g_0. To deal with this term during the filter-matrix collision step, it will be subtracted from g_0 before the start of the collision. After the filter-matrix operation has been applied, the latent term will again be added to g_0. This approach has also successfully been applied in the works of [10, 119]. The total enthalpy H can be retrieved from the distribution function g_i using

$$H(\boldsymbol{x}, t) = \sum_i g_i(\boldsymbol{x}, t) \tag{3.17}$$

Using a similar rationale as in Sec. 3.1, one can introduce the same reversible matrix $\omega_i E_{ik} = E_{ki}$ and a new thermal solution vector $\beta^{\pm}$, such that the enthalpy distribution function g_i can be described as

$$g_i\left(\boldsymbol{x} \pm \frac{\boldsymbol{c}_i \Delta t}{2}, t \pm \frac{\Delta t}{2}\right) = \sum_k w_i E_{ik} \beta_k^{\pm}(\boldsymbol{x}), \tag{3.18}$$

and the solution vector $\beta^{\pm}$ as

$$\beta_k^{\pm}(\boldsymbol{x}, t) = \sum_k E_{ki} g_i \left(\boldsymbol{x} \pm \frac{\boldsymbol{c}_i \Delta t}{2}, t \pm \frac{\Delta t}{2}\right). \tag{3.19}$$

For scalar quantities that are governed by an advection-diffusion equation, such as temperature, it can be sufficient to use a velocity scheme with fewer velocities than in an LBM that describes momentum transfer [56]. Therefore, the D3Q7 scheme will be the primary choice in this work. This scheme has proven to give satisfactory results in earlier turbulence studies with thermal LBM [120, 90, 114]. The D3Q7 filter-matrix LBM has yet only been applied in the turbulent channel DDF DNS-FMLBM by [119], who defined the corresponding filter matrix as

$$E_{ki}^{7} = \left[1, c_{ix}, c_{iy}, c_{iz}, 4c_{ix}^2 - 1, 4c_{iy}^2 - 1, 4c_{iz}^2 - 1\right]^{\top}, \tag{3.20}$$

and the thermal solution vector as

$$\beta_k^{\pm} = \begin{bmatrix} h \\ hu + \frac{-8\alpha \pm \Delta t}{8}\partial_x h \\ hv + \frac{-8\alpha \pm \Delta t}{8}\partial_y h \\ hw + \frac{-8\alpha \pm \Delta t}{8}\partial_z h \\ 0, \quad k = 4, ..., 7 \end{bmatrix}, \tag{3.21}$$

where the non-physical higher-order terms have been set to zero. A D3Q19 thermal scheme would be very similar to the D3Q7 scheme. First of all, the filter matrix E_{ki} is replaced with the D3Q19 version in Eq. 3.9. Second, the solution vector in Eq. 3.22 would be extended with an additional set of higher order terms, leading to

$$\beta_k^{\pm} = \begin{bmatrix} h \\ hu + \frac{-8\alpha \pm \Delta t}{8}\partial_x h \\ hv + \frac{-8\alpha \pm \Delta t}{8}\partial_y h \\ hw + \frac{-8\alpha \pm \Delta t}{8}\partial_z h \\ 0, \quad k = 4, ..., 19 \end{bmatrix}. \tag{3.22}$$

3.3. Solid-Liquid Interface

In the previous section it has been discussed how the FMLBM can be applied to a phase change situation, where the total enthalpy is tracked rather than temperature. The latent term Lf_l occurred in the zero-velocity component g_0 and determines the phase of lattice node. When a solid layer is forming, it is necessary to impose a no-slip condition on the fluid that flows into the solid-liquid interface. This condition will be imposed using the immersed boundary method, which is a convenient technique to integrate phase interface treatment with the collision step. This simplifies the process by not requiring explicit tracking of the moving boundary.

3.3.1. Immersed Boundary Method

The immersed boundary method was originally proposed by Noble and Torczynski [81], and has later been applied in many LBM applications [99, 49, 113]. In the LBM framework, the momentum-LBE is modified to resolve fluid-solid interaction [49] and is given by

$$f_i\left(\boldsymbol{x} + \frac{\boldsymbol{c}_i \Delta t}{2}, t + \frac{\Delta t}{2}\right) = f_i\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i \Delta t}{2}, t - \frac{\Delta t}{2}\right) + (1 - B)\Omega_i + B\Omega_i^s, \tag{3.23}$$

where Ω_s is a modified collision operator and B is a weighing function that depends on the liquid fraction f_l and a parameter ζ,

$$B = \frac{(1 - f_l)\,\zeta}{\zeta}. \tag{3.24}$$

The parameter ζ depends on the relaxation time τ_f, which occurs in the BGK approximation (Sec. 2.2.3), according to $\zeta = \tau_f - 0.5$. The relaxation time τ_f is related to the viscosity ν as [49]

$$\nu = c_s^2(\tau_f - 0.5)\Delta t, \tag{3.25}$$

leading to the definition

$$\zeta = \frac{\nu}{c_s^2 \Delta t}. \tag{3.26}$$

The modified collision operator Ω_s that appears in Eq. 3.23, imposes a zero-velocity bounce-back on nodes that have $f_l = 0$ (i.e., $B = 1$). To this end, it is defined as

$$\Omega_i^s = f_j\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i \Delta t}{2}, t - \frac{\Delta t}{2}\right) - f_i\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i \Delta t}{2}, t - \frac{\Delta t}{2}\right) + f_i^{\text{eq}}(\rho, \boldsymbol{u}_s) - f_j^{\text{eq}}(\rho, \boldsymbol{u}), \tag{3.27}$$

where velocity component j corresponds to the opposite direction of velocity component i.

The global algorithm to describe phase change now becomes:

1. Calculate H with Eq. 3.17 and update f_l according to Eq. 2.27.

2. Perform normal collision and calculate Ω_i, using

$$\Omega_i(f) = f_i^*\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i\Delta t}{2}, t - \frac{\Delta t}{2}\right) - f_i\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i\Delta t}{2}, t - \frac{\Delta t}{2}\right), \tag{3.28}$$

   where f^* is the post-collision distribution.
3. Calculate Ω_i^s using Eq. 3.27.
4. Calculate the corrected post-collision distribution f_i^s according to

$$f_i^s\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i\Delta t}{2}, t + \frac{\Delta t}{2}\right) = f_i\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i\Delta t}{2}, t - \frac{\Delta t}{2}\right) + (1-B)\Omega_i + B\Omega_i^s. \tag{3.29}$$

5. Perform a streaming step with the corrected post-collision distributions f_i^s.

3.4. Boundary Conditions

The situation that is being modeled in this research is the flow between infinite parallel plates. To achieve accurate results, the boundaries of the computational domain must be described by suitable boundary conditions (BCs) for both the momentum and thermal distributions. This section dives into the different types of boundary conditions that will be applied at the channel walls (parallel plates) and the stream- and span-wise directions. Sec. 3.4.1 describes the half-way bounce-back method and the anti-bounce-back method which will be applied to model a no-slip condition and fixed temperature at the wall, respectively. Subsequently, the application of periodic boundary conditions to model infinite parallel plates in the stream- and span-wise directions will be discussed in Sec. 3.4.2. An overview of the computational domain and the different boundary conditions that will be adopted is given in Fig. 3.1.

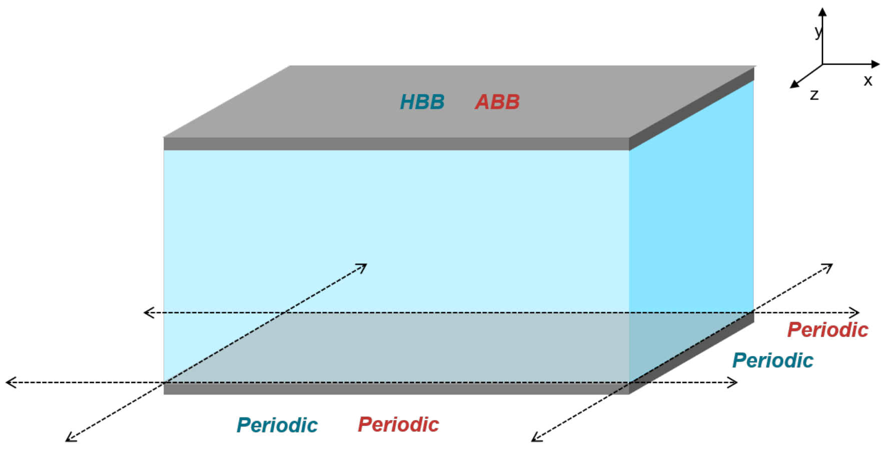


Figure 3.1: Overview of boundary conditions in the domain. Boundary conditions in blue correspond to the momentum distribution functions f, while boundary conditions in red correspond to thermal distribution functions g. Boundary conditions on one side of the channel are the same as boundary conditions on the opposite side of the channel. The x-direction is the stream-wise direction and the z-direction is the span-wise direction.

3.4.1. Solid Walls

At the solid walls, no-slip conditions will be imposed on the momentum distribution functions f and a fixed temperature should be imposed on the thermal distribution functions g.

The bounce-back method is widely adopted in the LBM framework that imposes a no-slip condition by inverting the directions of populations that flow into a wall. The half-way bounce-back method is a sub-category that assumes it takes one time step to invert the directions at the wall. The other option is a full-way bounce back which assumes two time steps for this process. However, the latter method degrades the time accuracy of the LB solution in transient problems [56] and the half-way bounce-back method is therefore the preferred choice in the current work. The corresponding definition of the post-streaming distribution $f_i\left(\boldsymbol{x}_b, t + \frac{\Delta t}{2}\right)$ at a boundary is given by [56]

$$f_i\left(\boldsymbol{x}_b, t+\frac{\Delta t}{2}\right) = f_j\left(\boldsymbol{x}_b, t-\frac{\Delta t}{2}\right), \tag{3.30}$$

where velocity component j corresponds to the opposite direction of velocity component i. Note that the boundary node $\boldsymbol{x}_b$ is located within the flow domain at a distance $0.5[ls]$ from the wall.

The anti-bounce-back scheme is similar to the half-way bounce-back scheme and can be used to impose a wall temperature on the boundary nodes. Because the boundary nodes are located a distance $0.5[ls]$ from the wall, they cannot be fixed at the wall temperature. Rather, they must "feel" the presence of the wall, which can be accomplished using the definition of the anti-bounce-back method

$$g_i\left(\mathbf{x}_b, t+\frac{\Delta t}{2}\right) = -g_i\left(\mathbf{x}_b, t-\frac{\Delta t}{2}\right) + 2w_i h_w, \tag{3.31}$$

where h_w is the sensible enthalpy at the wall and w_i are the weights of the velocity scheme [56]. The sensible enthalpy is used rather than the total enthalpy because the latent term is enclosed only in the zero-velocity population. This implies that there is no streaming of the latent term. The sensible enthalpy is defined in terms of the wall temperature according to Eq. 2.29.

3.4.2. Periodicity

The stream- and span-wise directions are assumed to be infinitely long by the use of periodic boundary conditions for f and g. These conditions consider the flow to be periodic in the applied directions, such that the fluid leaving the domain on one side re-enters the domain on the other side. Flow profiles of turbulent flows are not periodic in reality and this assumption may lead to non-physical behavior. However, if the domain is taken large enough for the largest turbulent structures to be well captured, realistic turbulent statistics can be obtained. In the LB framework, periodic boundary conditions in a channel of length L along the periodic axis can be applied using

$$f_i\left(\frac{\Delta x}{2}, t-\frac{\Delta t}{t}\right) = f_i\left(L-\frac{3\Delta x}{2}, t+\frac{\Delta t}{2}\right), \tag{3.32}$$

$$f_i\left(L-\frac{\Delta x}{2}, t-\frac{\Delta t}{t}\right) = f_i\left(\frac{3\Delta x}{2}, t+\frac{\Delta t}{2}\right). \tag{3.33}$$

where $\Delta x = \boldsymbol{c}_i \Delta t$ is the grid spacing.

3.5. Sub-Grid Scale Model

The turbulent behavior will be simulated using a large eddy simulation (LES). As was explained in Ch. 2.5.2, this implies that the turbulent structures are only resolved up to the filter width Δ and the unresolved eddies are modeled using a sub-grid-scale (SGS) model. It has also been explained that Eddy-Viscosity Models are the largest and most widely used class of LES models. They introduce a modeled SGS eddy-viscosity $\nu_t = \mu_t/\rho$ for which many different approaches have been proposed over the past decades.

3.5.1. Smagorinsky Model

The oldest and most widely used approach in determining ν_t is the model originally proposed by Smagorinksy [96]. It is a popular choice for LES due to its simplicity and ease of use. In the Smagorinsky model, the turbulent viscosity is given by [72]

$$\nu_t = C_S^2 \Delta^2 |\widetilde{S}_{ij}|, \tag{3.34}$$

where C_S is the flow-specific Smagorinsky constant, Δ is the cut-off length that distinguishes between large scales and unresolved small scales, and $|\widetilde{S}|$ is expressed in terms of the resolved strain-rate tensor $\widetilde{S_{ij}}$ through

$$|\widetilde{S}| = \sqrt{2\widetilde{S}_{ij}\widetilde{S}_{ij}}, \tag{3.35}$$

with

$$\widetilde{S}_{ij} = \frac{1}{2}\left(\frac{\partial \widetilde{u}_j}{\partial x_i} + \frac{\partial \widetilde{u}_i}{\partial x_j}\right). \tag{3.36}$$

Although the Smagorinsky model is popular due to its simple implementation, a major downside is its overly dissipative nature near walls [112]. Close to the walls, turbulent structures become smaller and the flow becomes effectively laminar. However, the resolved strain-rate $\widetilde{S}_{ij}$ does not go to zero here, which leads to an overestimation of ν_t and, thus, of energy dissipation. Another problem in the Smagorinksy model is the flow type dependence of the optimal value for C_S, which has values generally ranging from 0.05 to 0.16 [116]. A third shortcoming is that the (dynamic) Smagorinsky model bases its calculation of the eddy viscosity only on the resolved strain-rates. However, turbulent kinetic energy is concentrated around zones of both strain and vorticity [79]. The latter is not accounted for in Smagorinsky-like models.

3.5.2. Improvements to the Smagorinsky Model

An attempt to overcome the over-estimation of ν_t near walls has been introduced in the form of damping functions. These are functions of the wall distance that are multiplied with the Smagorinsky constant and go to zero at the wall. A popular example is the Van Driest damping function. However, the use of wall functions is not particularly robust in non-uniform geometries as they have a global dependence on the dimensionless wall distance [116]. In the present research this is especially unpractical, since there is a dynamic solid-liquid interface.

Germano et al. [37] and Lilly [64] proposed a way to calculate the Smagorinsky constant dynamically, aiming to solve the flow-dependency of the model constant. In such a model, the model coefficient is computed dynamically and changes both in space and time. To this end, two different filter levels are introduced to evaluate the sub-grid stresses [37]. However, the model is relatively complicated and the permanently changing LES model may lead to instabilities, as was mentioned in [116]. One source of instabilities is the possibility of obtaining negative values for the eddy viscosity in this procedure. The common implementation of the dynamic procedure involves ad-hoc clipping of the eddy viscosity, which is an additional procedure that ensures a minimum eddy viscosity of zero. The dynamic Smagorinsky model is not applied in the current work due to its relatively cumbersome implementation and the availability of techniques that are more straightforward with superior stability, while remaining accurate.

A choice of sub-grid scale model that does not involve additional filtering or clipping procedures and where the calculated eddy viscosity naturally goes to zero near walls, is the Vreman model [111]. This model is expressed in terms of the Smagorinsky constant and a combination of first-order derivatives, making it relatively easy to implement. It has successfully been applied in several recent applications such as the LES-FMLBM by Zhuo and Zhong [129] in 2016 and the DDF-LB by Ren et al. [90] who combined it with a dynamic procedure to determine the model constant.

Nicoud and Ducros [79] proposed a comparable SGS model, the wall-adapting local eddy viscosity (WALE) model. This model was developed for LES in complex geometries and does not involve explicit filtering or stabilizing procedures, similar to the Vreman model. In addition, the WALE model accounts for both the strain-rate and vorticity as sources of turbulence. The eddy viscosity also naturally goes to zero near the walls, without the need for any ad-hoc damping functions. Liu et al. [67] and Zhang et al. [125] both investigated the performance of WALE in comparison with the Vreman model in $Re_\tau = 180$ turbulent channel flow and both found a similar performance of the two models in terms of the flow statistics with maximum deviations of 1%[67]. However, the eddy viscosity of the WALE model reproduces proper scaling near the wall $\nu_t = \mathcal{O}(y^3)$, while Vreman produces $\nu_t = \mathcal{O}(y)$ [125]. Due to its simple implementation, high accuracy and accurate vanishing of eddy-viscosity near walls, the WALE model has been chosen as the SGS model in the current work.

3.5.3. WALE Model

In the WALE model, a tensor S^d_{ij} is introduced, which is defined in terms of resolved velocity gradient tensors $\widetilde{g}_{ij} = \partial \widetilde{u}_i / \partial x_j$:

$$S^d_{ij} = \frac{1}{2}\left(\widetilde{g}^2_{ij} + \widetilde{g}^2_{ji}\right) - \frac{1}{3}\delta_{ij}\widetilde{g}^2_{kk}, \tag{3.37}$$

where $\widetilde{g}^2_{ij} = \widetilde{g}_{ik}\widetilde{g}_{kj}$ [79]. Nicoud and Ducros showed that Eq. 3.37 can also be rewritten in terms of the resolved strain-rate tensor $\widetilde{S}_{ij} = \frac{1}{2}\left(\widetilde{g}_{ij} + \widetilde{g}_{ij}\right)$ and the resolved rotation-rate tensor $\widetilde{\Omega}_{ij} = \frac{1}{2}\left(\widetilde{g}_{ij} - \widetilde{g}_{ji}\right)$. This argument clarifies that the WALE model treats both strain and vorticity as sources of eddy-viscosity.

The WALE model expresses the eddy viscosity in terms of $\widetilde{S}_{ij}$ and S_{ij}^d:

$$\nu_t = (C_w \Delta)^2 \frac{\left(S_{ij}^d S_{ij}^d\right)^{3/2}}{\left(\widetilde{S}_{ij}\widetilde{S}_{ij}\right)^{5/2} + \left(S_{ij}^d S_{ij}^d\right)^{5/4}}. \tag{3.38}$$

The WALE model constant C_w is proportional to the Smagorinsky constant C_S and takes reported values ranging from 0.325 to 0.6, depending on grid spacing and the flow profile.[21, 79] The WALE-DUGKS simulation of wall-bounded flow by [125] and the WALE-LBM of turbulent channel flow by [116] both used a constant $C_w = 0.50$ with satisfactory results. This value was also originally proposed by Nicoud and Ducros who observed the best results for isotropic turbulence modeling at $C_w = 0.5$. For these reasons, it is decided also to use $C_w = 0.5$ in the current work.

The differentials will be obtained through the application of a central difference scheme on velocities $\widetilde{u}$, which is recommended for LES [112]. At the boundaries of the domain, a three-point scheme will be used due to the absence of adjacent nodes at the wall; this will be a forward scheme for lower-y boundaries and a backward scheme for upper-y boundaries. Discretization errors that are introduced by the latter schemes should have no significance near the walls since the eddy viscosity vanishes here.

3.5.4. SGS Eddy-Diffusivity

As was discussed in Sec. 2.5.2, the sub-grid-scale eddy-diffusivity α_τ can be modeled by introducing a turbulent Prandtl number that relates the turbulent kinematic viscosity α_τ to the turbulent viscosity ν_τ as

$$\alpha_\tau = \frac{\nu_\tau}{Pr_\tau}. \tag{3.39}$$

For molecular Prandtl numbers close to unity, the practical values of Pr_τ range from 0.3 to 0.9.[107] Zhuo et al. [130] used a value of 0.4 for the LBM-LES of natural circulation in a square cavity. A value of 0.5 was used in the LBM-LES of turbulent heat transfer by [66], but this value was mentioned to be too low at the peaks of the root-mean-square temperature fluctuation, resulting in a slight overprediction. In reality, Pr_τ is not spatially constant and there have been studies where it has been computed following a dynamical procedure, such as in the turbulent channel flow LBM-LES of [106]. They used a constant value $Pr_\tau = 0.9$ as a reference and observed that the dynamical procedure affected the predicted heat transfer only marginally, which led to the conclusion that a constant turbulent Prandtl number is justified in further research and that $Pr = 0.9$ is an acceptable choice. The value $Pr_\tau = 0.9$ has also successfully been used in the wall-modeled LBM-LES of turbulent heat transfer by [57] and the LES-LBM of a thermal impinging jet by [78]. The above arguments justify the choice $Pr_\tau = 0.9$ in the current work.

For larger molecular Prandtl numbers, one can resort to experimentally determined relations that relate Pr_t to the molecular Prandtl number [47]. Sometimes the Reynolds number is also incorporated in such relations.

3.5.5. LBM Implementation

As is the case in almost any LBM-LES application, the grid will act as an implicit filter, such that the filter width Δ is equal to the grid spacing Δx, which is taken to be unity in LB units. Furthermore, the use of an eddy-viscosity can be translated to the LBM framework by introducing an effective viscosity ν_e that is to be used in the collision step. This effective viscosity can simply be determined as the sum of the molecular viscosity ν and the eddy-viscosity ν_τ: [129]

$$\nu_e = \nu_0 + \nu_t. \tag{3.40}$$

3.6. Local Grid Refinement

A suitable grid is extremely important for achieving accurate results with good computational efficiency. In the current work, we are interested in 3D channel flows with no-slip boundary conditions and fixed temperatures at the channel walls. As the turbulent length scales become smaller and more an-isotropic near the wall, it is necessary to have a well-resolved grid for enough accuracy in this region. Towards the center of the channel, however, this is not a requirement, so a coarser grid is preferred to increase computational efficiency. To obtain a grid with the required characteristics, three different layers of refinement will be constructed. As schematically shown in Fig. 3.2, there is a coarse layer in the middle of the channel, which is enclosed by two fine layers near the walls. The coarse layer has grid spacing Δy_c and the fine layers have grid spacing Δy_f. All cells in the grid are cubic, such that $\Delta x_{f/c} = \Delta y_{f/c} = \Delta z_{f/c}$.

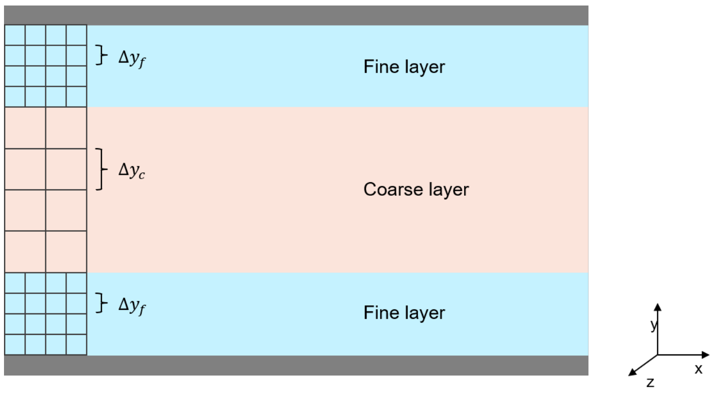


Figure 3.2: Schematic overview of local grid refinement in the computational domain. Fine layers with grid spacing $\Delta x_f = \Delta y_f = \Delta z_f$ are located near the walls; a coarse layer with $\Delta x_c = \Delta y_c = \Delta z_c$ is located in the center of the channel.

3.6.1. Hierarchical Grid Refinement Techniques

A widely recognized category of local grid refinement techniques for the Lattice Boltzmann Method adopts length and time scales in ratios of two between neighboring fine and coarse cells. These are so-called hierarchical grid refinement techniques. Three sub-categories can be distinguished here: cell-vertex methods, cell-centered methods, and combined methods [94]. The different grid structures adopted in these methods are schematically shown in Fig. 3.3.

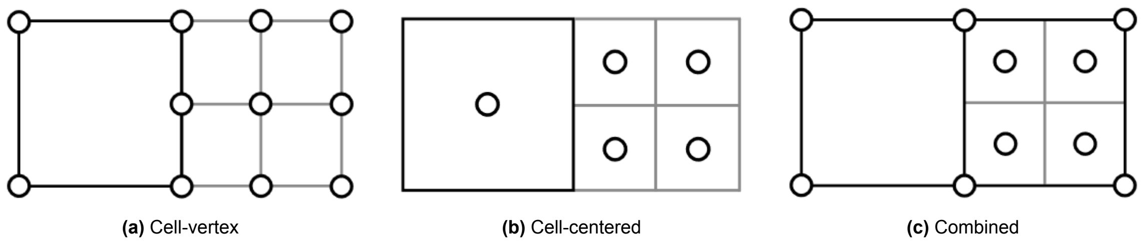


Figure 3.3: Interface layouts of fine-coarse grid transitions [94].

The cell-vertex approach has been implemented in many different LBM works such as [31, 65, 124, 26]. It uses a grid layout with all nodes at the cell corners, leading to overlapping fine and coarse cells on the layer interface. This method requires scaling of the non-equilibrium part of the distribution function to avoid introducing an error to the local components of the stress tensor [92]. In the cell-centered approach, all nodes reside at the center of their corresponding cells. Rohde et al. [92] developed a cell-centered method that guarantees mass and momentum conservation without the need to re-scale the non-equilibrium distribution. This becomes possible because nodes are arranged in a volumetric manner where they can be considered as masses rather than densities. The last method is a combination of both of the above methods. Fine nodes are located in the cell centers, while coarse nodes are located in the cell corners. Distribution functions at interface nodes are obtained through decomposition into equilibrium and non-equilibrium parts with second-order interpolation schemes [94]. The performance of the three types of grid layouts has been tested by [94] and it was found that the cell-centered approach showed superior stability compared to the other two. Furthermore, the cell-centered algorithm of Rohde et al. is relatively straightforward, as it uses no scaling of the non-equilibrium distribution and no interpolation. Therefore, this algorithm will be considered for the remainder of this work.

3.6.2. The Rohde algorithm

The working of the cell-centered local grid refinement algorithm that was developed by Rohde et al. [92] will now be discussed. The idea is to define an interface layer between the coarse and fine grids, on which we define both fine and coarse cells. Without communication between layers, errors would be introduced at the end of a layer because there is no propagation source adjacent to the end of a layer (except at domain boundaries, where boundary conditions are present). The algorithm solves this issue by combining contributions of both layers at the interface, to end up with physical distributions after each iteration.

The different steps of the algorithm will now be explained. They are also schematically shown in Fig. 3.4. In the algorithm steps below, the coarse layer is denoted as C, the fine layer as F, fine cells on the interface layer as I_F, and coarse cells on the interface layer as I_C. The interface layer is defined to be a part of the coarse layer, such that

$$I_C \subset C, \tag{3.41}$$
$$I_F \not\subset F. \tag{3.42}$$

This means that coarse cells on the interface layer are considered to be part of the domain, while the co-located fine cells are purely virtual. The algorithm is generalized for any refinement level r, which is the ratio between coarse and fine cell lengths. The steps of the algorithm are as follows:

1. **Collision** step on $\{F, C\}$.

2. **Homogeneous redistribution** of particle densities from I_C to I_F. Particle distributions are copied from coarse cells to the r^3 co-located fine cells, according to

   $$(f_i(\mathbf{x}_p, t))_f = (f_i(\mathbf{x}, t))_c, \quad p = 1, ..., r^3 \tag{3.43}$$

   Only distributions that point into the fine layer are redistributed, because these are the only directions that propagate into the physical domain.

3. **Propagation** step on $\{F, C\}$.

4. **Non-synchronous iterations**. Repeat steps 4a and 4b $(r-1)$ times.

   a **Collision** step on $\{F\}$.

   b **Propagation** step on $\{F, I_F\}$.

5. **Homogeneous redistribution** of particle densities from I_F to I_C. This is done by averaging particle distributions of r^3 fine cells and assigning this to the co-located coarse cell, according to

   $$(f_i(\mathbf{x}, t))_c = \sum_{p=1}^{r^3} (f_i(\mathbf{x}_p, t))_f. \tag{3.44}$$

   Only distributions that point into the coarse layer are redistributed, because these are the only directions that are non-physical due to the previous coarse propagation step (step 3).

Although this algorithm conserves both mass and momentum, a slight error is being introduced during the non-synchronous iteration steps (step 4). After the collision step in Step 1, coarse cells are ahead of fine cells by $r-1$ fine time steps. In step 2, fine interface cells receive these advanced distributions, which are then partly propagated onto the fine grid during step 3. In step 4, some of the advanced distributions are falsely colliding as they now partly reside on the fine grid. Rohde et al. observed that their technique introduced an error to the simulation that becomes larger for flows perpendicular to the refinement interface. For a refinement factor $r > 2$, multiple layers of advanced distributions propagate onto the fine grid, which leads to larger errors. However, such cases were not investigated by Rohde et al.

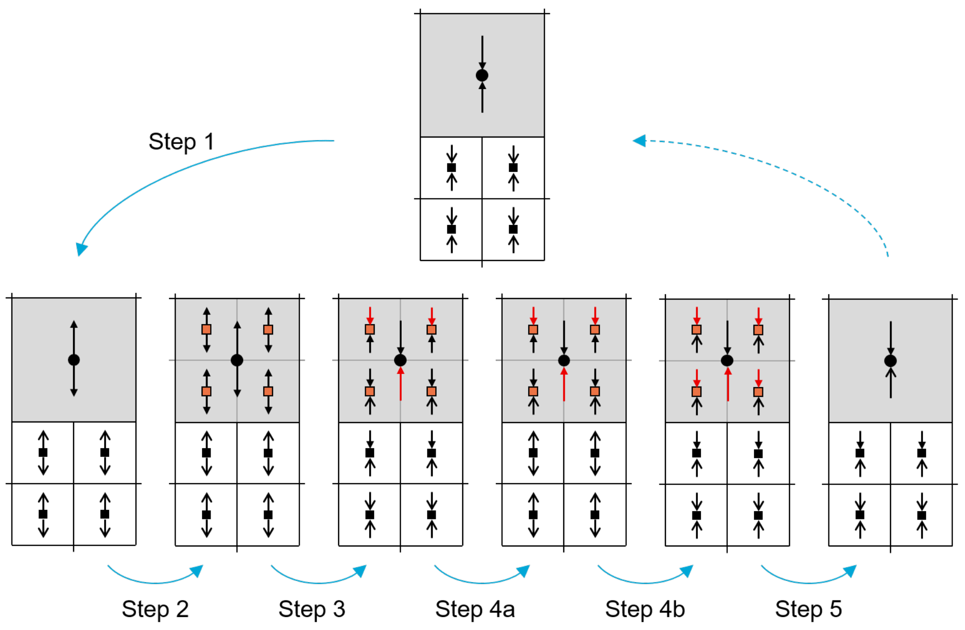


Figure 3.4: Schematic overview of the local grid refinement algorithm that was developed by Rohde et al. [92]. On the interface layer (grey) both fine and coarse cells are defined, denoted by squares (■) and circles (●), respectively. Particle distributions after a propagation step are represented by arrows pointing toward the cell center, while particle distributions after a collision step are represented by arrows pointing from the cell center. Virtual nodes are shown in orange, and domain nodes are shown in black. Non-physical distributions that result from an inability to propagate at the end of a layer, are shown in red. Physical distributions are shown in black, disregarding errors due to collisions of over-advanced distributions. The fine and coarse layers respectively stretch downward and upward.

3.6.3. A novel algorithm

Motivated by the nonphysical behavior that results from overly collided distributions at the end of the fine layer, a novel local grid refinement technique is presented that does not suffer from this issue.

To this end, we define an interface layer that stretches two coarse cell lengths instead of only one. We denote the interface layer half closest to the fine layer as IF and the half closest to the coarse layer as IC. Again, subscripts F and C denote the fine and coarse cells on the interface layer, respectively. Now IF is defined to be part of the fine layer and IC is defined to be part of the coarse layer, such that

$$IF_F \subset F, \tag{3.45}$$
$$IC_F \not\subset F, \tag{3.46}$$
$$IC_C \subset C, \tag{3.47}$$
$$IF_C \not\subset C, \tag{3.48}$$
$$I = \{IF, IC\}. \tag{3.49}$$

This means that fine cells on one half of the interface layer and coarse cells on the other half are considered to be part of the domain, while the remaining fine and coarse cells are purely virtual. The conceptual grid layout is shown in Fig. 3.5 To easily distinguish between the different cells in the interface layer, each layer of fine cells in I is labeled with a label n from 1 to $2r$. We count in the direction from I_F to I_C. The algorithm is schematically shown in Fig. 3.6. The different steps are as follows:

1. **Collision** step on all cells $\{F, C, I\}$.
2. **Propagation** step on $\{F, C, I\}$.
3. **Non-synchronous iterations**. Repeat steps 3a and 3b $(r - 1)$ times.

   a **Collision** step on all fine cells $\{F, IC_F\}$.

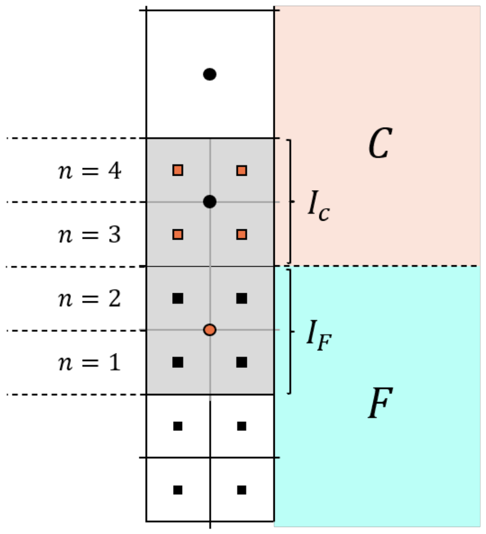


Figure 3.5: Schematic layout of the proposed layers for $r = 2$. C is the coarse layer, F is the fine layer, I_C is part of the interface layer that overlaps with the coarse layer, and I_F is part of the interface layer that overlaps with the fine layer. Fine cells in I ($= \{I_F, I_C\}$) are labeled with a label n from 1 to $2r$.

b **Propagation** step on all fine cells $\{F, I_F\}$.

4. **Homogeneous redistribution** of particle densities from IF_F to IF_C. This is done by averaging particle distributions of r^3 fine cells and assigning this to the co-located coarse cell, according to Eq. 3.44. Only distributions that point into the fine layer are redistributed because these are the only non-physical ones.

5. **Homogeneous redistribution** of particle densities from IC_C to IC_F. Particle distributions are copied from coarse cells to the r^3 co-located fine cells, according to Eq. 3.43. The directions that are redistributed depend on the location of the fine cells. Assuming a similar labeling as in Fig. 3.5, the following redistribution rules for different fine cell labels n:

   - $n = 2r$: redistribute directions pointing into the fine layer.
   - $n = 2r - 1$: redistribute directions pointing into the fine layer and directions parallel to the interface.
   - $r < n < 2r - 2$: redistribute all directions. This rule occurs only when $r > 2$

   These rules follow from the collision and propagation of non-physical directions in the interface layer.

The advantage of this technique is that there are no collisions of over-advanced distributions which was the case in the algorithm of Rohde et al. Its performance will be assessed in Ch. 4.

3.6.4. Coarse and Fine Lattice Units

As the fine and coarse layer have different discretization in space and time, it is necessary to adjust the simulation variables accordingly. For each coarse time step, r fine time steps are executed. The coarse grid spacing Δx_c and time step Δt_c are therefore related to the fine grid spacing Δx_f and Δt_f as

$$\Delta x_c = r\Delta x_f, \tag{3.50}$$

$$\Delta t_c = r\Delta t_f. \tag{3.51}$$

We can now easily derive similar relations for both the viscosity and body force. To this end, we recall the relationship between the non-dimensionalized and physical viscosity (Eq. 3.52) and body force (Eq. 3.53):

$$\nu_{LB} = \frac{\Delta t}{(\Delta x)^2} \cdot \nu_{phys} \tag{3.52}$$

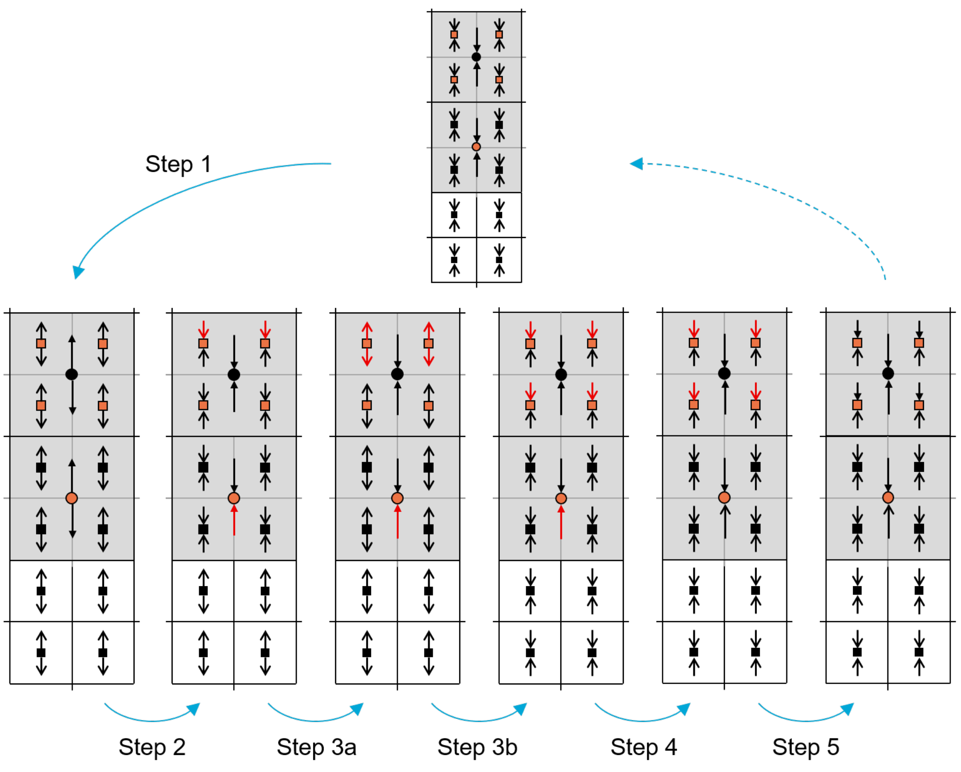


Figure 3.6: Schematic overview of the grid refinement algorithm proposed in the current work. On the interface layer (grey) both fine and coarse cells are defined, denoted by squares (■) and circles (●), respectively. Particle distributions after a propagation step are represented by arrows pointing toward the cell center, while particle distributions after a collision step are represented by arrows pointing from the cell center. Virtual nodes are shown in orange, and domain nodes are shown in black. Non-physical distributions that result from an inability to propagate at the end of a layer, are shown in red, while physical distributions are shown in black. The fine and coarse layers respectively stretch downward and upward.

$$g_{LB} = \frac{(\Delta t)^2}{\Delta x} \cdot g_{phys} \tag{3.53}$$

The above relations can be combined with Eqs. 3.50 and 3.51 to arrive at

$$\nu_{LB,c} = \frac{1}{r} \cdot \nu_{LB,f} \tag{3.54}$$

$$g_{LB,c} = r \cdot g_{LB,f}. \tag{3.55}$$

Eqs. 3.54 and 3.55 are ultimately used to scale the viscosity and body force in the coarse layer during the collision step.

3.6.5. Convergence Measures

To accurately describe the performance of different numerical models that will be applied in this study, it is important to define metrics that can indicate whether a time-dependent field converges towards a steady state or aligns with an analytical solution.

A quantity that can be used to check convergence of a simulated field to a steady state is the L2 difference $\epsilon_{\text{diff, L2}}$. It quantifies the deviation between velocity fields at consecutive time steps, and has been applied in various applications [13, 105]. It can be defined as

$$\epsilon_{\text{diff, L2}} = \sqrt{\frac{\sum_{x} \left(u(\boldsymbol{x},t) - u(\boldsymbol{x},t-\Delta t)\right)^2}{\sum_{x} \left(u(\boldsymbol{x},t-\Delta t)\right)^2}}, \tag{3.56}$$

where $u(\boldsymbol{x},t)$ is the numerical field at the current time step and $u(\boldsymbol{x},t-\Delta t)$ is the numerical field at the previous time step. When a solution converges to a steady state, $\epsilon_{\text{diff, L2}}$ becomes smaller with every time step until it approaches a value that is limited by the numerical error. According to [56], time convergence with double precision arithmetic is typically claimed for values around $\epsilon_{\text{diff, L2}} = 10^{-7}$.

A similar quantity that can be used to assess convergence towards an analytical solution is the L2 error norm $\epsilon_{\text{err, L2}}$. It is also known as the root-mean-square error and can be defined as

$$\epsilon_{\text{err, L2}} = \sqrt{\frac{\sum\limits_{x}\left(u(\boldsymbol{x},t)-u_a(\boldsymbol{x},t)\right)^2}{\sum\limits_{x}\left(u_a(\boldsymbol{x},t)\right)^2}}, \tag{3.57}$$

where $u(\boldsymbol{x},t)$ is a numerical field and $u_a(\boldsymbol{x},t)$ is the analytical field that is being approximated. A smaller value of $\epsilon_{err,L2}$ implies a more accurate model.

3.7. Initialisation

To study the behavior of turbulent channel flow, it is necessary to construct an initial flow profile that not only satisfies the relevant flow parameters but also exhibits the desired chaotic turbulent characteristics. To this end, existing flow data of developed turbulence with known Reynolds numbers are used as an initial condition for the simulations.

3.7.1. Initial Data

Initial data sets will be extracted from DNS profiles of Van Bemmelen [105], who constructed turbulent velocity profiles with characteristics as listed in Tab. 3.1.

Table 3.1: Numerical settings for turbulent channel flow as constructed by [105].

Re_τ	Re_τ^{out}	Re_m	N_x x N_y x N_z	u^+	Δy^+	ν	g
180	180.2	5590	256 x 128 x 128	6.667e-3	2.8	2.37e-3	6.94e-7
395	398.3	14040	460 x 230 x 230	5.714e-3	3.4	1.66e-3	2.84e-7

These profiles were constructed through direct numerical simulation of disturbed laminar flow fields that developed until sustained turbulence was achieved. All flow cases showed good qualitative agreement to benchmark results overall [105], except for slight deviations in RMS velocity and vorticity fluctuations near the walls. These deviations were attributed to a lack of grid refinement compared to the used benchmark studies.

3.7.2. Linear interpolation

To incorporate this data into the current project, it must be projected onto the new, locally refined grid while preserving its original turbulent properties. This is done through linear interpolation of the original velocity and density arrays such that the new set of arrays matches the new grid. The interpolated flow data is then converted to distribution arrays by calculating solution vectors α^- using Eq. 3.10 and subsequently calculating distribution functions f_i using Eq. 3.18. This procedure increases the cut-off length of the sub-grid scale eddies in regions where the grid becomes coarser. The resolved eddies will still have their original shape, leading to similar flow characteristics of scales above the cut-off length. It is therefore expected that linear interpolation will be successful in constructing the initial turbulent field.

The calculation of α^- is performed with the original body force and viscosity in Tab. 3.1, for both fine and coarse layers. This is necessary because the calculation of α^- implies a time step $-\Delta t/2$ that must be the same for the whole grid to be synchronous in time. Also, there is no propagation associated with the initialization procedure so differences in grid spacing are irrelevant here.

3.8. Code Implementation and Testing

The simulation will be performed using a Graphical Processing Unit (GPU) to parallelize the execution of calculations across the domain. The CUDA module from the Numba package in Python will facilitate communication with the GPU. Furthermore, a base code will be provided with basic CPU-based propagation, collision, and boundary condition algorithms. This base code is capable of simulating a simple laminar flow without grid refinement or heat transfer. This section will discuss concepts related to the adopted code implementation and the testing approach.

3.8.1. Grid Setup

Following the structure of the provided LBM base code, the current implementation will use a row of ghost nodes on the edges of the domain, which are used to store inverted distributions in the half-way bounce-back method. This yields an additional 2 cells in the wall-normal direction. In the streaming step, distributions located on these ghost nodes flow into the fluid domain. Second, the stream- and span-wise directions are extended such that one periodic unit of the flow domain is of the size mentioned in 3.1. In the stream- and span-wise directions, this yields an additional $2r + 2$ cells for the fine layer and an additional 4 cells for the coarse layer. The need for these cells becomes clear in Fig. 3.7, which shows a periodic flow unit in the stream-wise direction (stretching from first cell B to second cell A). The 2 ghost nodes in the stream- and span-wise directions are not necessary, but they are included for correspondence between different versions of the code.

The GPU kernel grid will have a 3D setup with the number of threads and blocks in each direction equal to the number of cells in the grid, rounded up to the nearest multiple of 32. This allows us to distinguish between thread blocks that fall within the fine layer and thread blocks that fall within the coarse layer. Each thread corresponds to an x-coordinate, while each block corresponds to a y- and z-coordinate.

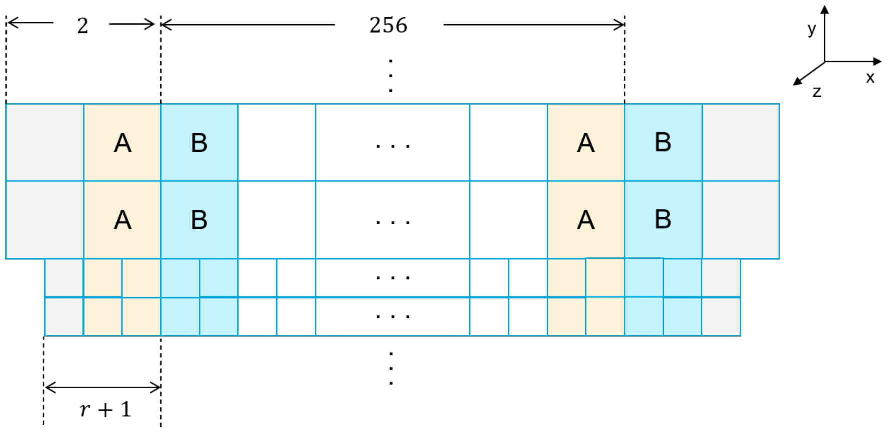


Figure 3.7: Schematic overview of the grid layout in the stream-wise direction. Cells with the same color and letter correspond to the same cells in different periodic units. The periodic fluid domain is 256 fine cells long, while the full grid has an additional $2r + 2$ fine cells and 2 coarse cells. The grid layout in the span-wise direction has the same amount of additional cells at the ends of the periodic unit.

3.8.2. GPU Implementation

Performance optimization of CUDA-based GPU implementations relies on four main strategies:

1. Maximize parallel execution to achieve maximum utilization
2. Optimize memory usage to achieve maximum memory throughput
3. Optimize instruction usage, which refers to
4. Minimize memory thrashing, which refers to allocating and deleting memory unnecessarily often.

One can resort to [84] for precise descriptions and recommendations based on these guidelines. These will be pursued in the written code.

Concerning optimizing memory usage, all input variables that are relevant to the simulation will be pre-transferred to the GPU. It is possible to insert CPU constants into GPU kernels, but this will cause them to be transferred to the GPU repetitively at each time step. Additionally, it has been recommended by [105, 119] to apply shared memory into the algorithms, so this will also be attempted.

Furthermore, concerning achieving maximum utilization, there will be a set of threads in the coarse layer that is unused due to the local grid refinement algorithm. This is because the coarse layer only has $N_{x,c} = (N_x - 2N_{x,f})/r$ cells in the stream-wise direction, leading to $N_x - N_{x,c}$ unused threads. To correct for this under-utilization, the propagation and collision algorithms will be adapted such that the unused threads are assigned to a new row of cells in the stream-wise direction. Some blocks are also unused, but this is less of an issue because no thread is executing anything, so they are quickly processed by the streaming multiprocessor. Unused threads in blocks that are partly utilized, however, occupy space on the streaming multiprocessor until the other threads in the same block have finished computing. This is undesired as was explained in Sec. 2.6.

3.8.3. Automated Testing

It has been explained in Sec. 2.6 that parallel programming on the GPU introduces additional sources of error and, at the same time, makes debugging more difficult due to limited functionalities and error messaging. In addition, the GPU that will be used in the current work is located on the external DelftBlue supercomputer, provided by the TU Delft [24]. For efficient execution of the project, it is therefore important to develop code that is robust and can be tested locally. Robust and testable code requires a modular software design, which yields the use of independently working modules that can easily be tested, integrated, or used in other simulations. Modules can be any independent piece of coding that serves a clear purpose, for example, specific functions, classes, files, or folders. Modules should have no or just a few dependencies upon other modules. [30]

Automated tests are implemented using Python's 'unittest' framework [104]. This framework enables one to write a large number of classes and methods that use a set of dummy variables to test the relevant algorithms. A large number of tests can be run simultaneously, after which one gets a clear overview of passing and failing tests. This leads to the efficient development of sound code. If possible, tests are made suitable for CPU execution, so they can be performed locally. To quickly execute GPU-based tests, the Tesla T4 GPU of Google Colab [41] is used to avoid waiting times for the DelftBlue GPUs.

4 Validation of Local Grid Refinement

In this research, steady-state freezing of turbulent channel flow is simulated using a Large Eddy Simulation. As was mentioned in Sec. 2.5.2, LES needs refinement near the walls due to the local an-isotropic turbulent behavior and small turbulent length scales. Because this is not the case in the center of the channel, significant computational acceleration can be achieved by adopting a local coarse grid here. This chapter will validate the two approaches of Local Grid Refinement (LGR) that were discussed in Sec. 3.6.2 and 3.6.3. The first approach is a GPU-based version of the method developed by [92], while the second approach is a novel one that aims to overcome some of the non-physical behavior encountered in the first approach. Both are tested using a grid convergence study, after which the best-performing option is selected for further use. The aim of this chapter is not to assess the code acceleration achieved by the GPU; this becomes relevant in the turbulent simulations that will be discussed in Ch. 5.

4.1. Computational Setup

In this work, a D3Q19 scheme has been adopted in the Filter-Matrix Lattice Boltzmann framework. A base code has been made available within the RPNM department of TU Delft that can simulate a Poiseuille flow profile on the CPU using an FMLB-D3Q19 scheme. This base code has been converted such that it can run on a Graphical Processing Unit. All simulations in the current research are performed on an NVIDIA Tesla A1000 GPU, which is facilitated by the DelftBlue supercomputer of the TU Delft [24]. The hardware specifications of the used GPU are listed in Tab. 4.1. The simulations are written in Python and all communication with the GPU is realized using the CUDA back-end that is included in the Numba library [83].

The flow case for validating the LGR models is a laminar channel flow between parallel plates on which no-slip conditions are imposed. To this end, the stream- and span-wise directions have periodic boundary conditions, while a half-way bounce-back scheme is adopted in the wall-normal direction. The initial velocity of the simulations is set to zero by initializing the domain with a zero-velocity equilibrium distribution (Eq. 2.19) on all nodes. At $t = 0$, a body force $(f_b, 0, 0)$ is imposed in the stream-wise direction and the simulation runs until a steady-state developed profile is obtained. The domain is split into two levels of refinement with a fine layer adjacent to each wall and a coarse layer in the center of the channel. The fine layers both have N_{y_f} cells along the y-dimension, while the coarse layer has $N_{y,c} = N_y - 2N_{y,f}$ cells along the y-dimension. The ratio between coarse and fine cell spacing is determined by the refinement factor $r = \frac{\Delta y_f}{\Delta y_c}$. The grid cells are cubic, such that the local cell spacing is the same in all Cartesian directions. The setup is schematically shown in Fig. 4.1b. The fine layer at low-y is referred to as the ”first” fine layer and the fine layer at high-y is referred to as the ”second” fine layer.

To study the performance of the LGR algorithms, simulations have been performed at different grid resolutions by varying the number of cells in the grid. The same physical situation is retrieved over different grid resolutions by correspondingly scaling the input parameters. If N_x is increased by a factor s, the dimensions N_y and N_z, viscosity ν, and body force f_b are also scaled. The correct scaling

Table 4.1: Specifications of the used GPU hardware [24, 85]

GPU Model	NVIDIA Tesla V100S
Number of SM	4
Single precision Cores / SM	64
Threads / warp	32
Max warps / SM	64
Max threads / SM	2048
Max blocks / SM	32
Max 32-bit registers / SM	65536
Max registers / block	65536
Max registers / thread	255
Max threads / block	1024
Shared Memory / SM (kB)	64-96
Local Memory per Thread (kB)	512
Global Memory (GB)	32
Memory Bandwidth (GB/s)	1134
Peak Float32 FLOPS	15.7

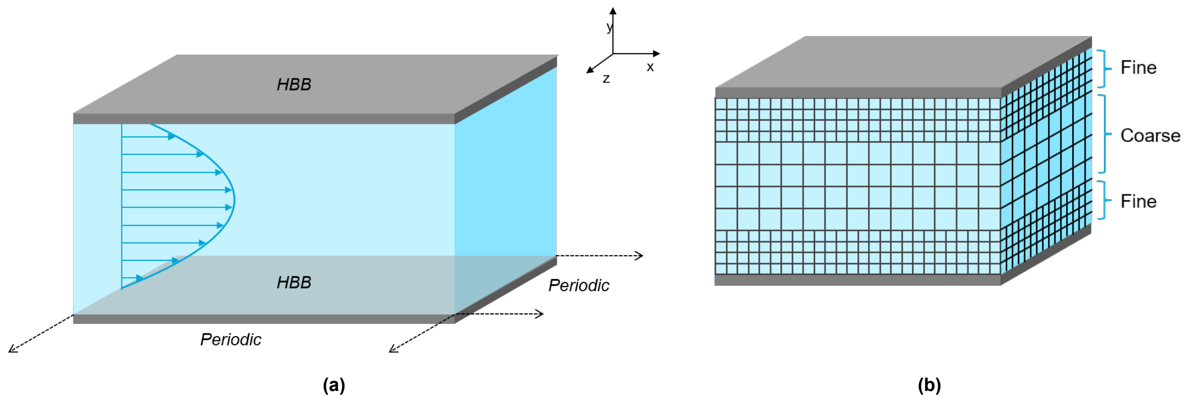


Figure 4.1: (a) The 3D laminar flow case with periodic boundary conditions in the stream- and span-wise directions and a half-way bounce back scheme in the wall-normal direction. (b) The adopted grid uses fine cells near the walls and coarse cells at the center of the channel.

follows from dimensional analysis and is given by

$$N_{y,1} = sN_{y,0}, \tag{4.1}$$

$$N_{z,1} = sN_{z,0}, \tag{4.2}$$

$$\nu_1 = s\nu_0, \tag{4.3}$$

$$f_{b,1} = \frac{1}{s} f_{b,0}, \tag{4.4}$$

where subscript 1 denotes the new value and subscript 0 denotes the initial value. The parameters that correspond to the lowest applied grid resolution are summarized in the middle column of Tab. 4.2, expressed in fine lattice units. The right column contains the scaling factors that must be applied to retrieve a variable in coarse lattice units. The simulation was repeated for the following y-dimensions:

$$N_y = \{32, 48, 72, 96, 120, 144, 168, 192\} \tag{4.5}$$

4.2. Algorithm Performance

The performance of the two LGR algorithms have been assessed using the L2 error norm that was introduced in Sec. 3.6.5. This has been done for the grid configurations that correspond to the listed values of N_y in Eq. 4.5. The simulation was ended when either one of these conditions were fulfilled:

Table 4.2: Input parameters to the LGR simulations with the coarsest grid. The values are given in terms of fine and coarse units. Refinement factors and numbers of cells have no unit, so their values are given in the rightmost column. Parameters of simulations with finer grids follow from Eqs. 4.1-4.4. The refinement factor does not change over different resolutions.

Description	Quantity	Fine units	Coarse units	No unit
x-dimension of grid	N_x	8	4	-
y-dimension of grid	N_y	32	16	-
z-dimension of grid	N_z	8	4	-
Refinement factor	r	-	-	2
Number of fine layer cells along y	$N_{y,f}$	-	-	8
Number of coarse layer cells along y	$N_{y,c}$	-	-	8
Viscosity	ν	1/9	2/9	-
Body force	g	0.000075	0.00015	-

1. The L2 difference reached a value $\epsilon_{\text{diff, L2}} < 10^{-7}$;
2. The L2 difference stopped decreasing.

The choice for a threshold at 10^{-7} in the first condition was substantiated in Sec. 3.6.5. For the definition of $\epsilon_{\text{diff, L2}}$, one can resort to Eq. 3.56.

Simulations have been performed for the novel algorithm at $r = 2$ and for the Rohde algorithm at $r = 2$ and $r = 4$. Rohde et al. [92] have already proven the performance of their algorithm on a CPU in the $r = 2$ case, but, to the writer's knowledge, the $r = 4$ case has not yet been applied up to now. An additional simulation has been performed without refinement, or equivalently, $r = 1$.

4.2.1. Time Convergence

The L2 differences have been plotted in Fig. 4.2 against the simulation time for the finest grid (i.e., $N_y = 192$). It can be seen that the novel algorithm does not converge to a steady state where $\epsilon_{\text{diff, L2}} < 10^{-7}$; around time step $t = 5 \cdot 10^3$, the L2 difference stagnates at $\epsilon_{\text{diff, L2}} = 10^{-2}$. This is in contrast to the results of the Rohde algorithm, which showed a logarithmic time convergence to a steady state for both the $r = 2$ and the $r = 4$ case.

When zooming in on the velocity profiles of the different simulations in Fig. 4.3a, it can be observed that all simulations follow the analytical solution very well, except for the novel algorithm which shows an overshoot of 2.6%. When plotting velocity derivatives du_x/dy in Fig. 4.3b, jumps are observed in the profile of the novel algorithm around $y = 48$ and $y = 144$, which corresponds to the locations of the fine-coarse layer interfaces. From these results, it can be concluded that information is not accurately transferred between adjacent layers, which leads to a false estimation of the flow profile.

4.2.2. Grid Convergence

Next, L2 errors $\epsilon_{\text{L2, err}}$ have been calculated for the different simulations at the resolutions mentioned in Eq. 4.5. The definition of $\epsilon_{\text{L2, err}}$ in Eq. 3.57 has been used and the analytical solution is given by the Poiseuille flow expression in Eq. 2.7. In Fig. 4.4, the errors have been plotted against the reciprocal of the domain size in the y-direction, measured in fine units.

From the plotted errors, it can be observed that the error $\epsilon_{\text{L2, err}}$ of the Rohde profiles (both $r = 2$ and $r = 4$) relates to $1/N_y$ as

$$log\left(\epsilon_{\text{L2, err}}\right) = C + p \cdot log\left(\frac{1}{N_y}\right), \tag{4.6}$$

where C is a constant and p is the order of convergence. From the figure, it can be determined that $p \approx 2$ for both refinements of the Rohde algorithm, which is also observed for the $r = 1$ simulation. The latter error originates from the half-way bounce-back method that is applied at the solid walls [92]. Because the order of convergence is the same for both Rohde algorithms, it can be concluded that the leading term of the truncation error is not caused by the introduction of the local grid refinement technique [28]. The novel algorithm does not show grid convergence, however, meaning that the truncation error is not dominant here. Instead, there is an error associated with the model that causes inaccuracies on all levels of refinement.

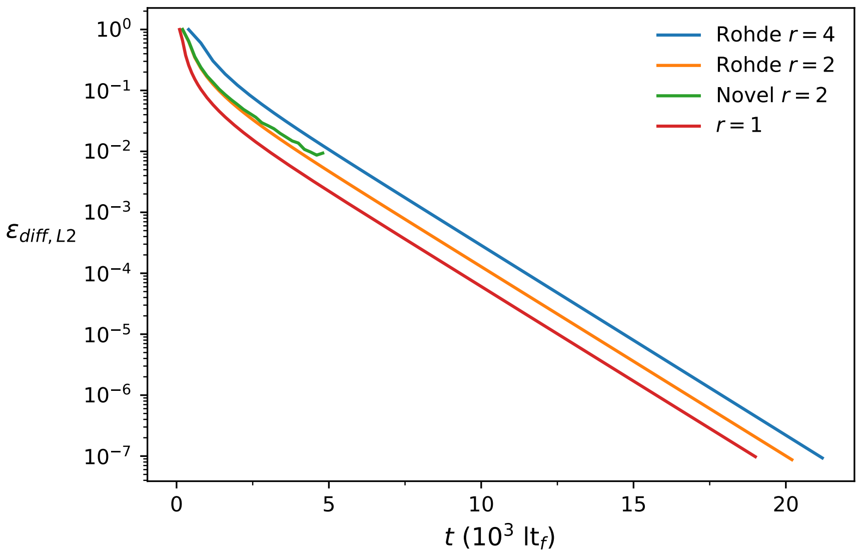


Figure 4.2: The L2 difference of the $N_y = 192$ grid plotted against simulation time in fine lattice units. Results are shown for the Rohde algorithm at refinement factors $r = 2$ and $r = 4$, and for the novel algorithm at $r = 2$. Results from a benchmark simulation without refinement ($r = 1$) are also shown.

4.2.3. Sources of Inaccuracy

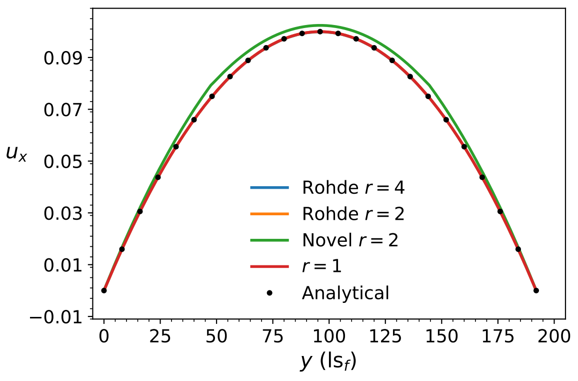
When looking at the different steps in the novel algorithm, the most likely source of the observed inaccuracies is the information transfer from coarse to fine cells (step 5 in Sec. 3.6.3). In this step, the information of one coarse cell is equally distributed over multiple co-located fine cells, which is the only step in the algorithm that yields a loss of information.

This hypothesis is substantiated by Qi et al. [87], who also applied a 3D hierarchical grid refinement technique similar to the one proposed in the current work. They similarly applied an interface region consisting of two coarse cells (i.e., two fine ghost cells and one coarse ghost cell). A major difference is that they adopted a space-time interpolation scheme to project coarse cells onto the fine layer, thereby suppressing information loss on the fine grid level. This approach was proven to be successful. It is therefore interesting to apply a comparable type of grid interpolation to the algorithm presented in the current work. However, because the Rohde algorithm gave satisfactory results and given the already broad scope of the current research, it was not attempted to improve the novel algorithm.

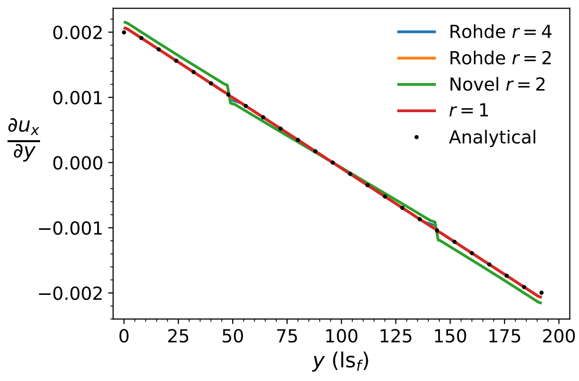
Furthermore, although the Rohde algorithm is second-order convergent, its errors are larger for increasing refinement as can be seen by the different offsets of the unrefined $r = 1$, Rohde $r = 2$, and Rohde $r = 4$ lines in Fig. 4.4. For the refined simulations, an error is introduced by the non-physical over-collisions that appear in the fine layer during the non-synchronous step in the algorithm. The mechanism of over-collisions has been explained in Sec. 3.6.2. There are more over-collisions for the $r = 4$ case due to a larger number of asynchronous steps, which causes a greater error. This error is complemented by an increased truncation error that results from the larger size of coarse cells.

4.3. Conclusion

It has been shown that the Rohde algorithm performed superior to the novel grid refinement technique proposed in the current work. This conclusion is based on both time and grid convergence studies, which showed that the novel algorithm introduces inaccuracies near the fine-coarse layer interface that results in a laminar flow solution with a grid-independent leading error. On the other hand, the Rohde algorithm proved to be second-order accurate for both the $r = 2$ and $r = 4$ algorithms with laminar flow parallel to the layer interface. The latter algorithm is applied in the remainder of this study and its performance is further investigated for turbulent channel flows in the next chapter.



(a)



(b)

Figure 4.3: (a) Velocity profiles u_x and (b) derivatives du_x/dy for the different algorithms, compared to the analytical Poiseuille profile. The simulation results represent intersections at $x = N_x/2$ and $z = Nz/2$.

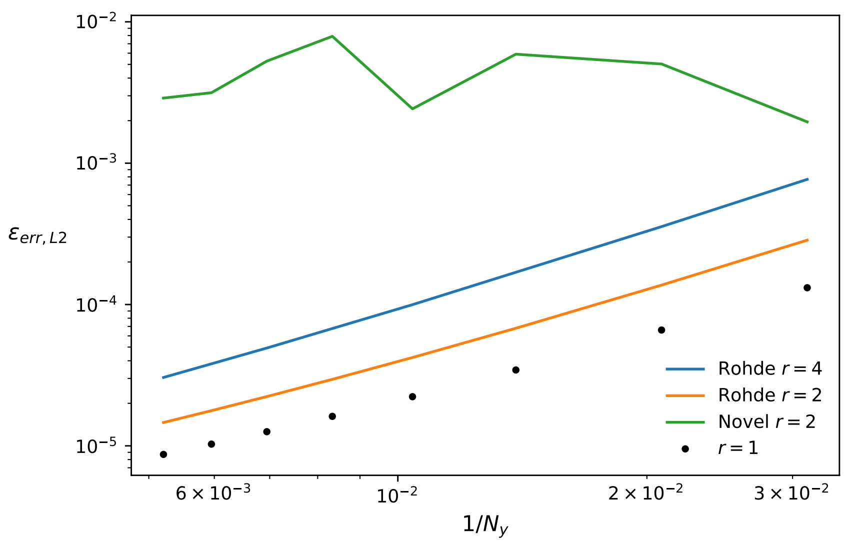


Figure 4.4: L2 error norm against the inverse y-dimension of the grid, measured in fine cells. Results are given for the Rohde algorithm at $r = 2$ and $r = 4$, the novel algorithm at $r = 1$, and a non-refined case at $r = 1$. The novel algorithm shows no grid convergence, while the latter algorithms show second-order grid convergence.

5 Turbulent Flow Validation and Results

The goal of this work is to investigate solidification of turbulent salt flow through a cooled MSFR channel. Because heat transfer in a turbulent flow is not only influenced by diffusion, but also by convection, it is important to get an accurate description of the flow field. This chapter assesses the different methodologies that have been applied in simulating turbulent channel flow. In particular, the local grid refinement technique by Rohde et al. [92], which was found to be accurate in the laminar regime in Ch. 4, has been applied to a DNS- and an LES-FMLBM simulation at different Reynolds numbers. To assess the accuracy of these simulations, existing turbulent statistics of previously performed simulations have been compared with current results. Studies that have been selected are discussed in Sec. 5.2. Results of simulations without the use of a sub-grid scale model (i.e., DNS) are discussed in Sec. 5.3, and results of simulations where sub-grid scales were modeled using the WALE model (i.e., LES) are discussed in Sec. 5.4. The obtained computational speed-ups that resulted from the present GPU-implementation, are presented in Sec. 5.5.

5.1. Computational Setup

In this thesis, turbulent flow simulations have been performed at $Re_\tau = \{180, 395\}$ and at refinement levels $r = 1, 2, 4$ using the Rohde algorithm with a fine-coarse layer interface at $y^+ = 40$. This is similar to the location of the interface in Rohde et al. [92], who also investigated the performance of the $r = 2$ grid. To the author's knowledge, the $r = 4$ grid with a Rohde algorithm has not been investigated yet. Half-way bounce-back boundary conditions are used in the wall-normal direction and periodic boundary conditions are used in the stream- and span-wise directions. A schematic overview of the simulated situation is given in Fig. 5.1. Simulations have been performed both with and without the WALE SGS model.

The input parameters to the $Re_\tau = 180$ and $Re_\tau = 395$ simulations with refinement $r = 2$ have been specified in Tab. 5.1 and 5.2, respectively. Simulation parameters for the $r = 1$ and $r = 4$ simulations can be deduced following the description in Sec. 3.6.4. The initial turbulent velocity and density fields have been provided by [105]. The corresponding initialization procedure has been described in Sec. 3.7. Linear interpolation of initial fields has only been applied for simulations with refinement $r > 1$. Due to the use of initial data, the present simulation needed effectively no start-up time to reach a statistically steady state.

For the $Re_\tau = 180$ simulation, snapshots of the flow have been taken at 80 evenly spaced time intervals of size $\Delta T_{\text{save}} = 6400$ in fine lattice units. This interval corresponds to a distance of 2.5 channels traveled at the bulk velocity. The first snapshot was immediately taken at $t = 6400$ as it was observed that no settling of the flow was necessary to achieve a statistically steady state on the interpolated grids. This led to a total simulation time of $N_{t,\text{max}} = 512000$ in fine lattice units. Similarly, the simulation time for the $Re_\tau = 395$ simulation was $N_{t,\text{max}} = 960000$, and the same amount of snapshots have been using $\Delta T_{\text{save}} = 12800$.

With the current GPU implementation, this corresponds to a total simulation time of less than 2 hours for the $Re_\tau = 180$ simulation with $r = 2$. This is a tremendous speed-up compared to the same simulation by Rohde et al. in 2004 [92], which needed two weeks of simulation time using eight

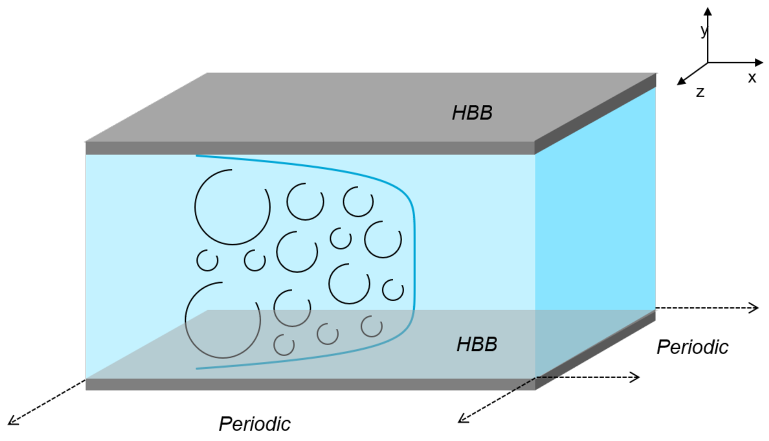


Figure 5.1: The simulated turbulent channel flow situation. HBB boundary conditions are used at the walls and periodic boundary conditions are used in the stream- and span-wise directions.

Table 5.1: Input parameters to the turbulent channel flow simulation with $Re_\tau = 180$ and refinement factor $r = 2$. Quantities are given in fine and coarse units.

Description	Quantity	Fine units	Coarse units
Size of the domain (ls^3)	$N_x \times N_y \times N_z$	$256 \times 128 \times 128$	$128 \times 64 \times 64$
Size of the fine grid, bottom channel (ls^3)	$N_{x,f} \times N_{y,f} \times N_{z,f}$	$256 \times 14 \times 128$	-
Size of the fine grid, top channel (ls^3)	$N_{x,f} \times N_{y,f} \times N_{z,f}$	$256 \times 14 \times 128$	-
Size of the coarse grid (ls^3)	$N_{x,c} \times N_{y,c} \times N_{z,c}$	-	$128 \times 50 \times 64$
Grid spacing (ls)	Δy^+	2.81	5.62
Viscosity (ls^2 · lt^{-1})	ν	2.37 $\cdot 10^{-3}$	1.185 $\cdot 10^{-3}$
Body force (ls^2 · lt^{-1})	g	6.94 · 10^{-7}	1.388 · 10^{-6}
Input density (ls^{-3})	ρ_0	1.0	1.0
Wall shear velocity (ls · lt^{-1})	u_τ	6.67 $\cdot 10^{-3}$	6.67 $\cdot 10^{-3}$
Number of time steps (lt)	$N_{t,\text{max}}$	256 000	512 000

CPUs. Sec. 5.5 provides a comparison between the computational efficiencies achieved in the different simulations.

5.2. Benchmark Studies

To evaluate the accuracy of the performed simulations, a set of previously published studies, similar to those carried out in this research, was selected. In order of publication date, the studies of interest are summarized below. Four DNS studies are summarized first, after which one LES study is summarized.

DNS Studies

1. **Kim et al. (1986) – $Re_\tau = 180$** [55]:
   The first detailed DNS of turbulent channel flow at $Re_\tau = 180$ has been performed by Kim, Moin, and Moser (KMM) in 1986 [55] and is still used as a reliable benchmark nowadays. They applied a spectral method for modeling the dynamics of the flow and used a function to determine the y^+ coordinate of the j'th grid cell from the lower wall:

$$y_j^+ \equiv zu_\tau/\nu = 180(1 - \cos((j-1)\pi/(N_y - 1))),$$

   were $N = 129$ is the number of cells in the wall-normal direction. This led to a very fine spacing $\Delta y^+ = 0.05$ near the walls and a coarser spacing $\Delta y^+ = 4.4$ at the center of the channel.
2. **Amati et al. (1999) – $Re_\tau = 180$** [6]:
   Amati, Succi, and Piva (ASP) performed a DNS simulation of turbulent channel flow with $Re_\tau = 180$ in 1999, using the same grid and input parameters as have been used in the $Re_\tau = 180$ simulation of the current work. It is convenient to compare results with an identical simulation for proper benchmarking. ASP used an LBGK framework with a constant grid spacing $\Delta y^+ = 2.8$.

Table 5.2: Input parameters to the turbulent channel flow simulation with $Re_\tau = 395$ and refinement factor $r = 2$. Quantities are given in fine and coarse units.

Description	Quantity	Fine units	Coarse units
Size of the domain (ls^3)	$N_x \times N_y \times N_z$	$460 \times 230 \times 230$	$230 \times 115 \times 115$
Size of the fine grid, bottom channel (ls^3)	$N_{x,f} \times N_{y,f} \times N_{z,f}$	$460 \times 12 \times 230$	-
Size of the fine grid, top channel (ls^3)	$N_{x,f} \times N_{y,f} \times N_{z,f}$	$460 \times 12 \times 230$	-
Size of the coarse grid (ls^3)	$N_{x,c} \times N_{y,c} \times N_{z,c}$	-	$230 \times 103 \times 115$
Grid spacing (ls)	Δy^+	3.43	6.87
Viscosity (ls^2 · lt^{-1})	ν	1.664 $\cdot 10^{-3}$	8.32 $\cdot 10^{-4}$
Body force (ls^2 · lt^{-1})	g	2.839 · 10^{-7}	5.678 · 10^{-7}
Input density (ls^{-3})	ρ_0	1.0	1.0
Wall shear velocity (ls · lt^{-1})	u_τ	5.71 $\cdot 10^{-3}$	5.71 $\cdot 10^{-3}$
Number of time steps (lt)	$N_{t,\text{max}}$	460 000	920 000

Table 5.3: Specifications of previously conducted turbulent channel flow simulations that are comparable to simulations in the current work.

Author	Abbrev.	Re_τ	(N_x, N_y, N_z)	$\Delta^+_{y,\text{min}}$	$\Delta+_{y,\text{max}}$	Dynamics	SGS model
Kim et al.	KMM	180	(192, 129, 160)	0.05	4.4	Spectral	-
Amati et al.	ASP	180	(256, 128, 128)	2.8	2.8	BGK-LBM	-
Rohde et al.	RKD	180	(256, 128, 128)	2.8	5.6	FMLBM	-
Moser et al.	MKM	395	(256, 193, 192)	0.02	6.5	Spectral	-
Zhuo et al.	ZZ	180	(270, 135, 90)	4	4	FMLBM	Vreman
	ZZ	180	(540, 270, 180)	2	2	FMLBM	Vreman

3. **Moser et al. (1999) – $Re_\tau = 395$** [76]:
   Moser, Kim, and Mansour (MKM) performed DNS simulations at Reynolds numbers $Re_\tau = \{180, 395, 590\}$ using the same approach as KMM. The $Re_\tau = 395$ case will be used for benchmarking in the current work. The corresponding grid spacing is calculated in the same way as KMM, which leads to a spacing $\Delta y^+ = 0.02$ at the wall and $\Delta y^+ = 7.7$ at the center of the channel.
4. **Rohde et al. (2006) – $Re_\tau = 180$** [92]:
   Rohde, Kandhai, Derksen, and van den Akker (RKD) performed the same DNS simulation as ASP but applied a filter-matrix LB scheme with an $r = 2$ grid refinement following the Rohde algorithm, which is also applied in the current work. The fine-coarse interface was located at $y^+ = 40$, which corresponds to $N_{y,f} = 14$ fine cells measured from the wall. This led to a grid spacing $\Delta y^+ = 2.8$ in the fine layer and $\Delta y^+ = 5.6$ in the coarse layer.

LES Study

1. **Zhuo and Zhong (2016) – $Re_\tau = 180$** [129]:
   Zhuo and Zhong performed an LES-FMLBM simulation of turbulent channel flow with $Re_\tau = 180$ and constant grid spacing. They adopted the Vreman SGS model, which was discussed in Sec. 3.5.2. Simulations were performed for two types of grids; a coarse grid with homogeneous $\Delta y^+ = 4$ and a fine grid with homogeneous $\Delta y^+ = 2$.

The specifics of the above studies are summarized in Tab. 5.3.

5.3. Direct Numerical Simulation

The obtained turbulent statistics of the performed DNS simulations are now compared with the DNS simulations discussed in the previous section. The statistics of interest are the mean stream-wise velocity, the RMS velocity fluctuation, and the Reynolds stress.

Mean Velocity

Fig. 5.2a shows the mean stream-wise velocity profile of the $Re_\tau = 180$ simulation with refinements $r = 2$ and $r = 4$ and the $Re_\tau = 395$ simulation with refinement $r = 2$. The former is compared with data of KMM and ASP and the latter is compared with data of MKM, both with analytical formulas that give expected profiles in the viscous sub-layer and log layer. As was also observed by RKD for $Re_\tau = 180$, the $r = 2$ velocity profiles approximate the results for a uniform, fine grid very closely. For both Reynolds

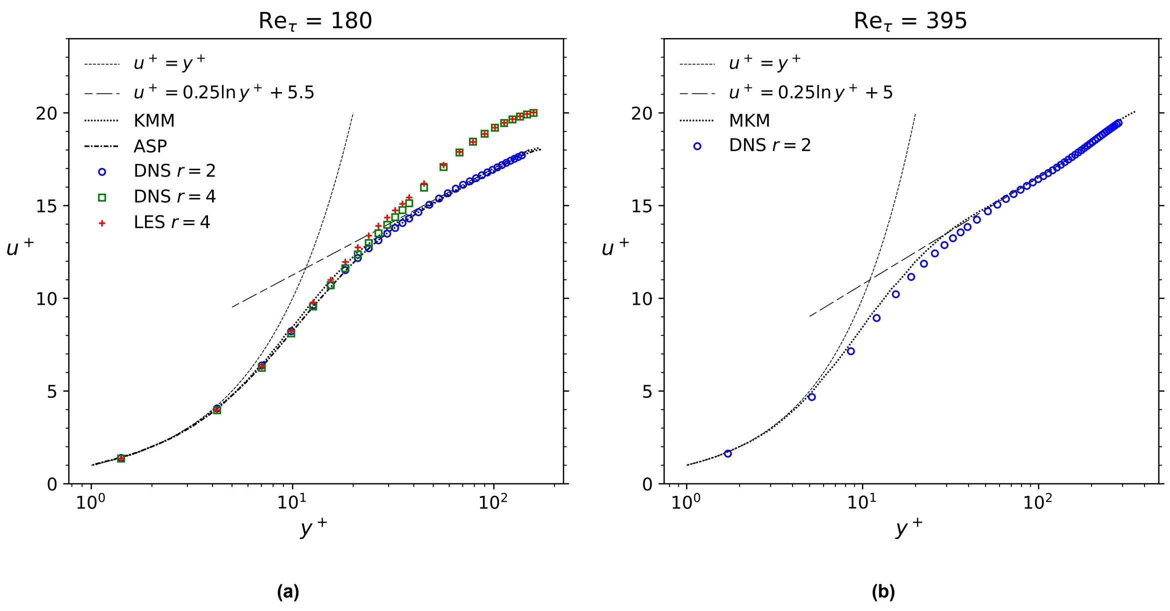


Figure 5.2: Mean stream-wise velocity u^+ against the wall distance y^+ for turbulent channel flow at (a) $Re_\tau = 180$, and (b) $Re_\tau = 395$. The grid has cubic cells of size $\Delta^+ = 2.8$ in the range of $y^+ = 0$–40 and $\Delta^+ = 5.6$ elsewhere. The results of Amati et al. [6] and Kim et al. [55] are also shown for comparison. Theoretical profiles are included for both the viscous sublayer and the log-layer.

numbers, there is a slight underestimation compared to KMM and MKM, which was also observed by ASP at $Re_\tau = 180$ and attributed to a lack of grid resolution Δy^+ near the wall. In contrast, the $r = 4$ velocity profile shows a significant deviation of up to 12% from the benchmark, which develops around the refinement interface ($y^+ = 40$).

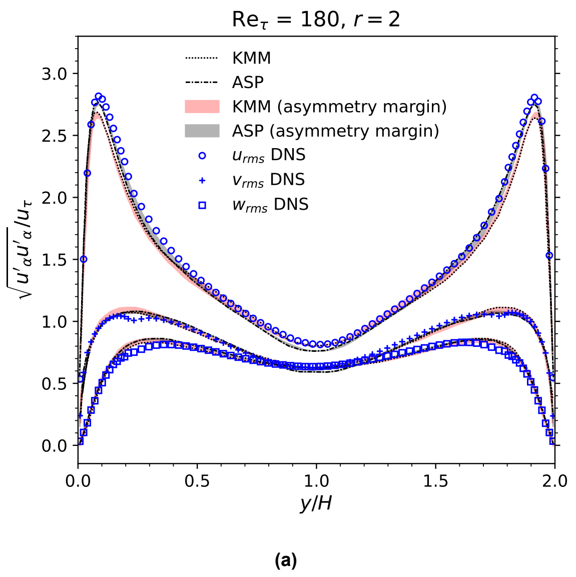
The reason for the observed inaccuracy at $r = 4$ is two-fold. First of all, the lower resolution in the coarse layer leads to decreased accuracy because less turbulent structures can be resolved. The local Kolmogorov length scale in the logarithmic layer follows $\eta u_\tau/\nu = (\kappa y^+)$ [86], where $\kappa \approx 0.4$ denotes the Von Karman constant. This yields a value $\eta u_\tau/\nu \approx 2$ at the refinement interface ($y^+ = 40$), while the coarse grid spacing with $r = 4$ is $\Delta y^+ = 10.2$. Generally, a grid cell can be roughly twice as large as the Kolmogorov length scale [92], so the large grid cells are too large for suitable resolution. To see if the inaccuracy could be resolved by modeling the sub-grid scales, an LES with $r = 4$ was performed using a similar procedure as will be discussed in the next section. No improvement was observed compared to the $r = 4$ DNS, so, likely, a lack of grid resolution is not the only cause of inaccuracy.

The second reason for the inaccuracy at $r = 4$ lies in the non-zero y-velocity component inherent in turbulent flows, directed normally to the fine-coarse layer interface. This conclusion is further supported by RKD's experiments, where laminar flow studies with various orientations revealed that a velocity normal to the grid transition interface introduced a staggered solution. As mentioned in Sec. 3.6.2, the Rohde algorithm introduces an error due to additional non-physical collisions in the non-synchronous iteration step. Nonetheless, no significant error is observed for the $r = 2$ mean stream-wise velocity. This discrepancy arises because there is only one non-synchronous iteration step for $r = 2$, while there are three non-synchronous iteration steps for $r = 4$. As a result, the non-physical collisions for $r = 2$ occur in only one row of cells adjacent to the interface, as opposed to three rows of cells for the $r = 4$ case.

Due to the substantial deviations observed in the $r = 4$ velocity profiles, only refinement factors up to $r = 2$ will be considered for the remainder of this study.

RMS Velocity Fluctuation

The root mean square (RMS) velocity fluctuations of the $Re_\tau = 180$ and $Re_\tau = 395$ simulations with refinement $r = 2$ are depicted in Figure 5.3a and 5.3b, respectively. The results of KMM, ASP, and MKM are included for comparison. For $Re_\tau = 180$, the data of the full channel is presented, whereas, for $Re_\tau = 395$ only the first 100 wall units are displayed to enable a clear comparison with MKM. Furthermore, the data of ASP and KMM are slightly asymmetric in $y/H = 1$, so to facilitate better



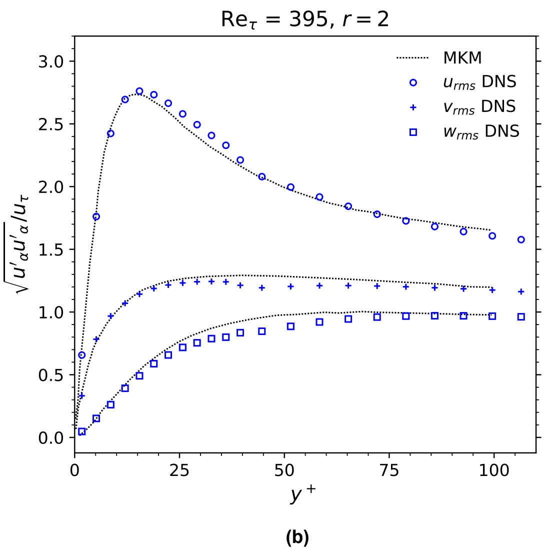


Figure 5.3: RMS velocity fluctuation components $\sqrt{u_\alpha u_\alpha}$ normalized by shear velocity u_τ against the wall distance y^+ for turbulent channel flow at (a) $Re_\tau = 180$, and (b) $Re_\tau = 395$. The grid has cubic cells of size $\Delta^+ = 2.8$ in the range of $y^+ = 0$–40 and $\Delta^+ = 5.6$ elsewhere. The results of Amati et al. [6] and Kim et al. [55] are also shown for comparison. Theoretical profiles are included for both the viscous sublayer and the log-layer.

comparison, the area between the original RMS profile and the mirrored version is highlighted for both studies.

For $Re_\tau = 180$, the u_{rms} profile of the current study almost completely overlaps with the highlighted ASP region, which leads to the conclusion that $r = 2$ refinement barely influences the RMS fluctuation in the stream-wise direction. For $Re_\tau = 395$, the u_{rms} results are also relatively accurate, except for a slight deviation from MKM around $y/H = 0.25$, which can also be observed when comparing the $Re_\tau = 180$ results with KMM. According to ASP, this was caused by a lack of resolution near the wall. Because MKM and KMM adopted a similar approach with a high near-wall resolution, the same explanation is valid for the $Re_\tau = 395$ case.

In the wall-normal and span-wise directions, a slight dip is observed around the refinement interface for both Reynolds numbers. The same was observed by RKD, who mentioned the grid refinement technique and a lack of grid resolution at $y^+ > 40$ as potential reasons for the smaller values.

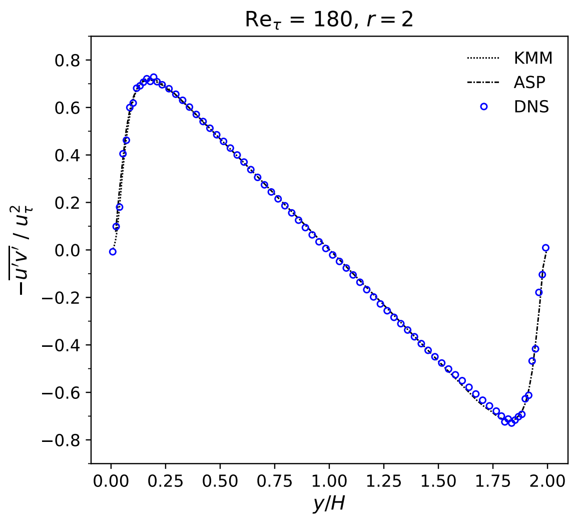
Reynolds Stress

The Reynolds stress profiles are shown in Figs. 5.4a and 5.4b for the $Re_\tau = 180$ and $Re_\tau = 395$ simulations, respectively. For the former simulation, the results are clearly in good agreement with the data of KMM and ASP. However, for the $Re_\tau = 395$ simulation, a slight under-prediction of 6% is observed at the peak. Similar FMLB simulations have been performed in the MSc Theses of Van Bemmelen [105] and Wortelboer [119], who both used a continuous grid with $\Delta y^+ = 3.4$. Interestingly, Van Bemmelen did not observe an under-prediction of the Reynolds stress with $Re_\tau = 395$, while Wortelboer observed an under-prediction of 2%. However, van Bemmelen found an under-prediction of 3-5% for Reynolds stress with $Re_\tau = 180$ in the region $0.15 < y/H < 0.6$. Such under-prediction was not observed in current simulations, nor in simulations by RKD. Because these works found under-predictions of Reynolds stress across different simulations, the observed deviation in Fig. 5.4b is assumed to be acceptable.

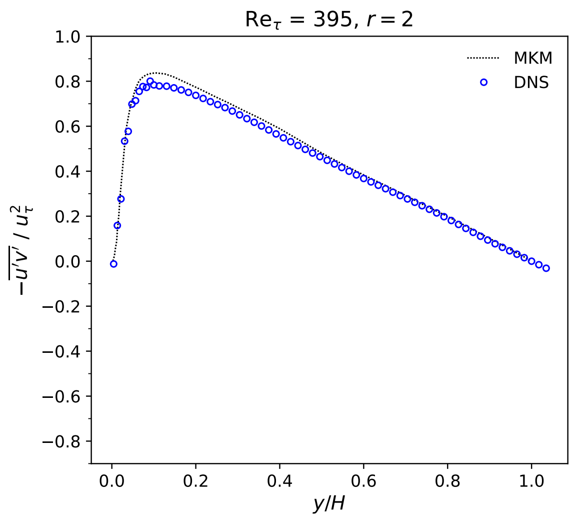
Furthermore, a slight staggering of the Reynolds stress data points is observed in the fine layer for both Reynolds numbers. At $r = 1$ this was not observed, so this can be attributed to the adopted refinement technique. However, it must be noted that the staggering was less intense in the results of RKD.

5.4. Large Eddy Simulation

In an attempt to obtain a more accurate and complete description of turbulent channel flow, a Large Eddy Simulation (LES) has been implemented using the WALE model with $C_w = 0.5$, following the



(a)



(b)

Figure 5.4: Reynolds stress against the wall distance y^+ for turbulent channel flow at (a) $Re_\tau = 180$, and (b) $Re_\tau = 395$. The grid has cubic cells of size $\Delta^+ = 2.8$ in the range of $y^+ = 0$–40 and $\Delta^+ = 5.6$ elsewhere. The results of Amati et al. [6] and Kim et al. [55] are also shown for comparison. Theoretical profiles are included for both the viscous sublayer and the log-layer.

methodology that was described in Sec. 3.5. The only other FMLBM-LES study of 3D turbulent channel flow was performed by Zhuo and Zhong [129] (ZZ), who implemented the Vreman model that was mentioned in Sec. 3.5.2. However, a WALE model implementation has not yet been encountered in the filter-matrix approach. In addition, this thesis is the first to implement LES-FMLBM using a local grid refinement technique.

The turbulent statistics of the LES simulations will now be discussed and compared with the DNS and LES simulations that were discussed in Sec. 5.2. The statistics of interest are the mean stream-wise velocity, the RMS velocity fluctuation, the Reynolds stress, and the mean eddy viscosity.

Mean Velocity

In Fig. 5.5a and 5.5b, the mean stream-wise velocity profiles of the performed LES $r = 2$ simulations have been compared with the corresponding DNS results that were introduced in the previous section. The data of KMM, ASP, MKM, and the analytical formulas have also been included for completeness. A promising observation for both Reynolds numbers is that the LES profiles are closer to the fine-grid results of KMM and MKM, compared to the performed DNS simulations. This is improvement is most significant around $y^+ = 25$ for $Re_\tau = 180$ and around $y^+ = 35$ for $Re_\tau = 395$. For both LES results, the maximum deviations from KMM and MKM are halved. Because the $Re_\tau = 180$ DNS results were already very accurate, the improvement is hardly visible in Fig. 5.5a. For $Re_\tau = 395$, however, the deviation of the DNS result compared to MKM is slightly larger. Hence, the improved accuracy of the corresponding LES simulation is clearer. In the WALE-LES of $Re_\tau = 180$ turbulent channel flow by [125], who applied a discrete unified gas-kinetic scheme (DUGKS), it was also observed that the use of a sub-grid scale model led to a slightly larger mean stream-wise velocity, compared to DNS. The obtained results are therefore not surprising.

RMS Velocity Fluctuation

The RMS velocity fluctuations are plotted in Fig. 5.6a and 5.6b. It is noted that all three components show similar behavior to the DNS data. Thus, the performed LES introduces no improvement for the RMS fluctuation. The stream-wise velocity fluctuation u_{rms} shows slightly higher peaks for the LES data, while v_{rms} and w_{rms} are slightly lower around the fine-coarse layer interface. In the LES of ZZ, a comparable overshoot with respect to KMM was also observed in the stream-wise direction. Similar to the previously mentioned hypothesis of ASP, ZZ attributed the overshoot to a lack of grid resolution as they observed better results for a finer grid ($\Delta y^+ = 2$ instead of $\Delta y^+ = 4$).

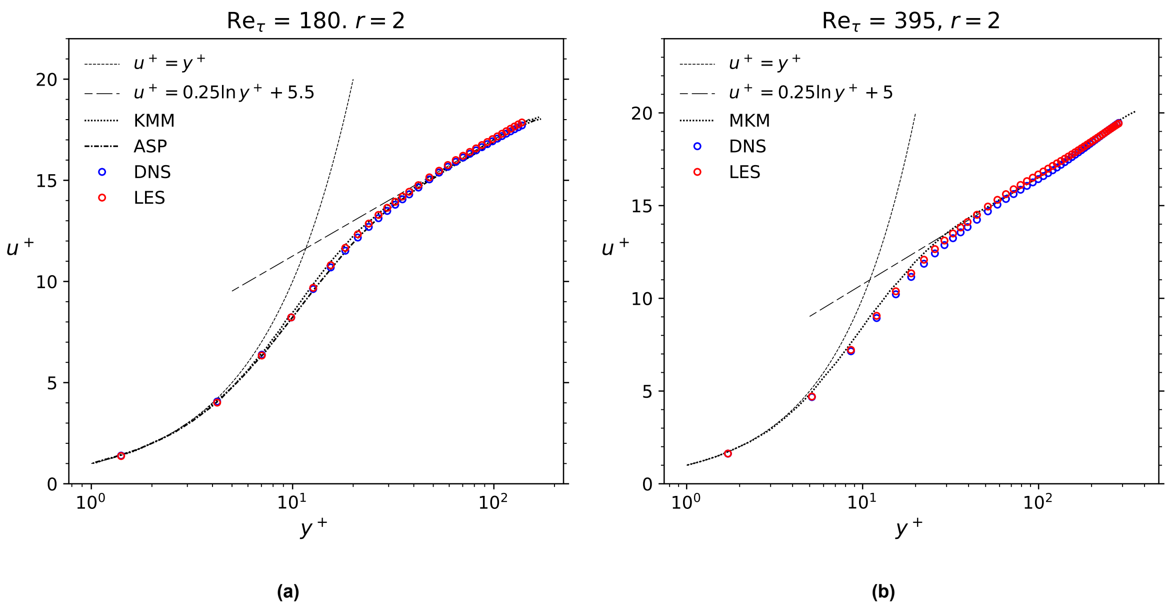


Figure 5.5: Mean stream-wise velocity u^+ against the wall distance y^+ for turbulent channel flow at (a) $Re_\tau = 180$, and (b) $Re_\tau = 395$. The grid has cubic cells of size $\Delta^+ = 2.8$ in the range of $y^+ = 0$–40 and $\Delta^+ = 5.6$ elsewhere. The results of Amati et al. [6] and Kim et al. [55] are also shown for comparison. Theoretical profiles are included for both the viscous sublayer and the log-layer.

Reynolds Stress

The Reynolds stress results of the LES and DNS simulations are shown in Fig. 5.7a and Fig. 5.7b. The profiles overlap very precisely, hence, the LES does not significantly influence Reynolds stress in the current simulations. The same result was found in the WALE-LES of turbulent channel flow by [125], who applied a discrete unified gas-kinetic scheme (DUGKS).

Mean Eddy Viscosity

The mean eddy-viscosity profiles for both Reynolds numbers are shown in Fig. 5.8. No benchmark has been plotted, due to the dependency of ν_t on the adopted grid. Furthermore, separate eddy-viscosity profiles are often not reported in the literature. From a qualitative point of view, the shapes of the plotted profiles make sense as they resemble the shape of the RMS velocity fluctuations in Fig. 5.6. As explained in Sec. 2.4.1, the sum of squared velocity fluctuations is proportional to the turbulent intensity. This quantity is a measure for the sub-grid scale dissipation and, consequently, for the eddy-viscosity.

Another observation is that the $Re_\tau = 395$ profile has higher values than the $Re_\tau = 180$ profile. This can be explained by (1) the larger dimensionless grid spacing $\Delta y^+ = 3.4$ for $Re_\tau = 395$, compared to $\Delta y^+ = 2.8$ for $Re_\tau = 180$; (2) the smaller turbulent scales of the $Re_\tau = 395$ simulation, which follows from the relation $\eta/\ell_0 \sim Re^{-3/4}$ in Eq. 2.67. These two arguments yield a higher concentration of turbulent energy in the sub-grid scales. Thus, the eddy-viscosity is expected to be larger.

It can also be observed that slight dips are present at the layer interfaces. This is a similar observation to the dips in the v_{rms} and w_{rms} profiles, which lead to a dip in turbulent intensity. This will have contributed to the dips observed for the eddy-viscosity profiles.

5.5. GPU Performance

All simulations have been performed on an NVIDIA Tesla V100S GPU with specifications as listed in Tab. 4.1. In this section, the performance of the DNS and LES simulations of turbulent channel flow will be presented. In addition, attempts for speeding up calculations are evaluated.

5.5.1. Measure of Performance

The speed of an LB algorithm can be measured in terms of the number of cell evaluations per second, better known as lattice updates per second. Because this number is often very large for modern com-

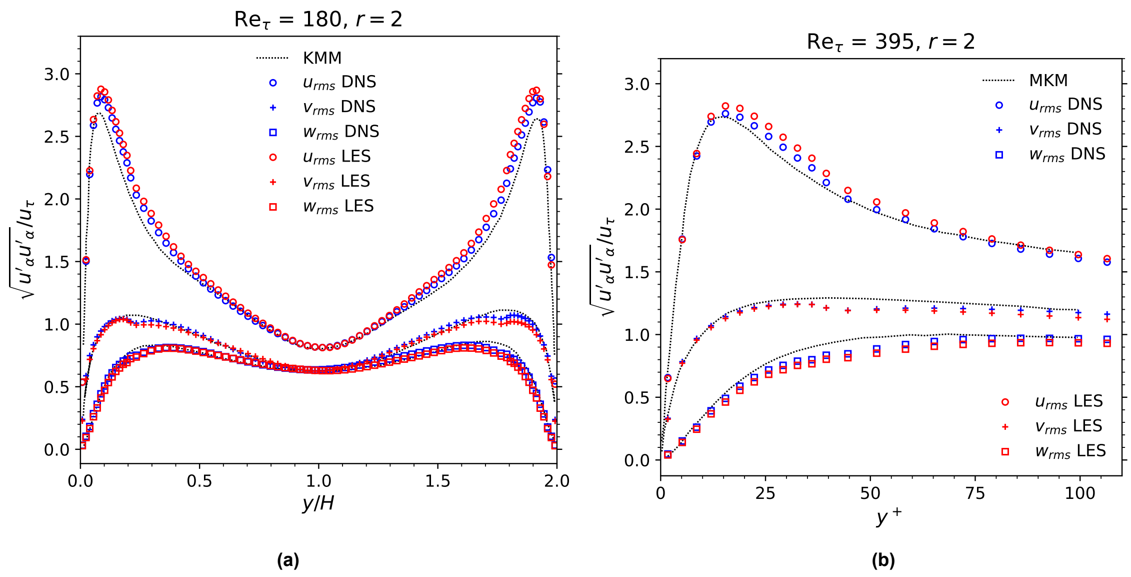


Figure 5.6: RMS velocity fluctuation components $\sqrt{u_\alpha u_\alpha}$ normalized by shear velocity u_τ against the wall distance y^+ for turbulent channel flow at (a) $Re_\tau = 180$, and (b) $Re_\tau = 395$. The grid has cubic cells of size $\Delta^+ = 2.8$ in the range of $y^+ = 0$–40 and $\Delta^+ = 5.6$ elsewhere. The results of Amati et al. [6] and Kim et al. [55] are also shown for comparison. Theoretical profiles are included for both the viscous sublayer and the log-layer.

puters, one commonly speaks of million lattice updates per second (MLUPS). This quantity is defined as

$$\text{MLUPS} = \frac{N_{\text{cells}} N_{T,\text{max}}}{T} \cdot 10^{-6}, \tag{5.1}$$

where N_{cells} is the number of cells in the flow domain, $N_{T,max}$ is the number of time steps, and T is the simulation time. The simulation time is taken in fine units.

The theoretical maximum performance of a GPU is determined by its memory bandwidth BW, which is 1134 GB/s for the NVIDIA Tesla V100S (see Tab. 4.1), and the amount of memory that is accessed by each node within one iteration step [123]. Variables are stored with single precision, which corresponds to 4 bytes of memory per variable. One thread (a grid node) accesses N_{vars} variables at each time step, which comprises of 19 velocity components f_i that are accessed multiple times and some auxiliary variables. Based on this amount, the maximum theoretical MLUPS becomes

$$\text{MLUPS}_{\text{max, theory}} = \frac{\text{BW}}{4N_{\text{vars}}} \cdot 10^{-6}. \tag{5.2}$$

In the MRT-LBM of [123], which used a D3Q19 + D3Q7 thermal model, each node accessed 143 variables per time iteration. Because the current simulations have no thermal model yet, a rough estimate of 105 variables per node per time step is assumed. The theoretical maximum performance then amounts to 2700 MLUPS. However, according to [23] and [34], the maximum performance is around 80% of the theoretical maximum, resulting in a maximum achievable performance of 2160 MLUPS with the current GPU.

5.5.2. Simulation Speed

This section discusses the achieved simulation speed for the different types of simulations that have been performed. Fig. 5.9 gives an overview of the MLUPS per simulation type. A distinction has been made between LES and DNS, $r = 1$, $r = 2$, $r = 4$, and $Re_\tau = 180$ and $Re_\tau = 395$ simulations. First, the differences in simulation speed among the performed simulations are discussed. Subsequently, the observed performance is compared with the achievable maximum that was discussed in the previous section.

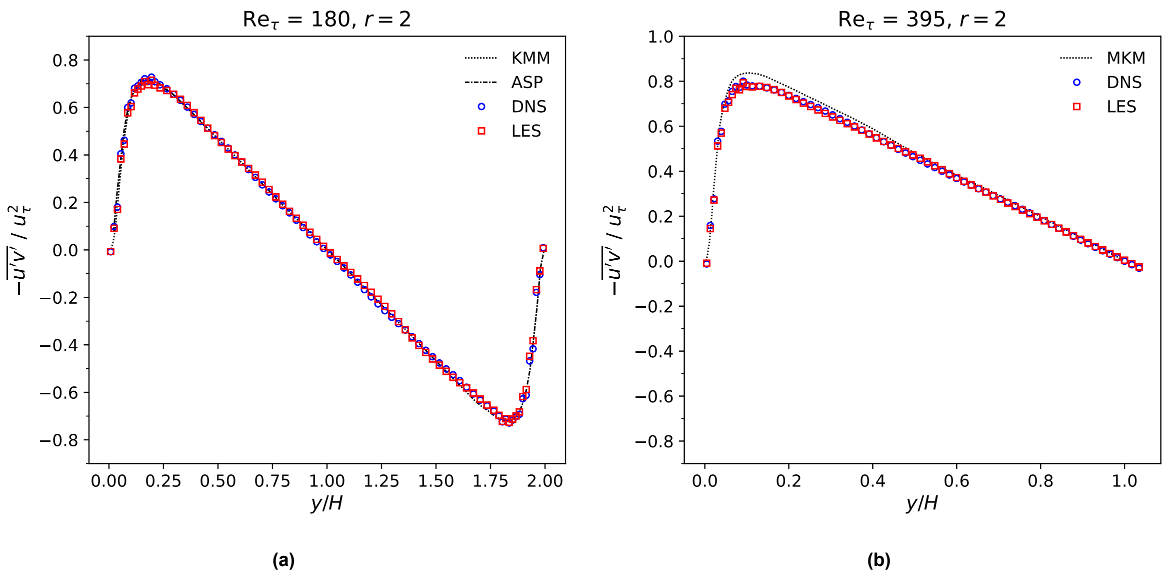


Figure 5.7: Reynolds stress against the wall distance y^+ for turbulent channel flow at (a) $Re_\tau = 180$, and (b) $Re_\tau = 395$. The grid has cubic cells of size $\Delta^+ = 2.8$ in the range of $y^+ = 0$–40 and $\Delta^+ = 5.6$ elsewhere. The results of Amati et al. [6] and Kim et al. [55] are also shown for comparison. Theoretical profiles are included for both the viscous sublayer and the log-layer.

Observed Performance

The highest amount of observed MLUPS is 419 and has been achieved in the unrefined $r = 1$ DNS simulation with $Re_\tau = 395$. This is an improvement of 20% compared to the highest performing simulations in the MSc theses of [105] and [119], who also adopted a CUDA implementation using the Numba library in Python.

The $Re_\tau = 395$ simulations performed slightly better than the $Re_\tau = 180$ simulations, which can be attributed to the different 'blocks per grid' (bpg), and 'threads per block' (tpb) settings. These settings are based on the different grid sizes, as mentioned in Sec. 3.8, and impact the performance of simulations. This is further discussed in Sec. 5.5.3.

Furthermore, the $r = 2$ DNS simulations showed a performance of less than 50%, compared to the $r = 1$ simulations. Similarly, the $r = 4$ simulation performs 26% less than the $r = 2$ simulation. However, the total simulation times were still significantly smaller for the $r = 2$ and $r = 4$ simulations, compared to the $r = 1$ simulation. This is indicated by the striped bars in Fig. 5.9, representing the 'effective MLUPS', which assumes a homogeneous fine grid in Eq. 5.1. The effective MLUPS for $r = 2$ is given by

$$\text{MLUPS}_\text{eff} = \frac{N_{\text{cells},r=1} N_{T,\text{max},r=2}}{T_{r=2}} \cdot 10^{-6} \tag{5.3}$$

This means that the number of cells N_cells of the $r = 1$ grid is inserted in the MLUPS calculation of the $r = 2$ simulation. In Fig. 5.9, it can be observed that the effective MLUPS of the $r = 2$ simulations is 40% higher for the $Re_\tau = 180$ simulation and 240% higher for the $Re_\tau = 180$ simulation, compared to their regular MLUPS. These percentages also give the corresponding reductions in simulation time in comparison with the unrefined simulations.

The performance reduction of the $r = 2$ simulations is a result of the non-homogeneous grid in the local grid refinement implementation. In the coarse layer, the number of cells is halved in the stream- and span-wise directions, compared to the number of cells along these directions in the fine layers. This means that there is a reduced occupancy in the coarse grid. Nonetheless, the overall simulation time of the $r = 2$ simulation was reduced. This results from a significantly reduced amount of required memory to store the large distribution function arrays. These arrays are stored in global memory and the time it takes for threads to read and write values from/to them is the bottleneck of simulation speed. Smaller arrays allow faster evaluations per cell and therefore lead to an increase in effective MLUPS.

The $r = 4$ simulation does not show much increase in performance, compared to the $r = 2$ simulations. This is because the bottleneck in simulation time lies in the fine layers. A reduction of cells in the

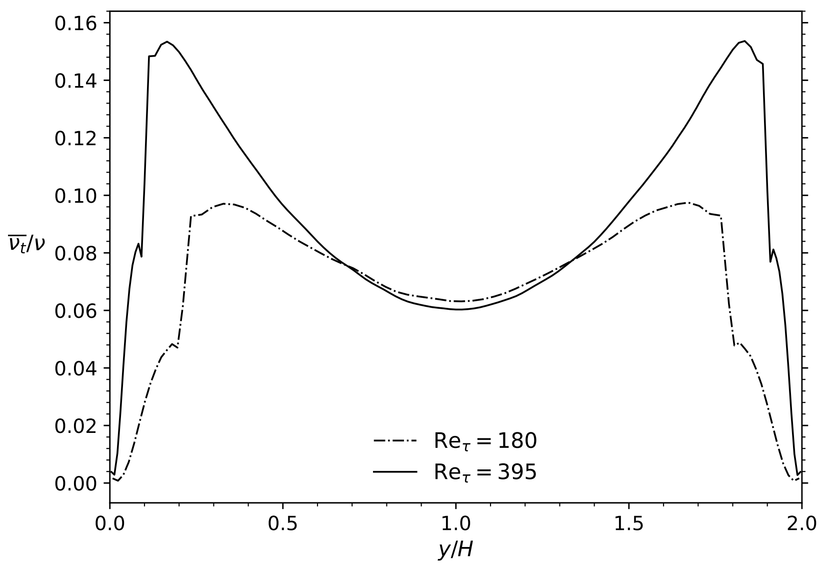


Figure 5.8: Mean eddy-viscosity normalized by the molecular viscosity in fine units against wall-normal position y/H. A grid refinement $r = 2$ has been applied. The eddy-viscosity was calculated using the WALE model and results for $Re_\tau = 180$ and $Re_\tau = 395$ are given. Due to the larger coarse grid spacing, the eddy-viscosity is larger in the coarse grid ($0.22 < y/H < 1.78$ for $Re_\tau = 180$ and $0.1 < y/H < 1.9$ for $Re_\tau = 395$).

coarse layer does not have much influence on the total amount of cells. Simulation time is therefore not significantly reduced.

Gap with GPU Potential

As mentioned before, a maximum of 419 MLUPS has been achieved in the performed simulations. This is only 19% of the potential 2160 MLUPS that was identified in the previous section. The lack of shared memory usage in the stream and collision stages is the primary difference between the current GPU implementation and more efficient simulations such as [102, 122]. This improvement significantly speeds up access to intrinsic and neighboring distributions inside a block. These studies also combined the collision and propagation steps into one kernel.

Although more sophisticated algorithms were tried, race conditions could not be prevented using the current implementation of the Numba library in Python. Numba does not yet contain all CUDA-functionalities that C-oriented languages have [83]. It might be more convenient to implement more sophisticated kernels using these programming languages. An example of missing functionality is dynamic parallelism, which is a functionality that allows the kernel to call other kernels [1], or cooperative groups, which allow synchronization of grouped threads across different blocks [84].

5.5.3. Speed-up Attempts

Throughout the coarse of this research, several attempts have been made to increase the speed of the present GPU-based algorithms. These will be summarized below.

- Determining optimal blocks per grid (bpg) and threads per block (tpb) settings. This was found to have a large impact on the performance of separate kernels. Furthermore, one choice of configuration could be optimal for one kernel, while it performed poorly for another kernel. It is highly recommended to configure the optimal bpg and tpb independently for each separate kernel. This could ideally be automated at the start of a simulation. Note also that the optimal bpg and tpb are not only kernel-dependent but also GPU-dependent.
- For the locally refined simulations, it was attempted to assign unused threads in the stream-wise direction to a new row of cells during the collision and propagation steps. This led to a 50% reduction of blocks along the wall-normal direction in the coarse layer, but only a slight increase in performance was observed ($\sim$ 5%).

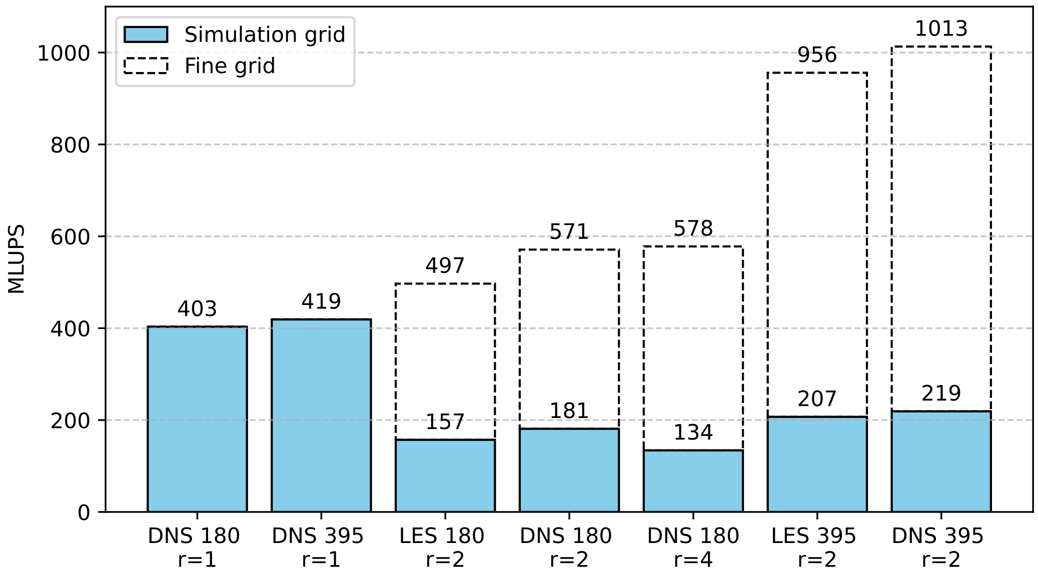


Figure 5.9: Million lattice updates per second (MLUPS) achieved in the turbulent flow simulations. Results for the actual used grids are indicated in blue for both $r = 1$ and $r = 2$ simulations. For $r = 2$ simulations, the effective MLUPS is also indicated with a dotted border, assuming a homogeneous fine grid. The effective MLUPS is calculated using Eq. 5.3

- Store 4D distribution arrays as 1D flattened arrays: This had a significant impact and increased computation speed by a factor $\sim$2. Index functions were used to convert 1D indices to 3D and vice versa. This methodology was originally recommended by [105] and [119].
- Pre-load variables on GPU: In speed tests with separate kernels this was observed to increase speed significantly. Impact on the speed of a full simulation has not been tested.
- Reduce unnecessary variable declarations in the kernel to minimize register spilling (see Sec. 3.8). This led to no apparent improvements. However, the amount of unnecessary variable declarations was already quite low, so the amount of register spilling might not have been significant.
- Use of constant memory. This was found to not affect the Numba implementation. As was explained Sec. 3.8 this is specific to Numba, as global constants in the global memory are automatically transferred to the constant memory. Other programming languages might benefit from the constant memory.

5.6. Conclusion

It has been shown that both DNS and WALE-LES implementations could accurately describe the dynamics in turbulent channel flows with $Re_\tau = 180$ and $Re_\tau = 395$. This conclusion follows from a comparison of the obtained turbulent statistics with previous studies. The LES simulations yield an improved approximation of the mean stream-wise velocity, a slight overshoot in the RMS fluctuation, and no change in the Reynolds stress.

The computational performance in lattice updates per second was shown to be superior for high Reynolds number simulations without refinement. Although the use of local grid refinement led to an efficiency reduction, the effective simulation time was reduced. The achieved performance (MLUPS) surpassed recent simulations that also adopted the Numba framework. However, there is still room for improvement, particularly through the adoption of more sophisticated algorithms, for which C-based languages offer greater potential.

6
Heat Transfer and Solidification Results

In this chapter, an extension will be made to the modeling of phase change in turbulent channel flows, now that the LES-FMLBM application has been proven to be accurate. The adopted computational setup is introduced in Sec. 6.1. Then, a new method is proposed in Sec. 6.2 that stabilizes the DDF-LBM in turbulent channel flows. In sec. 6.3, the obtained thermal statistics will be discussed and compared with previous data. In sec. 6.4, an extension will be made to the modeling of a developed ice layer in steady-state flow. The results will be compared with analytical solutions and Nusselt correlations from experimental data. Results for non-eutectic fluids will also be presented.

6.1. Computational Setup

The case that is being simulated in this chapter is turbulent flow through flat parallel plates with a constant temperature difference. The upper wall temperature is set at $T = T_U$ and the lower wall temperature at $T = T_L$. The physical setup is schematically shown in Fig. 6.1. In Sec. 6.2, T_L is assumed to be above the freezing point of the fluid, while in Sec. 6.4, it is assumed to be below the freezing point. An ABB scheme is used as the thermodynamic boundary condition at the walls, and periodicity of the thermal field is assumed in the stream- and span-wise directions. This has also been described in Sec. 3.4.

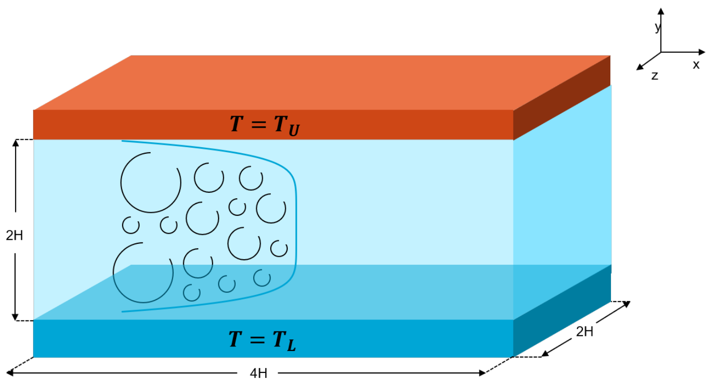


Figure 6.1: Schematic overview of turbulent channel flow through parallel plates with fixed temperatures T_U and T_L for the plates at $y = 2H$ and $y = L$, respectively.

Simulations have been performed using the DDF-FMLBM WALE-LES model, again adopting a WALE model constant $C_w = 0.5$. The turbulent Prandtl number was set at $Pr_\tau = 0.9$, following the reasoning in Sec. 3.5.4. The initial temperature field is set at a constant value T_{init}. The Reynolds number is set at $Re_\tau = 180$ and a Prandtl number of $Pr = 0.71$ is chosen, which is a canonical case

Table 6.1: Input parameters to the performed thermal simulations in physical and LB units. LB units of domain size, viscosity, density, and temperature have been chosen. The conversion factors for other quantities can be obtained using dimensional analysis.

Description	Quantity	Physical units	LB units
Size of the domain	$N_x \times N_y \times N_z$	$10 \times 5 \times 5$ cm	$256 \times 128 \times 128$ ls^3
Viscosity	ν	$1.7 \cdot 10^{-6}$ m^2/s	$2.37 \cdot 10^{-3}$ ls^3/lt
Body force	g	5.99 N/m^3	$6.94 \cdot 10^{-7}$ ls/lt^2
Density	ρ	1000 kg/m^3	1 lm/ls^3
Thermal diffusivity (s/l)	$\alpha_{s/l}$	$9.58/2.39 \cdot 10^{-6}$ m^2/s	$13.4/3.34 \cdot 10^{-3}$ ls^2/lt
Specific heat capacity (s/l)	$C_{p,s/l}$	$2.1/4.2 \cdot 10^3$ J/kg K	$623/1246$ lJ/lm lK
Latent heat	L	$3.34 \cdot 10^5$ J/kg	$9.91 \cdot 10^4$ lJ/lm
Liquidus temperature	T_l	273.15 K	273.15 lK
Solidus temperature	T_s	273.15 K	273.15 lK

that has been used in many previously performed simulations [53, 120, 90]. This allows for a good comparison of results. The refinement is set at $r = 1$, because the choice $r = 2$ leads to instabilities in the coarse layer, as will be explained in Sec. 6.2.

The domain size and kinematic input parameters remain unchanged compared to the $Re_\tau = 180$ situation in the previous chapter. The assumed thermal properties of the fluid were mainly based on water. An exception is the thermal diffusivity which was determined as $\alpha = \nu/Pr$, using $Pr = 0.71$ instead of $Pr \approx 10$ which is valid for water at $T \sim 280$ K [115]. The collection of input parameters is listed in Tab. 6.1, given both in physical and LB units. Note that the temperature conversion factor is chosen to be unitary, following the implementation by [10] and [119].

6.2. Stabilizing the Thermal Field

This section discusses the fluctuations and instabilities that were observed in thermal simulations using the enthalpy-based DDF-LBM approach, which was described in Sec. 3.2.2. Subsequently, a set of transformation rules is proposed that is shown to solve these issues. This is a novel approach that solves the same difficulties observed in a recent previous study of DDF-LBM in turbulent channel flows by Wortelboer [119]. Lastly, the maximum stable Prandtl number is discussed.

6.2.1. Thermal Fluctuations and Instability

This section describes the thermal fluctuations and instabilities that were encountered when the physical temperatures were used as inputs to the simulation. This means that the temperature conversion factor is $C_T = 1$.

Currently, our interest is to simulate a thermal flow without freezing. To this end, the top and bottom wall temperatures are chosen to be

$$T_L = 275 \text{ K}, \quad T_U = 280 \text{ K}, \tag{6.1}$$

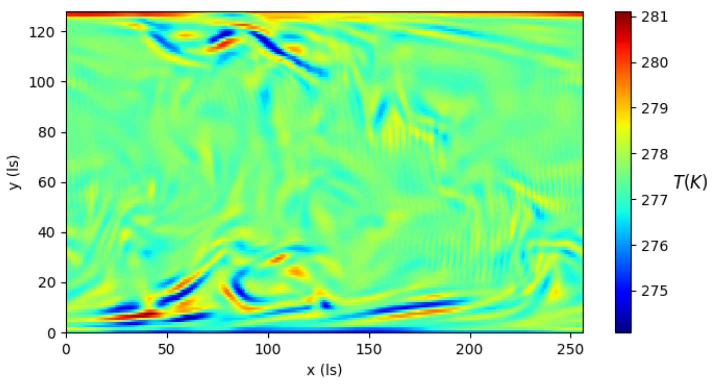
where T_L corresponds to the temperature at $y = 0$ and T_U corresponds to the temperature at $y = 2H$. The initial temperature in the channel is set to a constant value $T_{init} = 277.5$ K. Because there is no phase change involved yet, it suffices to define the enthalpy using the simple relation

$$h = C_{p,l} T. \tag{6.2}$$

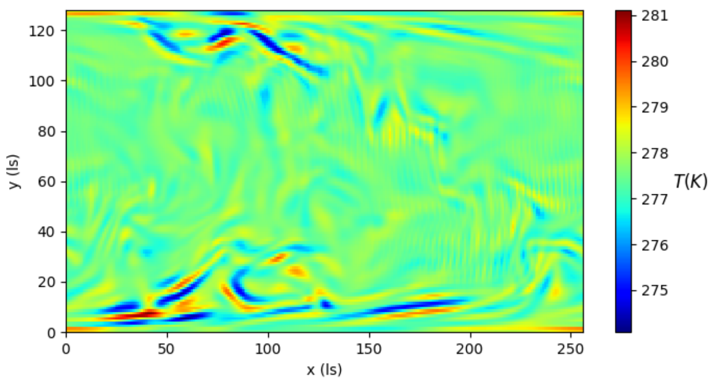
When performing a simulation with this set of parameters, and using DDF-FMLBM with a D3Q19 momentum scheme and a D3Q7 thermal scheme, instabilities were observed that led to a diverging temperature field after $\mathcal{O}(10^4)$ time steps. Also very early on in the simulation, significant fluctuations are observed. To illustrate this, a snapshot of $T(x, y, z = H)$ is taken at $t^+ = 0.11$, which is equivalent to 500 time steps. The result is shown in Fig. 6.2a. To better quantify the size of the fluctuations, the same simulation is also performed with a constant temperature

$$T_{\text{init}} = T_L = T_U = 277.5\text{K} \tag{6.3}$$

throughout the whole channel. The resulting profile is shown in 6.2b. It can be seen that the most signif-



(a)



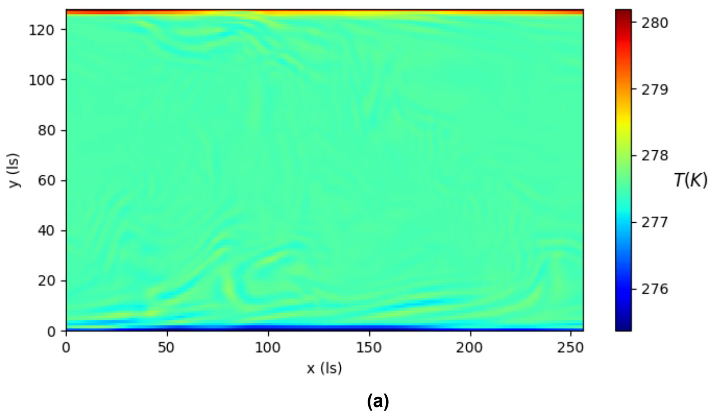
(b)

Figure 6.2: Instantaneous temperature profiles $T(x, y, z = H)$ using a D3Q7 thermal scheme with initial temperature $T_{in} = 277.5$ K and wall temperatures given by (a) $T_L = 275$ K and $T_U = 280$; (b) $T_L = T_U = 277.5$ K. Coordinates x and y are measured in lattice units, such that $H = 64$ [ls]. A snapshot is taken at $t^+ = 0.11$ to show the spatially fluctuating temperature early on in the simulation.

icant temperature fluctuations are around 3 K. This is an undesirable amount given that the fluctuation is of the same order of magnitude as the original temperature difference $(T_U - T_L)$.

In an attempt to stabilize the simulation, a thermal D3Q19 scheme was adopted as it is reported to preserve Galilean invariance as opposed to the D3Q7 scheme [130], which leads to increased stability. The collision step is still executed using an FMLB approach. With a larger velocity set, additional higher-order error terms are filtered out in the filter-matrix approach, as was mentioned in Sec. 3.2.2. A snapshot of the resulting profile at $t^+ = 0.11$ is shown in Fig. 6.3a. As was done in the D3Q7-D3Q19 approach, an additional simulation with a constant temperature $T_{in} = T_L = T_U = 277.5$ was performed to be able to quantify the fluctuations (see Fig. 6.3b). It can be seen that the fluctuations in the off-wall region have decreased to around 0.5 K, which is a factor 6 smaller than what was observed in the D3Q7 simulation.

This result is surprising given the successful prior studies on heat transport in turbulent channel flows using the lattice Boltzmann framework, such as the MRT-LES of Ren et al. [90], which used a comparable grid spacing of $\Delta^+ = 3$. However, the difference with the current work is that Ren et al. conveniently defined their flat plate temperatures as $T_L = 0$ lK and $T_U = 1$ lK. Due to their use of a temperature updating scheme, their distribution function remains of the order $\mathcal{O}(1)$. In contrast, the current study adopts an enthalpy updating scheme, resulting in a distribution function of the order $\mathcal{O}(C_pT)$, which, with the current parameters, leads to the order $O(10^5)$. Given that all other parameters are comparable to the parameters of Ren et al., the different orders of magnitude of the distribution functions may have contributed to the unexpected instabilities and fluctuations. To test whether this is the case, the enthalpy was scaled such that



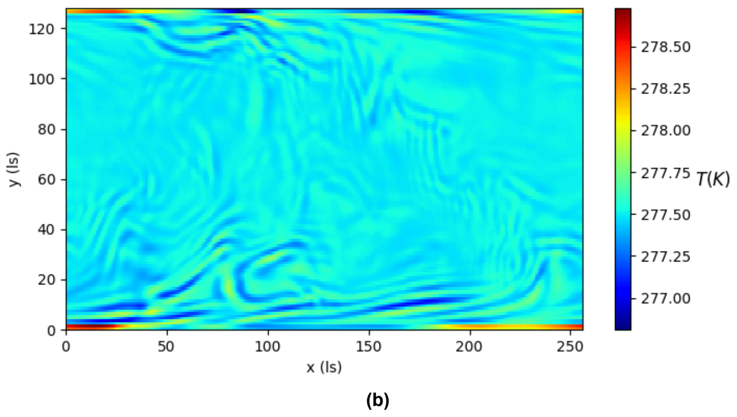


Figure 6.3: Instantaneous temperature profiles $T(x, y, z = H)$ using a D3Q19 thermal scheme with initial temperature $T_{in} = 277.5$ K and wall temperatures given by (a) $T_L = 275$ K and $T_U = 280$; (b) $T_L = T_U = 277.5$ K. Coordinates x and y are measured in lattice units, such that $H = 64$ [ls]. A snapshot is taken at $t^+ = 0.11$ to show the spatially fluctuating temperature early on in the simulation.

$$h_U = 1 \text{ lJ/lm}, \quad h_L = 0.9821 \text{ lJ/lm},$$

leading to an order of magnitude of the distribution function that is similar to the order in the simulations of Ren et al. The corresponding temperatures are now

$$T_U = 7.885 \cdot 10^{-4} \text{ lJ/lm}, \quad T_L = 8.028 \cdot 10^{-4} \text{ lJ/lm}.$$

Interestingly, this scaling led to the same temperature profile and fluctuations as in the unscaled case of Fig. 6.3a. This means that the fluctuation scales directly with the local value of the temperature or enthalpy. Hence, to suppress these fluctuations, it is logical to increase the relative difference between the minimum and maximum enthalpies, denoted as h_{min} and h_{max}, respectively. Since fluctuations are proportional to h, increasing the gap between h_{max} and h_{min} should result in less significant fluctuations relative to $(h_{max} - h_{min})$. We term this a "stretched" enthalpy spectrum, indicating a broader distribution of enthalpy values. This concept resembles contrast stretching in imaging, where intensities are expanded across a wider range to increase contrast. The idea is visualized in Fig. 6.4.

Several simulations with stretched enthalpy spectra were performed. It was found that the enthalpy stretching procedure only gave a stable result when the maximum enthalpy was not too large. For example, the combination

$$h_{max} = h_U = \mathcal{O}(10^3), \quad h_{min} = h_L = 0$$

led to an unstable field, while

$$h_{max} = h_U = \mathcal{O}(10^2), \quad h_{min} = h_L = 0$$

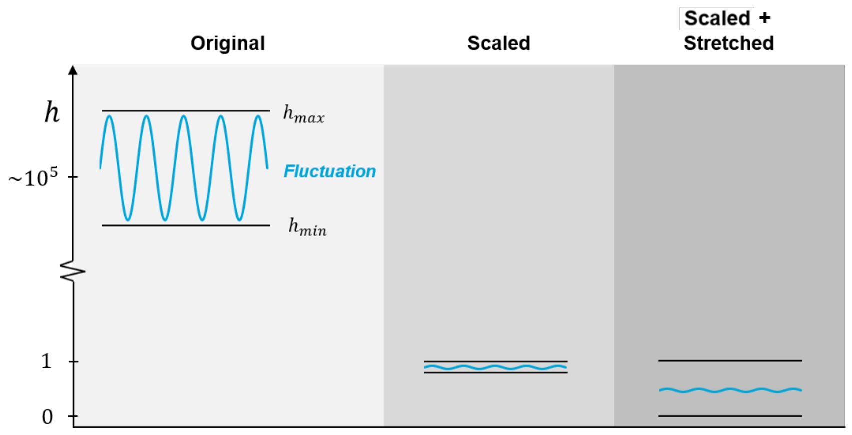


Figure 6.4: Influence of scaling and stretching on the sensible enthalpy fluctuation. The fluctuation is dependent on the absolute value of h; it can be suppressed by increasing the relative difference between h_{min} and h_{max}

was stable. Phase change remains disregarded. These findings lead to the formulation of two conditions that need to be met for a stable thermal simulation. These are

1. **The maximum absolute enthalpy in the domain must be small**

$$h_{max} < \mathcal{O}(10^2) \tag{6.4}$$

2. **The maximum relative enthalpy difference in the domain must be large**

$$(h_{max} - h_{min})/h_{max} \approx 1 \tag{6.5}$$

6.2.2. Transformation Rules

To obtain a stable simulation that is still physically meaningful, transformation rules for temperature, enthalpy, and latent heat are proposed that ensure correspondence with conditions 6.4-6.5. The transformation rules for enthalpy and temperature are

$$\widehat{h} = (h - h_{min})\frac{\widehat{h}_{max} - \widehat{h}_{min}}{h_{max} - h_{min}} + \widehat{h}_{min}, \tag{6.6}$$

$$\widehat{T} = (T - T_{min})\frac{\widehat{h}_{max} - \widehat{h}_{min}}{h_{max} - h_{min}} + \widehat{T}_{min}, \tag{6.7}$$

where a hat indicates a transformed variable and no tilde indicates a physical variable. Variables $\widehat{h}_{min}$ and $\widehat{h}_{max}$ are the desired transformed minimum and maximum sensible enthalpy, which correspond to wall temperatures T_L and T_U. These can be chosen in a way that conditions 6.4-6.5 are satisfied (e.g., $\widehat{h}_{min} = 0$ and $\widehat{h}_{max} = 1$). The values h_{min} and h_{max} denote the physical minimum and maximum sensible enthalpies. The transformed minimum temperature $\widehat{T}_{min}$ that appears in Eq. 6.7 can be determined as

$$\widehat{T}_{min} = \begin{cases} \frac{\widehat{h}_{min}}{C_{p,s}} & T_{min} < T_s \\ \widehat{T}_s + \frac{\widehat{h}_{min} - \widehat{h}_s}{C_{p,m}} & T_s \leq T_{min} \leq T_l \\ \widehat{T}_l + \frac{\widehat{h}_{min} - \widehat{h}_l}{C_{p,l}} & T_{min} > T_l, \end{cases} \tag{6.8}$$

where the transformed solidus temperature $\widehat{T}_s$ and liquidus temperature $\widehat{T}_l$ are given by

$$\widehat{T}_s = \frac{\widehat{h}_s}{C_{p,s}}, \tag{6.9}$$

$$\widehat{T}_l = \widehat{T}_s + \frac{\widehat{h}_l - \widehat{h}_s}{C_{p,m}}. \tag{6.10}$$

Because the goal of this research is to simulate phase change, it is also necessary to define a transformed latent heat. The latent heat expresses a *difference* in total enthalpy and must therefore be scaled directly without the use of an offset (as opposed to Eqs. 6.6-6.7):

$$\widehat{L} = L\,\frac{\widehat{h}_{max} - \widehat{h}_{min}}{h_{max} - h_{min}}. \tag{6.11}$$

The enthalpy, temperature, and latent heat are now transformed according to Eqs. 6.6 - 6.11, leading to the set of transformed simulation variables listed in Tab. 6.2. At the end of the simulation, the enthalpy and temperature are transformed back using inverse relations that can easily be derived from Eqs. 6.6 - 6.11.

This procedure gave the desired result as the simulation ran without instabilities or significant fluctuation. A snapshot of the temperature profile after $\sim 1.6M$ time steps is shown in Fig. 6.5. Qualitatively, clear turbulent structures can be observed that consist of equithermal packets of the fluid, and there is no significant fluctuation.

The accuracy of the obtained thermal statistics is assessed in the next section. The observed fluctuations in Figs. 6.2-6.3 are found to be an inherent effect of the LBM streaming algorithm. A detailed explanation for this can be found in App. A.

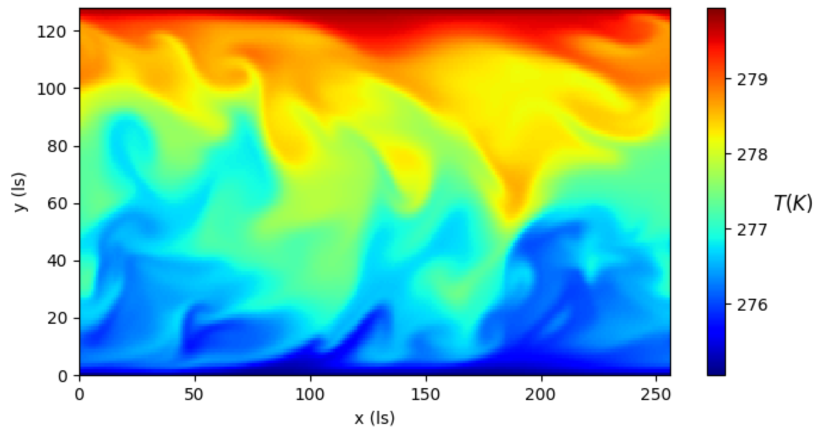


Figure 6.5: Instantaneous temperature profile $T(x, y, z = H)$ after $1.6 \cdot 10^6$ time steps, using the proposed transformations of the enthalpy field. The end-profile is transformed back with inverse relations.

Table 6.2: Thermal simulation variables given in physical, LB, and transformed units. Transformations have been performed using Eqs. 6.6 - 6.11. Note that the latent heat plays no role in the simulation since h_s and h_l are below the minimum enthalpy h_L. For convenience, $C_{p,l} = C_{p,s} = 4.2 \cdot 10^3 J/kgK$ has been chosen, which does not impact the thermodynamics as there is no solid layer.

Description	Quantity	Physical units	LB units	Transformed units
Lower wall temperature	T_L	275 K	275 lK	0
Upper wall temperature	T_U	280 K	280 lK	8.03 $\cdot 10^{-4}$
Solidus temperature	T_s	273.15 K	273.15 lK	-2.97 $\cdot 10^{-4}$
Liquidus temperature	T_l	273.15 K	273.15 lK	-2.97 $\cdot 10^{-4}$
Lower wall sensible enthalpy	h_L	1.16 $\cdot 10^6$ J/kg	3.43 $\cdot 10^5$ lJ/lm	0
Upper wall sensible enthalpy	h_U	1.18 $\cdot 10^6$ J/kg	3.49 $\cdot 10^5$ lJ/lm	1
Solidus sensible enthalpy	h_s	1.15 $\cdot 10^6$ J/kg	3.40 $\cdot 10^5$ lJ/lm	-0.37
Liquidus sensible enthalpy	h_l	1.15 $\cdot 10^6$ J/kg	3.40 $\cdot 10^5$ lJ/lm	-0.37
Latent heat	L	3.34 $\cdot 10^5$ J/kg	9.91 $\cdot 10^4$ lJ/lm	4.72

6.2.3. Maximum Prandtl number

It was found that the maximum stable Prandtl number had a value of $Pr \approx 0.8$. At higher Prandtl numbers, thermal fluctuations occurred in the computational domain, which led to instability.

Fluctuations are commonly encountered in flows with $Pr \gtrsim 1$ and are referred to as ”wiggles” [42]. They are numerical dispersion errors that arise due to a mismatch between the time scales for flow and thermal fields. This mismatch can lead to serious instability [25]. However, in the performed simulations, instabilities were also observed at $Pr \approx 1$, while at lower Prandtl numbers ($Pr < 0.8$) the simulations remained stable.

Because there is in principle no mismatch between viscous and thermal time scales at $Pr = 1$, the observed instabilities are likely related to the size of thermal diffusivity α. When α becomes too low, instabilities are observed. With $\nu = 0.00237$ ls^2/lt and a maximum Prandtl number $Pr = 0.8$, this corresponds to a minimum value $\alpha_{min} = 0.0030$ ls^2/lt at which the simulation remains stable. This hypothesis is substantiated by the following observations:

- No instabilities were observed for high-Prandtl laminar flows, which could easily reach $Pr \sim \mathcal{O}(10)$. The viscosity of simulated laminar flows was $\nu = 0.17$, which yields a thermal diffusivity that remains well above the minimum stable value for such Prandtl numbers.
- When a local grid refinement with $r = 2$ was adopted to the turbulent thermal simulations, the maximum stable Prandtl number was halved. The thermal diffusivity in the coarse layer is scaled by a factor $1/2$, so the same minimum stable value $\alpha_{min} = 0.0030$ ls^2/lt remains valid in the coarse layer.

The vast majority of turbulent thermal LB studies are also restricted to $Pr = 0.71$ or lower, but clear stability regions are mostly not mentioned. There are several studies such as [90] and [42], where new methods were developed to simulate high Prandtl numbers $Pr \gg 1$. However, such methods were outside the scope of this research. A more detailed description of these studies is given in Sec. 6.5.1.

To stay within the observed stability limits, the remainder of this thesis considers only $Pr = 0.71$ and adopts no local grid refinement ($r = 1$) for the aforementioned reasons.

6.3. Thermal Statistics

The transformation rules presented in the previous section have been applied to DNS and LES simulations of turbulent flows with a D3Q19-D3Q19 scheme for momentum and temperature. The initial field was set at $T_{\text{init}} = (T_U + T_L)/2$ and the simulations were run until the maximum allowed simulation time of 24h was reached on Delftblue. For the DNS, this resulted in a total of 1.95M time steps, and for the LES, this number was 1.7M. Saving started after 1M and 1.2M timesteps, respectively, with a total of 80 snapshots for both simulations. This offset was chosen to ensure a developed temperature field.

The thermal turbulent statistics will now be compared with previous studies that simulated a similar case. The thermal turbulent statistics of interest are the mean temperature, the RMS temperature fluctuation, and the turbulent heat flux. The first study that is used for comparison is the DNS by Kawamura et al. (KAS) [53], who applied a spectral method and a very fine spacing $\Delta y^+ = 0.40$ near the walls that gradually increased towards the center of the channel where $\Delta y^+ = 11.5$. The second study is the MRT-LBM-LES by Ren et al. (RSH) [90], who applied a Vreman model with a dynamically computed model coefficient. RSH adopted a D3Q19 momentum scheme and a D3Q7 thermal scheme, using a temperature-based DDF approach with $T_L = 0$ and $T_U = 1$. They adopted a spacing $\Delta y^+ = 1.5$ for the first layer of cells at the walls and a spacing $\Delta y^+ = 3$ elsewhere. The specifics of these studies are summarized in Tab. 6.3.

Table 6.3: Specifications of previously conducted thermal turbulent channel flow simulations that are comparable to simulations in the current work. For RSH, the amount of nodes in the y-direction

Author	Abbrev.	Re_τ	Pr_t	(N_x, N_y, N_z)	$\Delta^+_{y,\text{min}}$	$\Delta^+_{y,\text{max}}$	Dynamics	SGS model
Kawamura et al.	KAS	180	0.71	(128, ?[1], 128)	0.40	11.5	Spectral	-
Ren et al.	RSH	180	0.71	(256, 122, 96)	1.5	3	MRT-LBM	Vreman

[1]Amount of nodes in y-direction of KAS is unknown, but the mesh dimensions (x^+, y^+, z^+) are ($1152 \times 360 \times 576$).

Mean Temperature

The mean temperature profile, RMS temperature fluctuation, and turbulent heat flux of the LES and DNS are shown in Figs. 6.6a-6.6c. The mean temperature profiles in Fig. 6.6a follow the data of KAS and RSH very precisely. A steep slope is present near the walls, which is caused by the dominant diffusive heat transfer in the viscous sublayer, and a more shallow profile is present towards the center of the channel, which is caused by the dominant convective heat transfer in the bulk.

Temperature Fluctuation

The RMS temperature fluctuations are shown in Fig. 6.6b. There are three extrema of the temperature fluctuation: two near the walls and one at the center of the channel. Debusschere and Rutland [22] analyzed temperature fluctuations in similar plane channel flow and gave a reason for the extremum at the center. They observed that fluid from the walls travels to the centerline, forming packets of hot or cold fluid. These packets travel along with the mean flow, causing large temperature fluctuations along the centerline. This mechanism is also visible in the temperature snapshot of Fig. 6.5. The other two extrema, near the walls, follow the shape of the velocity fluctuation and result from a combination of high turbulent intensity and a strong temperature gradient near the wall [120].

When comparing the present results with the data of RSH and KAS, a slight under-prediction of is observed at the center of the channel. Compared to the data of KAS, this under-prediction is 4.5% for the DNS and 2.5% for the LES. In the MRT-LES by Wu et al. [120], who used a dynamic Smagorinsky model, a 3.6% under-prediction of the temperature fluctuation was observed, compared to KAS. The deviation in the current work is therefore assumed to be within an acceptable margin. Because Wu et al. use a slightly different measure for their temperature fluctuation, their results have not been included in the figure.

The LES shows a slightly higher estimation of the RMS temperature fluctuation in the center of the channel than the DNS. It is therefore in closer agreement with the data of KAS and RSH. At the local minima, a slight asymmetry is observed in the LES profile. This is a result of the shorter simulation time of the LES, which resulted from restrictions on the allowed computing time on the Delftblue supercomputer, which is 24 hours. As mentioned, the LES needed 24 hours for 1.7M time steps and the DNS needed 24 hours for 1.95M time steps. Saving of data started in the last 40% of the simulations to ensure a developed profile. However, in the future, it is recommended to use developed profiles as input to the simulation. Longer effective simulation times can then be achieved.

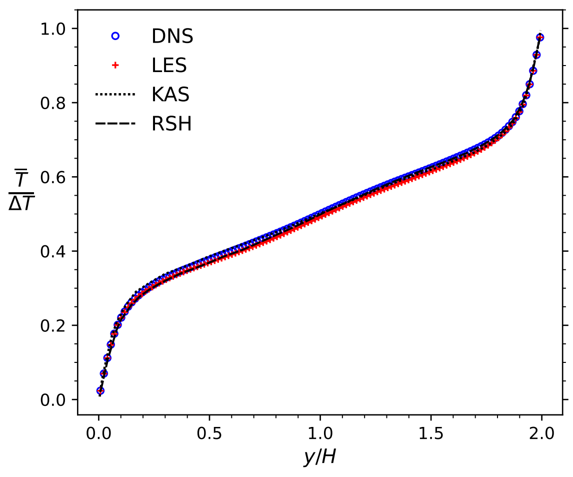
For comparison, RSH simulated for an equivalent 2.4M time steps and also used the last 40% for saving data. This means they only simulated $\sim$1M time steps for extracting turbulent statistics. In the current simulation, this is achievable within $\sim$14 hours if an initial turbulent thermal field is used.

Turbulent Heat Flux

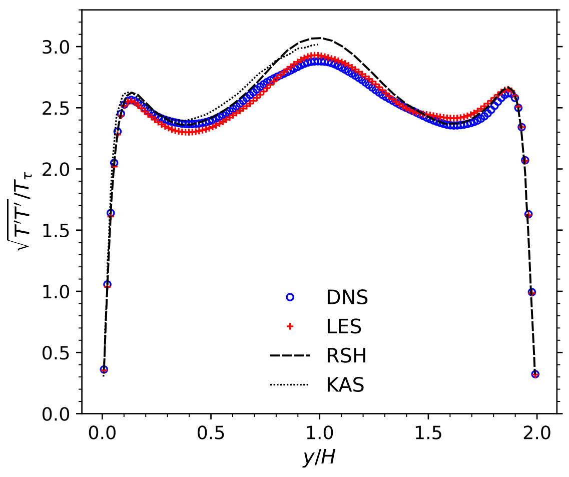
The results in Fig. 6.6c demonstrate that the turbulent heat flux is in good agreement with the data of RSH and KAS. Note that the data of KAS is slightly asymmetric, resulting in a small discrepancy near the minimum. The extrema follow the peaks of the streamwise velocity fluctuation and the anti-symmetric profile results from the shape of the mean temperature [53].

6.4. Steady-State Freezing

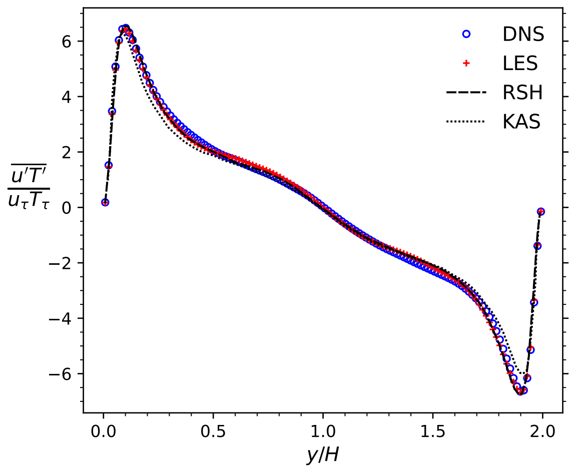
This section presents and discusses results that were obtained in the freezing simulations of turbulent channel flow. This is done without local grid refinement to stay within stability limits, as mentioned in Sec. 6.2.3. The WALE-LES model was found to give accurate results in the previous section and has been applied in the remainder of this work. The difference with the previous situation is that the lower wall temperature is now set below the freezing point. The no-slip condition on the liquid-solid interface is imposed using the immersed boundary method that was described in Sec. 3.3. First, analytical expressions are derived in Sec. 6.4.1 for the steady-state ice thickness in a channel with a fixed temperature difference, which will be used for benchmarking later on. Second, a revised version of the immersed boundary method is discussed in 6.4.2 which was necessary to obtain stable results. Then, the steady-state ice thickness of a zero-velocity system will be presented and compared with the analytical solution in Sec. 6.4.3. Then, results will be discussed for turbulent flows of eutectic fluid and compared with analytical solutions using Nusselt correlations. Lastly, the same is done for turbulent flows of non-eutectic fluids.



(a)



(b)



(c)

Figure 6.6: Thermal turbulent statistics of the performed DNS (no SGS model) and LES (WALE model) simulations. The shown statistics are (a) mean temperature normalized by the temperature difference $\Delta T = T_U - T_L$, (b) RMS temperature fluctuation normalized by the friction temperature, (c) turbulent heat flux normalized by the friction temperature and wall shear velocity.

The thermal diffusivities, specific heats, solidus and liquidus temperatures, and latent heat are the same as listed in Tab. 6.1. An exception is encountered in the non-eutectic case, where the liquidus temperature is set at $T_l = 275.15$ K, while the solidus temperature remains at $T_s = 273.15$ K. For all simulations, the upper wall temperature is set at $T_U = 300K$, the initial temperature is set at $T_{\text{init}} = 285$ K throughout the domain, and the lower wall temperature is varied to obtain different solid layer thicknesses. The initial Reynolds number and the Prandtl number remain at $Re_{\tau,0} = 180$ and $Pr = 0.71$, respectively. The enthalpy, temperature, and latent heat transformation rules that were introduced in Sec. 6.2 have been applied to ensure stability and reduce fluctuations.

6.4.1. Steady-state Ice Thickness

In this section, a convenient analytical expression of ice layer thickness in a steady-state channel flow is derived. This expression later serves as a benchmark solution for the performed simulations. The methodology is based on the balancing of heat fluxes at different locations in the channel.

Consider two flat plates with spacing L, upper plate temperature T_U, and lower plate temperature T_L, as sketched in Fig. 6.7. Because the situation is steady-state, heat flux ϕ_1'' through the upper wall

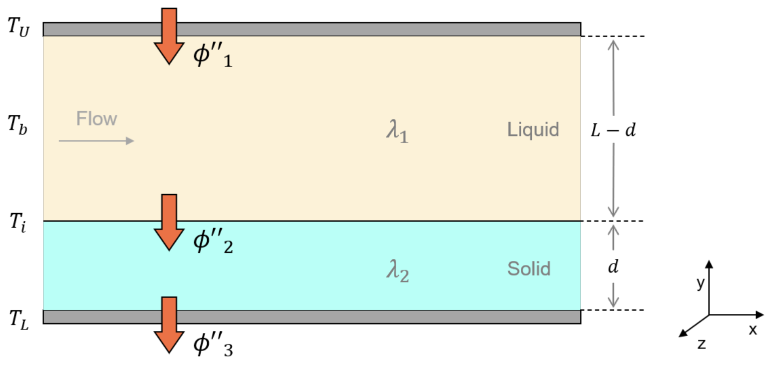


Figure 6.7: Developed ice layer with thickness d in a channel flow with steady-state heat transfer. The upper and lower walls are set at T_U and T_L, respectively, and have spacing L. The bulk temperature of the flow, temperature at the interface, liquid thermal conductivity, and solid thermal conductivity are denoted by T_b and T_i, λ_1 and λ_2, respectively. The heat flux through the upper wall, solid-liquid interface, and lower wall are denoted by ϕ_1'', ϕ_2'', and ϕ_3'', respectively.

is the same as heat flux ϕ_2'' through the solid-liquid interface, which is also the same as heat flux ϕ_3'' through the lower wall:

$$\phi_1'' = \phi_2'' = \phi_3''. \tag{6.12}$$

Each heat flux results from a local temperature gradient according to Fourier's law:

$$\phi'' = -\lambda \frac{dT}{dy}, \tag{6.13}$$

where λ is the heat conductivity of the medium. In the solid layer, there is a linear temperature profile from T_L to T_f, resulting in the following expression for ϕ_3'':

$$\phi_3'' = \frac{\lambda_2}{d}(T_i - T_L) \equiv h_3(T_i - T_L). \tag{6.14}$$

Here, h_3 is the heat transfer coefficient and in the solid layer it is defined as

$$h_3 = \frac{\lambda_2}{d}. \tag{6.15}$$

Similarly, fluxes ϕ_1'' and ϕ_2'' can be defined from the local temperature gradient in the fluid. However, the steady-state temperature profile in a turbulent channel flow is not linear from one boundary to the other. Instead, there is a small fluid layer near the wall where a mean temperature profile develops linearly. Beyond this layer, there exists a nearly constant bulk temperature T_b due to temperature mixing, which is a characteristic of turbulent flows. Fluxes ϕ_1'' and ϕ_2'' are therefore expressed in terms of the bulk temperature as

$$\phi_1'' = h_1(T_U - T_b), \quad \phi_2'' = h_2(T_b - T_i). \tag{6.16}$$

The corresponding heat transfer coefficients now also contain information about the flow and can be expressed as

$$h_1 = \frac{\lambda_1}{L-d} Nu_1, \quad h_2 = \frac{\lambda_1}{L-d} Nu_2, \tag{6.17}$$

where Nu_1 and Nu_2 are the Nusselt numbers at the upper wall and the solid-liquid interface, respectively. The Nusselt number is a non-dimensional quantity that gives the ratio between total and diffusive heat transfer [52]. Thus, it represents the impact of the flow on the heat flux. It is generally a function of the Prandtl number Pr and Reynolds number Re, and many experiments have been performed to determine corresponding relationships in different flow scenarios.

By setting $\phi_1'' = \phi_2''$ in Eq. 6.16, it is possible to express T_b in terms of T_L and T_h, yielding

$$T_b = \frac{h_1 T_U + h_2 T_i}{h_1 + h_2}. \tag{6.18}$$

Now, after filling the above expression in to the definition of ϕ_1'' in Eq. 6.16, we obtain

$$\begin{aligned} \phi_1'' &= h_1 \left(T_U - \frac{h_1 T_U + h_2 T_i}{h_1 + h_2} \right) \\ &= h_1 h_2 \frac{T_U - T_i}{h_1 + h_2}. \end{aligned} \tag{6.19}$$

Subsequently, one sets $\phi_1'' = \phi_3''$ using the above definition for ϕ_1'' and the definition in Eq. 6.14 for ϕ_3'',

$$h_1 h_2 \frac{T_U - T_i}{h_1 + h_2} = h_3 (T_i - T_L). \tag{6.20}$$

After inserting the definitions for h_1, h_2, and h_3 in Eqs. 6.17 and 6.15, the following equality is obtained:

$$\left(\frac{\lambda_1}{L-d} \right)^2 Nu_1 Nu_2 \frac{T_U - T_i}{\frac{\lambda_1}{L-d}(Nu_1 + Nu_2)} = \frac{\lambda_2}{d}(T_i - T_L). \tag{6.21}$$

Simplifying leads to

$$\frac{d}{L-d} \equiv A = \left(\frac{1}{Nu_1} + \frac{1}{Nu_2} \right) \frac{\lambda_2}{\lambda_1} \frac{T_i - T_L}{T_U - T_i}. \tag{6.22}$$

Now d can be isolated and expressed in terms of A, which yields

$$d = \frac{A}{1+A} L. \tag{6.23}$$

Eqs. 6.23 and 6.22 provide an expression of the steady-state ice layer thickness in channel flows with given Nusselt numbers. Note that the fraction of λ_1 and λ_2 can be expressed in terms of heat diffusivity α and specific heat C_p using $\lambda = \alpha \rho C_p$. For a uniform density, this leads to

$$\frac{\lambda_2}{\lambda_1} = \frac{(\alpha C_p)_2}{(\alpha C_p)_1}. \tag{6.24}$$

Zero Velocity

Another convenient flow case is a stationary fluid with zero velocity. Because there is no convection now, the thermodynamics are dictated by diffusive heat transfer only. This means that the fluid layer effectively becomes a solid and fluxes ϕ_1'' and ϕ_2'' can simply be described as

$$\phi_1'' = \phi_2'' = \frac{\lambda_1}{L-d}(T_U - T_i). \tag{6.25}$$

Following a similar approach as in the non-zero flow scenario, the factor A that appears in Eq. 6.23 now reduces to

$$A = \frac{\lambda_2}{\lambda_1} \frac{T_i - T_L}{T_U - T_i}. \tag{6.26}$$

6.4.2. Revised Phase Interface Treatment

Before continuing to discuss the performed simulations, a revision to the immersed boundary method will be introduced that was necessary to get meaningful results. In addition, a recommendation on the choice of ζ will be given, which is the parameter that occurs in the definition of B in Eq. 3.24.

It has been observed that the immersed boundary method as described in Sec. 3.3 introduced a negative stream-wise velocity in the fluid, right after the solid-liquid interface. Because this error only occurred in the presence of a phase interface, the most likely source of inaccuracy is the modified collision operator Ω^s which activates when $f_l < 1$ (see Eq. 3.23). From the definition of Ω^s in Eq. 3.27, a regular zero-velocity bounce-back is represented by the first two terms $f_j - f_i$, while an additional correction is introduced by the last two terms $f_i^{eq} - f_j^{eq}$. The latter is likely the source of any non-physical velocities since the regular bounce-back term inherently sets all velocities to zero.

This hypothesis led to the implementation of a simplified version of Ω^s, which is similar to a version used by [99], and omits the last two equilibrium distribution terms. The result is a regular bounce-back that inherently sets all velocities with $f_l = 0$ to zero:

$$\Omega_i^s = f_j\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i \Delta t}{2}, t - \frac{\Delta t}{2}\right) - f_i\left(\boldsymbol{x} - \frac{\boldsymbol{c}_i \Delta t}{2}, t - \frac{\Delta t}{2}\right). \tag{6.27}$$

This approach gave the desired result and removed the negative velocity and non-physical temperature fluctuation. This version will therefore be considered in the remainder of this work.

Furthermore, it was observed that the parameter ζ gave rise to instabilities when chosen at $\mathcal{O}(10^{-3})$ or smaller. It is recommended to use a value no smaller than 0.01, or ideally, apply the definition in Eq. 3.26 if possible. For eutectic flows, the choice of parameters was not observed to influence the resulting solutions, however. This is likely because the value ζ influences the mushy zone only and, for non-eutectic fluids, this zone spans only one row of cells.

6.4.3. Zero-velocity System

The first case that was simulated is a channel with zero velocity. It was initiated using a zero-velocity equilibrium distribution on all nodes. In addition, the body force was set to zero, so the fluid remained at zero velocity throughout the simulation. Due to the simplicity of this flow case, a convenient set of parameters was chosen that allowed quick convergence of the simulation. As such, the grid size was set at $(32 \times 96 \times 32)$ and the solid and liquid thermal diffusivities at $\alpha_s = 0.9390$ and $\alpha_l = 0.2347$, respectively. The simulation was run for $2 \cdot 10^5$ time steps, which allowed ample time for the solid layer to develop as the thickest solid layer was already at its final thickness after $1.37 \cdot 10^4$ time steps. Furthermore, the lower wall temperatures were varied across different simulations. The input temperatures are listed in Tab. 6.4, together with the corresponding temperature fractions $(T_i - T_L)/(T_U - T_i)$, where $T_i = T_s = T_l$ is the temperature at the solid-liquid interface. Temperatures T_U and T_L are 300K and 273.15K, respectively. The specific heat and latent heat are the same as listed in Tab. 6.1.

Table 6.4: Lower wall temperatures T_L used in zero-velocity freezing simulations. The upper wall and interface temperatures are set at $T_U = 300$K and $T_i = 273.15$K, respectively. The corresponding fraction $(T_i - T_L)/(T_U - T_i)$ is given as well. The transformed lower wall temperatures $\widetilde{T}_L$, interface temperatures $\widetilde{T}_i$, and upper wall temperatures $\widetilde{T}_H$ are also listed. This corresponds to $\widetilde{h}_{min} = 0$ and $\widetilde{h}_{max} = 1$.

Simulation no.	$T_L(K)$	$(T_i - T_L)/(T_U - T_i)$	$\widetilde{T}_L$ (lK)	$\widetilde{T}_i$ (10^{-6} lK)	$\widetilde{T}_U$ (10^{-6} lK)
1	272.83	0.0118	0	2.82	240
2	271.90	0.0464	0	10.8	244
3	270.78	0.0882	0	20.1	248
4	268.68	0.167	0	36.6	256
5	265.92	0.269	0	56.5	266
6	262.17	0.409	0	80.8	279
7	259.73	0.5	0	95.2	286

The analytical profile that describes the steady-state solid layer thickness in this situation was derived in Sec. 6.4.1 and is given by

$$d = \frac{A}{1+A} L, \tag{6.28}$$

where $L = 2H$ is the channel width and A is given by

$$A = \frac{(\alpha C_p)_s}{(\alpha C_p)_l} \cdot \frac{T_i - T_L}{T_U - T_i}. \tag{6.29}$$

The solid layer thicknesses d/L of the different simulations are shown in Fig. 6.8. The analytical solution is plotted as a reference. The results follow the shape of the analytical profile well. However, for all results, a constant under-prediction of $\Delta(d/L) = 0.010 - 0.014$ is observed, which corresponds to approximately $1 - 1.3$ cells. An error of one grid cell is not surprising, because the evaluation of the liquid fraction is based on the average temperature in one cell. However, this does not explain why only under-predictions are observed, and no over-predictions. This cannot be due to the adapted immersed boundary scheme in Eq. 6.27, since this scheme imposes a zero-velocity in the solid and the mushy zone, which is already zero throughout the domain. It is therefore not different from the normal immersed boundary scheme here.

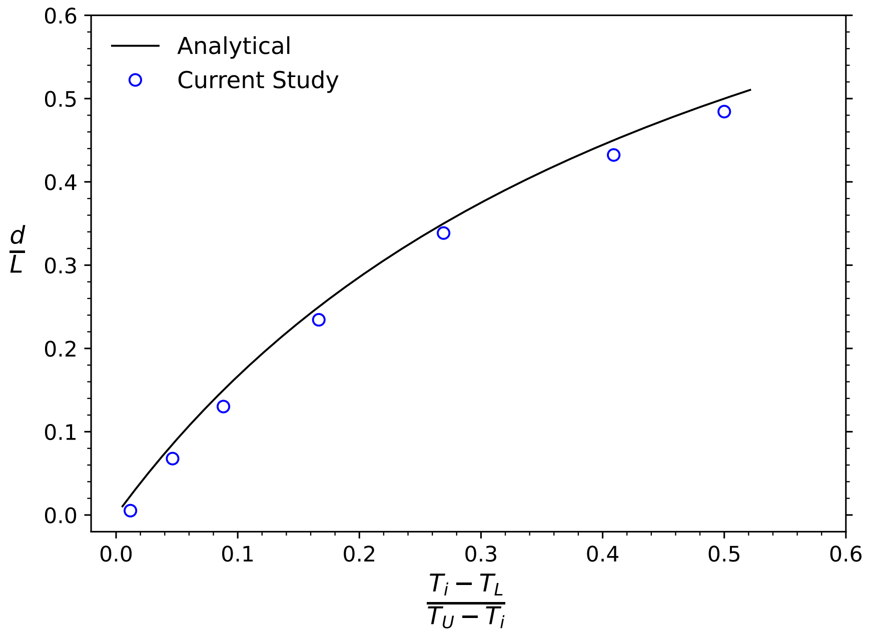


Figure 6.8: Steady state ice layer thickness normalized by the channel height L in a static channel for different values of the temperature fraction $(T_i - T_L)/(T_U - T_i)$

Despite the slight deviations, the simulation proved to be relatively accurate in determining the ice layer thickness. One can conclude that solidification in a static system with diffusive heat transfer is well described using the immersed boundary method in an FMLBM framework. The next section will expand the model with convective heat transfer by introducing a turbulent flow.

6.4.4. Turbulent Flow with Eutectic Fluid

The results of the simulations of freezing in turbulent channel flow will now be discussed. The simulation has been initialized with a $Re_\tau = 180$ turbulent flow field following the procedure explained in Sec. 3.7. No local grid refinement has been applied to remain within stability limits for the thermal conductivity (i.e., $Pr_{\text{max}} \approx 0.8$). The same viscosity, body force, and grid size have been applied as in the $r = 1$ simulations that were described in Ch. 5. The initial temperature in the channel was set at $T = 285$ K. The simulations operated continuously for approximately 17 hours, resulting in a cumulative total of 1.3 million time steps.

Again, the lower wall temperature T_L has been varied across different simulations and the interface temperature is still $T_i = T_l = T_s = 273.15$K. The input temperatures of the different simulations can be found in Tab. 6.5, together with corresponding temperature fractions $(T_i - T_L)/(T_U - T_i)$. The thermal diffusivities, specific heats, and latent heat are the same as listed in Tab. 6.1.

Table 6.5: Lower wall temperatures T_L used in turbulent freezing simulations. The upper wall and interface temperatures are set at $T_U = 300$K and $T_i = T_s = T_l = 273.15$K, respectively. The corresponding fraction $(T_i - T_L)/(T_U - T_i)$ is given as well. The transformed lower wall temperatures $\widetilde{T}_L$, interface temperatures $\widetilde{T}_i$, and upper wall temperatures $\widetilde{T}_H$ are also listed. This corresponds to $\widetilde{h}_{min} = 0$ and $\widetilde{h}_{max} = 1$.

Simulation no.	$T_L(K)$	$(T_i - T_L)/(T_U - T_i)$	$\widetilde{T}_L$ (IK)	$\widetilde{T}_i$ (10^{-6} IK)	$\widetilde{T}_U$ (10^{-6} IK)
1	271.21	0.0723	0	16.6	246
2	269.21	0.1469	0	16.6	246
3	267.14	0.2238	0	47.9	262
4	263.55	0.3575	0	72.2	274
5	258.98	0.5278	0	99.4	288
6	254.09	0.7098	0	125	300

Gnielinski's Nusselt Correlation

To properly evaluate the results obtained from the simulations in comparison with an analytical solution for a steady-state ice profile, it is necessary to select an appropriate Nusselt correlation. As described in Sec. 6.4.1, an expression for the Nusselt number must be inserted into Eqs. 6.22 and 6.23, to arrive at a useful expression for steady-state ice thickness d.

The Nusselt correlation that will be used is the Gnielinski correlation [39]. This is a more accurate correlation at lower Reynolds numbers ($Re_m < 10^4$) than more traditional relations such as the Dittus-Bolter or Colburn relations [100], and it is valid at $4 \cdot 10^3 \leq Re_m \leq 10^6$ and $0.5 \leq Pr \leq 200$. Assuming an infinitely long channel and a constant Prandtl number throughout the domain, the Nusselt number is expressed as

$$Nu = \frac{\frac{\xi}{8}(Re_m - 1000)Pr}{1 + 12.7\sqrt{\frac{\xi}{8}}\left(Pr^{2/3} - 1\right)}, \tag{6.30}$$

where ξ denotes the friction factor for smooth tubes and is calculated as

$$\xi = (1.82 \log Re_m - 1.64)^{-2}. \tag{6.31}$$

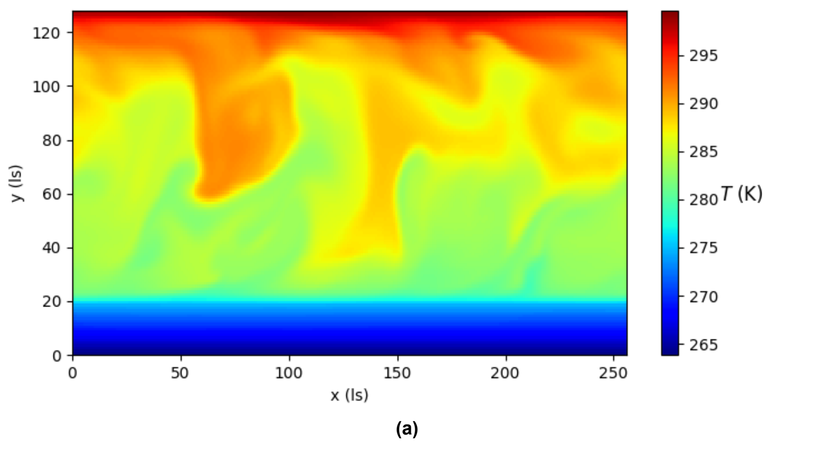
Note that the above Nusselt correlation depends only on the Prandtl and the bulk Reynolds number.

Because the bulk Reynolds number is dependent on the channel height and the bulk velocity, which are both influenced by the development of an ice layer, it must be adjusted accordingly before calculating Nu. The corrected channel height is simply given by the wet channel height $L - d$, and the bulk velocity is calculated at the end of the simulation, using its definition in Eq. 2.45.

Results

Figs. 6.9a and 6.9b show the final instantaneous temperature profile $T(x, y, z = N_z/2)$ and absolute velocity $|\boldsymbol{u}(x, y, z = Nz/2)|$ of simulation 4 (Tab. 6.5). Note that a smooth ice layer has developed with homogeneous temperature development and zero velocity. For all simulations, the final ice thickness d/L has been plotted against the corresponding temperature fractions $(T_i - T_L)/(T_U - T_i)$ in Fig. 6.10. The theoretical ice thickness, given by the Gnielinski Nusselt correlation, is plotted for the same temperature fractions. They have been calculated using the final bulk Reynolds numbers $Re_{m,f}$ that were determined at the end of each simulation. To this end, average bulk velocities have been calculated using velocity data of all time steps in the final 5% of the simulation. The velocities have been averaged over space ($d < y < L$) and time ($0.95N_{T,max} < t < N_{T,max}$). The final bulk velocities, Reynolds numbers, and Nusselt numbers of the simulations are listed in Tab. 6.6. The simulation number corresponds with Tab. 6.6.

Examining Fig. 6.10, it is clear that the predicted ice layer thickness d/L exceeds the expected value by approximately 30% in each simulation. To provide context for this variation, we turn to the work of Meyer et al. [73], who analyzed the performance of the Gnielinski correlation across various pipe flow scenarios documented in the literature. For low Nusselt numbers of $Nu < 20$, which corresponds to the values encountered in this thesis, they found a significant deviation of $> \pm 20$% for all low-Nu studies, i.e., [7, 70, 12]. However, Meyer et al. also did two experiments of their own for $Nu < 20$ and found a slightly better agreement with a maximum deviation of 10%.



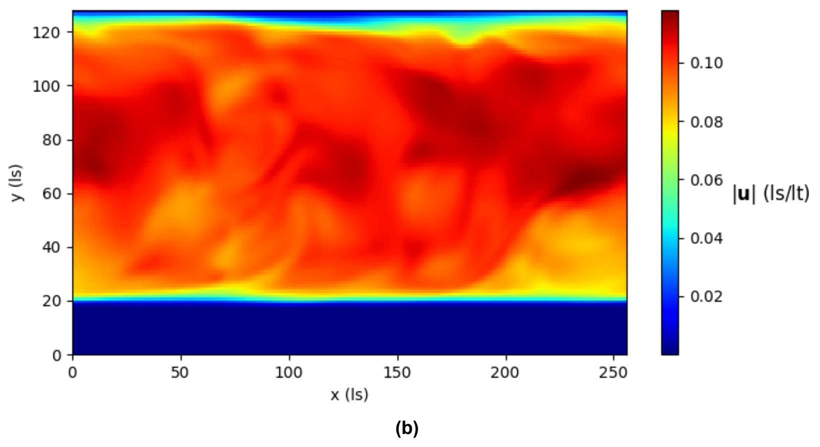


Figure 6.9: (a) Instantaneous temperature and (b) instantaneous velocity profiles of the turbulent freezing simulation with lower wall temperature $T_L = 263.55$ K (simulation 4). Snapshots have been taken at $z = N_z/2$ after 1.3M time steps.

To see the impact of a $\pm 20\%$ deviation from the predicted Nusselt number, corresponding error bars have been plotted in Fig. 6.10. It is visible that the ice layer thickness of the performed simulations approximately coincides with the upper error margins, corresponding to a Nusselt number $\sim 20\%$ lower than predicted using the Gnielinsky correlation. As such deviations are not uncommon in reported experimental data of low-Pr pipe flows, decent accuracy of the performed simulations may still be assumed. To the author's knowledge, there are currently no other numerical simulations of steady-state ice layer development in turbulent channel flows. It would be interesting to see the results of comparable simulations to better assess the accuracy achieved in this research.

6.4.5. Turbulent Flow with non-eutectic Fluid

This research aims to simulate the freezing of MSFR salt in turbulent channel flows. Up to now, it has been assumed that the fluid has a single solidus and liquidus temperature $T_s = T_l$, corresponding to a eutectic fluid. In reality, however, MSFR salts are non-eutectic and have different solidus and liquidus temperatures $T_s < T_l$, leading to a mushy region in which the fluid is in an intermediate state between liquid and solid. In the mushy region, the fluid is a mixture of liquid and solid particles and therefore has different properties compared to the purely liquid or solid phase.

First of all, the solid particles in the mushy region act as a momentum sink for the liquid particles that flow through. This is conveniently accounted for by the immersed boundary method since a liquid fraction $0 < f_l < 1$ leads to a combined effect of collision operators Ω_s and Ω (Eq. 3.23). The former collision operator imposes zero velocity on a node when $f_l = 0$, but it imposes only a reduced velocity when $0 < f_l < 1$. To ensure a stable field, the adapted immersed boundary method in Eq. 6.27 will

Table 6.6: The final bulk velocity, Reynolds number, and Nusselt number at the end of each simulation. The subscripts f denote that the quantities relate to the end of the simulation. The Nusselt number is determined using the Gnielinski correlation. The simulation numbers correspond to Tab. 6.5.

Simulation no.	$u_{m,f}$ (ls/lt)	$Re_{m,f}$	Nu_f
1	0.1018	5378	17.8
2	0.0998	5156	17.2
3	0.0989	4969	16.6
4	0.0950	4531	15.3
5	0.0913	4082	13.8
6	0.0868	3626	12.3

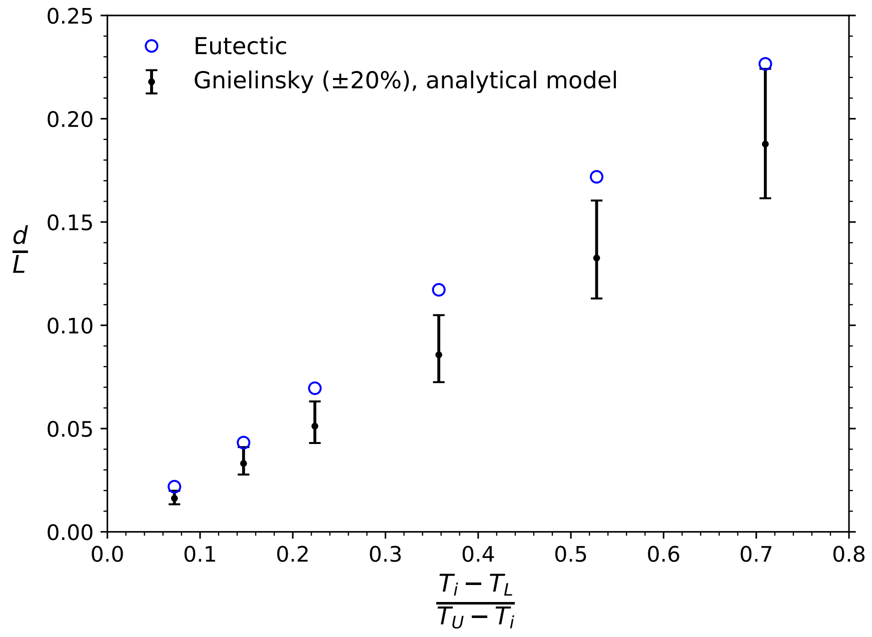


Figure 6.10: Steady state ice layer thickness normalized by the channel height L in a turbulent channel for different values of the temperature fraction $(T_i - T_L)/(T_U - T_i)$. The interface temperature is defined as $T_i = T_s$, where T_s is the solidus temperature. The ice layer thickness d is defined as the average amount of solid nodes in the wall-normal direction at the end of the simulation.

again be used to calculate Ω_s.

Second, the physical properties of the fluid are also changed in the mushy zone. For simplicity, it is assumed that only the specific heat and thermal diffusivity are changed, while other parameters such as viscosity remain unchanged from the liquid phase. It must be noted that the specific properties of MSFR salts are different from the ones that were used in this simulation. This is decided because the current research serves as a proof of concept of the applied methodologies and techniques. This research does not aim to model the exact behavior of specific MSFR salts.

For this reason, most of the input parameters of the performed eutectic simulations (Sec. 6.4.4) have been re-used for the non-eutectic simulations. All parameters are the same as in Tab. 6.1, except for the liquidus temperature, which is set at $T_l = 275.15$. This yields a mushy region between $273.15 < \widetilde{T} < 275.15$. Again, the temperature field is transformed following the procedure in Sec. 6.2.2, using $\widetilde{h}_{min} = 0$ and $\widetilde{h}_{max} = 1$. The input temperatures are listed for all non-eutectic simulations in Tab. 6.7. The Prandtl number remains at $Pr = 0.71$ and no local grid refinement is applied.

Following the recommendation of [10], the specific heat and thermal diffusivity of the mushy zone are both assumed to be averages of the solid and liquid phases. The ice layer thickness d is defined as the thickness of the solid layer, excluding the mushy zone.

The resulting ice layer thickness is shown in Fig. 6.11 for both the eutectic and non-eutectic fluids. Note that the ice layer is slightly thicker for the non-eutectic fluid than for the eutectic fluid. This is a

Table 6.7: Lower wall temperatures T_L used in non-eutectic turbulent freezing simulations. The upper wall, solidus, and liquidus temperatures are set at $T_U = 300$K, $T_s = 273.15$K, and $T_l = 275.15$K, respectively. The corresponding fraction $(T_s - T_L)/(T_U - T_s)$ is given as well. The transformed lower wall temperatures $\widetilde{T}_L$, solidus temperatures $\widetilde{T}_s$, liquidus temperatures $\widetilde{T}_l$, and upper wall temperatures $\widetilde{T}_H$ are also listed. This corresponds to $\widetilde{h}_{min} = 0$ and $\widetilde{h}_{max} = 1$.

Simulation no.	$T_L(K)$	$(T_s - T_L)/(T_U - T_s)$	$\widetilde{T}_L$ (IK)	$\widetilde{T}_s$ (10^{-6} IK)	$\widetilde{T}_l$ (10^{-6} IK)	$\widetilde{T}_U$ (10^{-6} IK)
1	271.21	0.0723	0	16.6	33.7	246
2	269.21	0.1469	0	32.6	49.1	246
3	267.14	0.2238	0	47.9	63.9	262
4	263.55	0.3575	0	72.2	87.3	274
5	258.98	0.5278	0	99.4	113	288
6	254.09	0.7098	0	125	138	300

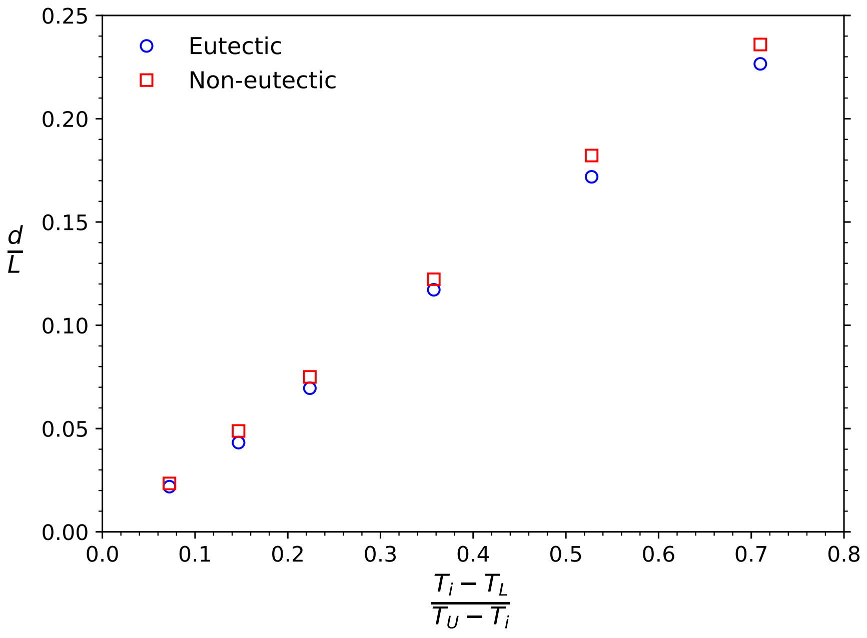


Figure 6.11: Steady state ice layer thickness normalized by the channel height L in a turbulent channel for different values of the temperature fraction $(T_i - T_L)/(T_U - T_i)$. Results for a eutectic and a non-eutectic fluid are given. The interface temperature is defined as $T_i = T_s$ for both simulations, where T_s is the solidus temperature. The ice layer thickness d is defined as the average amount of solid nodes in the wall-normal direction at the end of the simulation.

sensible result, since the heat conductivity $\lambda = \rho\alpha C_p$ is larger in the mushy zone than in the liquid. This means that heat from the fluid is more effectively being conducted towards the cold wall. A thicker ice layer is thus expected.

In the absence of comparable simulations or experimental data on steady-state non-eutectic freezing at varying liquidus temperatures, a quantitative comparison could not be performed. There are studies, such as Mahdaoui et al. [69], that investigate transient ice fronts in laminar flow of binary non-eutectics for different mushy regions. However, changing the concentrations of the binary components changes both the solidus and liquidus temperatures. The independent effect of a varying liquidus temperature can therefore not be determined from such results. In the MSc thesis of Bus [10], transient ice layer growth was compared for eutectic and non-eutectic laminar flows. However, a qualitative comparison with the current steady-state results is not possible, because (1) their latent heat was taken significantly higher for the non-eutectic case, leading to a slower developing non-eutectic ice front, (2) their non-eutectic solidus temperature was chosen to be higher than the eutectic freezing temperature.

Thus, the absence of steady-state freezing results and the discrepancy with simulation parameters of existing transient studies leads to an inability to make accurate comparisons with a reference case. It would be interesting to see comparable simulations of non-eutectic freezing simulations in the future, which can be compared with results obtained in this thesis.

6.5. Conclusion and Discussion

It has been shown that the DDF-FMLBM WALE-LES implemented in this research is suitable for simulating turbulent channel flow in conjunction with heat transfer. A novel set of transformation rules has been introduced that reduces fluctuations and instabilities in thermal simulations of turbulent channel flow. These fluctuations are dependent on the relative difference between the minimum and maximum enthalpies in the domain. Both DNS and LES yielded accurate turbulent statistics that were in good agreement with selected benchmark studies. In addtion, LES proved to be superior to DNS in predicting the turbulent statistics at the center of the channel.

Moreover, the development of a steady-state ice layer has been simulated using the immersed boundary method. An alternative approach to the standard method has been applied to prevent non-physical velocities and temperature fluctuations near the solid-liquid interface. Accurate results were obtained for a stationary channel with zero velocity and a close agreement with the analytical solution was observed. However, an under-prediction of approximately one cell was found in all simulations.

Simulations of freezing in turbulent channel flows have been performed using the same methodology and were benchmarked with analytical solutions using Gnielinski's Nusselt correlation. The obtained ice layers were ~ 30% thicker than expected from the analytical solutions, which corresponds to a 20% over-prediction of the Nusselt number by Gnielinski's correlation. However, previously performed experiments showed deviations of a similar order and the results are therefore still within acceptable limits. Furthermore, simulations of non-eutectic fluids were also performed, which yielded slightly thicker ice layers in comparison with the eutectic simulations. This was expected due to the higher heat conductivity in the mushy region, compared to the liquid phase.

6.5.1. High Prandtl Numbers

The goal of this research was to simulate the solidification of non-eutectic MSFR salt in turbulent channel flow. Although many aspects of this goal have been achieved, the most important oversimplification in this work is the choice of Prandtl number. While this work restricted itself to $Pr = 0.71$, preventing instabilities, the actual Prandtl numbers of realistic MSFR salts lie in the range $7.5 \leq Pr \leq 20$ [33]. The low Prandtl restriction was caused by instabilities that occurred when the thermal diffusivity became too low. When simulating laminar flows, a high Prandtl number poses no immediate problems because the viscosity can generally be much higher compared to turbulent flows.

The difficulty in studying high Prandtl turbulent flows in the LBM framework is widespread. Most thermal studies of turbulent channel flow restrict themselves to $Pr \leq 1$. However, there are recent studies that introduce additional stability to the simulation, such as Gruszczynski and Laniewski-Wollk [42], who implemented a novel collision kernel, a D3Q27 lattice, and interpolated boundary conditions to reach very high Prandtl numbers (up to 1000 with minimum thermal conductivities of $\mathcal{O}(10^{-5})$).

A study that might be particularly promising for the filter-matrix LBM is the new LBM approach that was introduced for the SRT and MRT framework in 2021 by Du et al. [25]. This approach enabled them to reach values up to $Pr = 56.2$. They proposed a modification of the thermal equilibrium distribution function using a scaling factor η and included the same factor in the definition of the Prandtl number. This factor makes it possible to reach high Prandtl numbers while ensuring stability during the collision step. Because this approach was constructed in the SRT framework, it might be relatively straightforward to translate the same methodology to the FMLBM framework. The derivation of an adjusted solution vector and/or filter matrix was excluded from the scope of this research but is recommended for future implementation.

7 Conclusion and Recommendation

This thesis investigated freezing of non-eutectic fluids in 3D turbulent channel flows. To this end, a large eddy simulation has been implemented in the FMLB framework with local grid refinement to model turbulent flow, heat transfer, and solidification in a channel bounded by infinite parallel plates. To accelerate the simulation, a GPU implementation was adopted for parallel calculations. This chapter discusses the conclusions that follow from the performed work and follows up with recommendations for future research.

7.1. Conclusion

The conclusions from this work are now stated and provide an answer to the research questions formulated in Ch. 1. Conclusions are given on (1) the local grid refinement implementations, (2) the modeling of turbulent channel flow using WALE-LES in FM-LBM, and (3) the modeling of phase change of eutectic and non-eutectic fluids. These will be integrated with conclusions on adopted BCs, input parameters, and comparisons with reference cases.

7.1.1. Local Grid Refinement

In this research, a large eddy simulation (LES) is implemented, which does not fully resolve turbulent scales beyond a specific cut-off length, defined as the grid spacing. For channel flows, LES permits a coarser grid spacing in the central region while requiring full resolution at the walls. To leverage this coarser grid spacing, two local grid refinement techniques were evaluated: the Rohde algorithm developed by [92] and a novel algorithm introduced in this study. Unlike the Rohde algorithm, the new approach avoids additional non-physical collisions.

For laminar flows, the new technique did not converge to the analytical Poisueille profile over time and no grid convergence was observed. This was attributed to a loss of information during the redistribution from coarse to fine cells in the interface layer. Better results were expected when an interpolation scheme is used during this step, as was also done in a similar approach by [87]. However, the Rohde algorithm was both time-convergent and second-order grid-convergent for the $r = 2$ and $r = 4$ versions. This led to its implementation in subsequent simulations.

During the turbulent channel flow simulations, the $r = 2$ Rohde algorithm closely agreed with the data by [6, 55, 76]. However, slight dips in the RMS velocity fluctuations in the wall-normal and spanwise directions were observed near the refinement interface. This behavior was also observed by [92] and was attributed to non-physical steps in the algorithm. The $r = 4$ algorithm was also tested for turbulent flows but showed significant over-prediction of the velocity and it was therefore not deemed useful.

Furthermore, the $r = 2$ algorithm yielded effectively 40% higher efficiency for $Re_\tau = 180$ and 240% for $Re_\tau = 395$, compared to the unrefined simulations. It can therefore be concluded that a local grid refinement technique significantly reduces computation time, without noticeably affecting the accuracy of the obtained results. However, the MLUPS of the local grid refinement simulations were more than halved, compared to the unrefined cases. Higher computational performance is expected for the local grid refinement technique when using better management of threads and blocks.

7.1.2. WALE-LES in Turbulent Channel Flow

Multiple LES simulations of turbulent channel flows were performed using the WALE sub-grid scale model. They were compared to DNS simulations with the same input parameters and conditions. The WALE model constant was set to $C_w = 0.5$, based on recommendations in prior work. For both $Re_\tau = 180$ and $Re_\tau = 395$ simulations, the WALE-LES approach gave a superior estimation of the mean stream-wise velocity compared to the performed DNS. The detailed DNS by [55] and [76] were used as reference cases.

It was found that deviation from the reference cases was halved using LES, compared to the performed DNS simulations, for both $Re_\tau = 180$ and $Re_\tau = 395$. Results from the $Re_\tau = 395$ DNS gave a larger deviation from the reference case than the $Re_\tau = 180$ DNS, so the largest absolute error reduction was achieved in the $Re_\tau = 395$ LES. The difference in accuracy for the two Reynolds numbers was attributed to the coarser $Re_\tau = 395$ grid compared to the $Re_\tau = 180$ grid ($\Delta y^+ = 3.4$ vs $\Delta y^+ = 2.8$, respectively). Additionally, LES is more effective for larger Reynolds numbers, because more turbulent energy is concentrated at the smaller scales.

7.1.3. Thermal Simulations

A thermal simulation of turbulent flow between walls with a fixed temperature difference was implemented, assuming a fully liquid phase. When the relative temperature differences $(T_U - T_L)/T_U$ between the upper and lower walls were small, significant fluctuations appeared in the thermal field. This resulted in an unstable simulation that diverged after $\mathcal{O}(10^4)$ time steps. This is a known problem that was also encountered in earlier work, where it could not be resolved. In this thesis, it was shown that the fluctuations scale directly with the local enthalpy, which led to the introduction of transformation rules for temperature, enthalpy, and latent heat. They lead to a lower fluctuation with less relative significance because the temperature and enthalpy spectra are broadened. This led to stable simulation results without significant fluctuation.

Using this transformation procedure, thermal statistics were obtained from thermal DNS and LES simulations. The mean temperature and turbulent heat flux agreed closely with data of [90, 53] for both LES and DNS. The temperature fluctuations showed a slight undershoot, compared to the reference data, with LES results being slightly more accurate. A comparable undershoot was also observed in the LES of [120], so the observed deviation falls within a reasonable margin.

In the thermal simulations, no local grid refinement was tested, because the lower thermal diffusivity in the coarse layer causes instability at $Pr = 0.71$.

7.1.4. Phase Change

Thermal simulations with phase change were incorporated using the standard immersed boundary method by [81]. This gave rise to a negative stream-wise velocity close to the freezing interface, which was resolved by applying an adapted collision operator Ω^s that inherently sets all velocities in the solid phase to zero. Furthermore, a value $\zeta > 0.01$ was recommended for the parameter ζ, which appears in the definition of the parameter B in the adapted LBE of the immersed boundary method. A lower value for ζ led to instability.

The first freezing simulations were performed with a eutectic zero-velocity system and a cold wall temperature below the freezing point. Each simulation had a different cold wall temperature and the resulting steady-state ice thicknesses were compared with analytical solutions. A close agreement was observed with an undershoot of one cell thickness across all simulations.

Subsequently, freezing simulations with non-zero velocity were conducted using a eutectic turbulent fluid, and results were compared with analytical solutions using Gnielinski's Nusselt correlation. Across all simulations, the ice thickness was found to be over-predicted by 30%, corresponding to a Nusselt number 20% lower than estimated by the Gnielinski correlation. Deviations of ±20% are within the expected range based on a prior study on the accuracy of Nusselt correlations by [73]. This discrepancy is attributed to the inherent limitations of Nusselt correlations in accurately capturing individual experimental scenarios. These correlations provide an average representation based on numerous experiments and may not precisely match specific cases.

Lastly, freezing simulations of a non-eutectic turbulent fluid were performed. It was observed that the steady-state ice layer thickness was slightly larger in the presence of a mushy zone when using a higher liquidus temperature than in the eutectic case, keeping other thermal inputs the same. This was expected due to the higher thermal conductivity in the mushy zone, compared to the liquid phase.

7.2. Recommendation

Recommendations for further research are now presented. They can be categorized into local grid refinement recommendations, recommendations on LB-LES applications, recommendations on thermal LB, and GPU-related recommendations.

Local Grid Refinement Recommendations

- **Increased refinement levels**: This study employed only two levels of refinement, resulting in excessive refinement in the central channel area. By incorporating additional refinement levels, optimal refinement can be achieved at every location in the channel. In determining the amount of refinement levels, one must balance the computational gain with the increased complexity.
- **Adaptive refinement for tracking solid-liquid interface**: The fine-liquid interface behaves as a solid no-slip wall, and therefore requires a fine grid in this region. Due to the dynamic nature of the interface, an adaptive grid is beneficial. This approach prevents unnecessarily fine grids at the start of the simulation and ensures that the solid-liquid interface remains adequately resolved throughout the process.

Thermal LB and Phase Change

- **Adapted schemes for high Prandtl numbers**: Due to stability restrictions, the simulated Prandtl number in this research was 0.71, while MSFR salt generally has a value $7.5 \leq Pr \leq 20$. It is recommended to implement advanced methods that allow for high Prandtl number calculations in turbulent flows. Potential options are a modification to the equilibrium distribution function, as proposed by [25], or one of the approaches that were proposed by [42].
- **Inflow and outflow Boundary Conditions**: This research investigated periodic boundary conditions in the stream-wise direction. However, nearly all research on freezing turbulent channel flow assumes the inflow of an equithermal fluid at the inlet which leaves the system at the outlet. To allow for extensive benchmarking, it is recommended to simulate the in- and outflow of salt, instead of a periodic flow.

GPU Implementation

- **Advanced Collision and Propagation Algorithms**: The slowest steps in the algorithm are the propagation step (slowest) and the collision step (second slowest). A more efficient use of memory can lead to a significant reduction in computation time. This can be achieved with more sophisticated algorithms, as was implemented in [102, 122], and by implementing a combined collision and propagation kernel. C-based languages offer greater potential for this than the Numba library in Python because they have more CUDA functionalities available.

For more hands-on recommendations on GPU-based LBM, one can resort to Sec. 5.5.3.

References

[1] Andy Adinets. *CUDA Dynamic Parallelism API and Principles*. 2014. URL: `https://developer.nvidia.com/blog/cuda-dynamic-parallelism-api-principles/`.

[2] International Atomic Energy Agency. "IAEA Projections for Nuclear Power Growth Increase for Second Year Amid Climate, Energy Security Concerns". In: (Sept. 2022). URL: `https://www.iaea.org/newscenter/pressreleases/iaea-projections-for-nuclear-power-growth-increase-for-second-year-amid-climate-energy-security-concerns#:~:text=According%5C%20to%5C%20the%5C%20new%5C%20projections,from%5C%20its%5C%2010%5C%25%5C%20share%5C%20today`. (visited on 03/03/2023).

[3] International Energy Agency. "An updated roadmap to Net Zero Emissions by 2050". In: (2022). URL: `https://www.iea.org/reports/world-energy-outlook-2022/an-updated-roadmap-to-net-zero-emissions-by-2050` (visited on 03/03/2023).

[4] Frank J Alexander, Shiyi Chen, and JD Sterling. "Lattice boltzmann thermohydrodynamics". In: *Physical Review E* 47.4 (1993), R2249.

[5] M. Allibert et al. "7 - Molten salt fast reactors". In: *Handbook of Generation IV Nuclear Reactors*. Ed. by Igor L. Pioro. Woodhead Publishing Series in Energy. Woodhead Publishing, 2016, pp. 157–188. ISBN: 978-0-08-100149-3. DOI: `https://doi.org/10.1016/B978-0-08-100149-3.00007-0`. URL: `https://www.sciencedirect.com/science/article/pii/B9780081001493000070`.

[6] G Amati, S Succi, and R Piva. "Preliminary analysis of the scaling exponents in channel flow turbulence". In: *Fluid dynamics research* 24.4 (1999), pp. 201–209.

[7] JF Barnes and JD Jackson. "Heat transfer to air, carbon dioxide and helium flowing through smooth circular tubes under conditions of large surface/gas temperature ratio". In: *Journal of Mechanical Engineering Science* 3.4 (1961), pp. 303–314.

[8] Jiri Blazek. "Chapter 7 - Turbulence Modeling". In: *Computational Fluid Dynamics: Principles and Applications (Third Edition)*. Ed. by Jiri Blazek. Oxford: Butterworth-Heinemann, Jan. 2015. ISBN: 978-0-08-099995-1. URL: `https://www.sciencedirect.com/science/article/pii/B9780080999951000075` (visited on 04/10/2023).

[9] Joseph Boussinesq. *Essai sur la théorie des eaux courantes*. Impr. nationale, 1877.

[10] Celeke Bus. "Simulating the Transient Freezing in cooled Non-eutectic Molten Salt Channel Flow". MA thesis. Delft University of Technology, 2022.

[11] Celeke Bus and Martin Rohde. "A Filter Matrix lattice-Boltzmann method for melting and solidification in convective flows". Unpublished manuscript. 2024.

[12] Orhan Buyukalaca, Veysel Ozceyhan, and Sibel Gunes. "Experimental investigation of thermal performance in a tube with detached circular ring turbulators". In: *Heat transfer engineering* 33.8 (2012), pp. 682–692.

[13] N Caruso, M Portapila, and Henry Power. "An efficient and accurate implementation of the localized regular dual reciprocity method". In: *Computers & Mathematics with Applications* 69.11 (2015), pp. 1342–1366.

[14] Suman Chakraborty and Dipankar Chatterjee. "An enthalpy-based hybrid lattice-Boltzmann method for modelling solid–liquid phase transition in the presence of convective transport". In: *Journal of Fluid Mechanics* 592 (Dec. 2007), pp. 155–175. ISSN: 0022-1120, 1469-7645. DOI: `10.1017/S0022112007008555`. URL: `https://www.cambridge.org/core/product/identifier/S0022112007008555/type/journal_article` (visited on 04/16/2023).

[15] Yongguang Cheng and Hui Zhang. "A viscosity counteracting approach in the lattice Boltzmann BGK model for low viscosity flow: Preliminary verification". In: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3690–3702.

[16] *Climate Change: Atmospheric Carbon Dioxide*. 2023. URL: `https://www.climate.gov/news-features/understanding-climate/climate-change-atmospheric-carbon-dioxide#:~:text=Without%20carbon%20dioxide%2C%20Earth's%20natural,causing%20global%20temperature%20to%20rise.`.

[17] Christophe Coreixas et al. "Impact of collision models on the physical properties and the stability of lattice Boltzmann methods". In: *Philosophical Transactions of the Royal Society A* 378.2175 (2020), p. 20190397.

[18] *CUDA - Memory Hierarchy*. URL: `http://thebeardsage.com/cuda-memory-hierarchy/`.

[19] *CUDA – Streaming Multiprocessors*. URL: `http://thebeardsage.com/cuda-streaming-multiprocessors/`.

[20] *CUDA Toolkit*. URL: `https://developer.nvidia.com/cuda-toolkit`.

[21] Lars Davidson. *pyCALC-LES: A Python Code for DNS, LES and Hybrid LES-RANS*. 2023.

[22] B Debusschere and CJ Rutland. "Turbulent scalar transport mechanisms in plane channel and Couette flows". In: *International Journal of heat and mass transfer* 47.8-9 (2004), pp. 1771–1781.

[23] Nicolas Delbosc et al. "Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation". In: *Computers & Mathematics with Applications* 67.2 (2014), pp. 462–475.

[24] *DelftBlue Documentation*. 2024. URL: `https://doc.dhpc.tudelft.nl/delftblue/`.

[25] Wenhui Du, Sheng Chen, and Guoqiang Wu. "A new lattice Boltzmann method for melting processes of high Prandtl number phase change materials". In: *Journal of Energy Storage* 41 (2021), p. 103006.

[26] Alexandre Dupuis and Bastien Chopard. "Theory and applications of an alternative lattice Boltzmann grid refinement algorithm". In: *Physical Review E* 67.6 (2003), p. 066707.

[27] J. G.M. Eggels and J. A. Somers. "Numerical simulation of free convective flow using the lattice-Boltzmann scheme". In: *International Journal of Heat and Fluid Flow* 16.5 (1995), pp. 357–364. ISSN: 0142727X. DOI: `10.1016/0142-727X(95)00052-R`.

[28] *Examining Spatial (Grid) Convergence*. NASA. 2021. URL: `https://www.grc.nasa.gov/www/wind/valid/tutorial/spatconv.html`.

[29] ExxonMobil. *Energy demand: Three drivers*. 2024. URL: `https://corporate.exxonmobil.com/what-we-do/energy-supply/global-outlook/energy-demand#:~:text=Global%20demand%20reaches%20about%20660,growing%20population%20and%20rising%20prosperity.`.

[30] David Farley. *Modern Software Engineering: Doing What Works to Build Better Software Faster*. Addison-Wesley Professional, 2021.

[31] Olga Filippova and Dieter Hänel. "Grid refinement for lattice-BGK models". In: *Journal of Computational physics* 147.1 (1998), pp. 219–228.

[32] Carlo Fiorina et al. "Modelling and analysis of the MSFR transient behaviour". In: *Annals of Nuclear Energy* 64 (2014), pp. 485–498.

[33] Carlo Fiorina et al. "Thermal-hydraulics of internally heated molten salts and application to the Molten Salt Fast Reactor". In: *Journal of Physics: Conference Series*. Vol. 501. 1. IOP Publishing. 2014, p. 012030.

[34] TOM Forslund et al. "A dual-lattice hydrodynamic-thermal MRT-LBM model implemented on GPU for DNS calculations of turbulent thermal flows". In: *International Journal of Numerical Methods for Heat & Fluid Flow* ahead-of-print (2022).

[35] Eric Garnier, Pierre Sagaut, and Michel Deville. "Large eddy simulation of shock/boundary-layer interaction". In: *AIAA journal* 40.10 (2002), pp. 1935–1944.

[36] *Generation IV Systems*. URL: `https://www.gen-4.org/gif/jcms/c_40465/generation-iv-systems`.

[37] Massimo Germano et al. “A dynamic subgrid□scale eddy viscosity model”. en. In: *Physics of Fluids A: Fluid Dynamics* 3.7 (July 1991), pp. 1760–1765. ISSN: 0899-8213. DOI: 10.1063/1.857955. URL: http://aip.scitation.org/doi/10.1063/1.857955 (visited on 04/11/2023).

[38] Sandip Ghosal. “An Analysis of Numerical Errors in Large-Eddy Simulations of Turbulence”. en. In: *Journal of Computational Physics* 125.1 (Apr. 1996), pp. 187–206. ISSN: 00219991. DOI: 10.1006/jcph.1996.0088. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999196900881 (visited on 09/05/2023).

[39] Volker Gnielinski. “New equations for heat and mass transfer in the turbulent flow in pipes and channels”. In: *NASA STI/recon technical report A* 41.1 (1975), pp. 8–16.

[40] Stephen M. Goldberg and Robert Rosner. *Nuclear Reactors: Generation to Generation*. American Academy of Arts & Sciences, 2011.

[41] *GPU Architecture*. Google Colab. URL: https://colab.research.google.com/github/d2l-ai/d2l-tvm-colab/blob/master/chapter_gpu_schedules/arch.ipynb.

[42] G. Gruszczyński and Ł. Łaniewski-Wołłk. “A comparative study of 3D cumulant and central moments lattice Boltzmann schemes with interpolated boundary conditions for the simulation of thermal flows in high Prandtl number regime”. In: *International Journal of Heat and Mass Transfer* 197 (Nov. 2022), p. 123259. ISSN: 0017-9310. DOI: 10.1016/j.ijheatmasstransfer.2022.123259. URL: http://dx.doi.org/10.1016/j.ijheatmasstransfer.2022.123259.

[43] Zhaoli Guo and Chang Shu. *Lattice Boltzmann method and its application in engineering*. Vol. 3. World Scientific, 2013.

[44] Pradeep Gupta. *CUDA Refresher: The CUDA Programming Model*. URL: https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/.

[45] R. Haberman. *Applied Partial Differential Equations with Fourier series and boundary value problems*. 5th ed. Pearson, 2013.

[46] Nancy Hall. *Navier-Stokes Equations*. 2021. URL: https://www.grc.nasa.gov/www/k-12/airplane/nseqs.html.

[47] Basim O Hasan. “Turbulent Prandtl number and its use in prediction of heat transfer coefficient for liquids”. In: *Al-Nahrain Journal for Engineering Sciences* 10.1 (2007), pp. 53–64.

[48] Zhuojie He and Ruifang Liang. “Coupled double-distribution-function lattice Boltzmann model with streaming-collision process”. In: *Journal of Physics: Conference Series*. Vol. 2441. 1. IOP Publishing. 2023, p. 012052.

[49] Rongzong Huang, Huiying Wu, and Ping Cheng. “A new lattice Boltzmann model for solid-liquid phase change”. In: *International Journal of Heat and Mass Transfer* 59.1 (2013), pp. 295–301. ISSN: 00179310. DOI: 10.1016/j.ijheatmasstransfer.2012.12.027.

[50] Ashleigh J Hutchinson. “The extended Prandtl closure model applied to the two-dimensional turbulent classical far wake”. In: *2019-20 MATRIX Annals* (2021), pp. 375–385.

[51] IAEA. *MSFR (CNRS, France)*. IAEA Report. URL: https://aris.iaea.org/PDF/MSFR.pdf.

[52] L.P.B.M. Janssen and M.M.C.G. Warmoeskerken. *Transport Phenomena Data Companion*. VSSD, 2006.

[53] Hiroshi Kawamura, Hiroyuki Abe, and Kenji Shingai. “DNS of turbulence and heat transport in a channel flow with different Reynolds and Prandtl numbers and boundary conditions”. In: *Turbulence, Heat and Mass Transfer* 3 (2000), pp. 15–32.

[54] John E. Kelly. “Generation IV International Forum: A decade of progress through international cooperation”. In: *Progress in Nuclear Energy* 77 (2014), pp. 240–246. ISSN: 0149-1970. DOI: https://doi.org/10.1016/j.pnucene.2014.02.010. URL: https://www.sciencedirect.com/science/article/pii/S0149197014000419.

[55] John Kim, Parviz Moin, and Robert Moser. “Turbulence statistics in fully developed channel flow at low Reynolds number”. In: *Journal of fluid mechanics* 177 (1987), pp. 133–166.

[56] Timm Krüger et al. *The Lattice Boltzmann Method Principles and Practice*. Springer, 2017. URL: http://www.springer.com/series/8431.

[57] Yoshiaki Kuwata and Kazuhiko Suga. “Wall-modeled large eddy simulation of turbulent heat transfer by the lattice Boltzmann method”. In: *Journal of Computational Physics* 433 (2021), p. 110186.

[58] Pierre Lallemand and Li-Shi Luo. “Theory of the lattice Boltzmann method: Acoustic and thermal properties in two and three dimensions”. In: *Physical review E* 68.3 (2003), p. 036706.

[59] N Le Brun, GF Hewitt, and CN Markides. “Transient freezing of molten salts in pipe-flow systems: Application to the direct reactor auxiliary cooling system (DRACS)”. In: *Applied Energy* 186 (2017), pp. 56–67.

[60] Athony Leonard. “Energy cascade in large-eddy simulations of turbulent fluid flows”. In: *Advances in geophysics*. Vol. 18. Elsevier, 1975, pp. 237–248.

[61] Michael Leschziner. *Statistical turbulence modelling for fluid dynamics-demystified: an introductory text for graduate engineering students*. World Scientific, 2015.

[62] E. Lévêque et al. “Shear-improved Smagorinsky model for large-eddy simulation of wall-bounded turbulent flows”. en. In: *Journal of Fluid Mechanics* 570 (Jan. 2007). Publisher: Cambridge University Press, pp. 491–502. ISSN: 0022-1120, 1469-7645. DOI: `10.1017/S0022112006003429`. URL: `https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/shearimproved-smagorinsky-model-for-largeeddy-simulation-of-wallbounded-turbulent-flows/6D55B21C69CEFAE0845AEBFEF0644119` (visited on 03/29/2023).

[63] Z Li et al. “A novel nonlinear radiative heat exchanger for molten-salt applications”. In: *Applied Thermal Engineering* 225 (2023), p. 120157.

[64] Douglas K Lilly. “A proposed modification of the Germano subgrid-scale closure method”. In: *Physics of Fluids A: Fluid Dynamics* 4.3 (1992), pp. 633–635.

[65] Ching-Long Lin and Yong G Lai. “Lattice Boltzmann method on composite grids”. In: *Physical Review E* 62.2 (2000), p. 2219.

[66] Tong-Miin Liou and Chun-Sheng Wang. “Large eddy simulation of rotating turbulent flows and heat transfer by the lattice Boltzmann method”. In: *Physics of Fluids* 30.1 (Jan. 2018), p. 015106. ISSN: 1070-6631, 1089-7666. DOI: `10.1063/1.5005901`. URL: `http://aip.scitation.org/doi/10.1063/1.5005901` (visited on 04/06/2023).

[67] Ming Liu, Xiao-Peng Chen, and Kannan N Premnath. “Comparative study of the large eddy simulations with the lattice Boltzmann method using the wall-adapting local eddy-viscosity and Vreman subgrid scale models”. In: *Chinese Physics Letters* 29.10 (2012), p. 104706.

[68] Li-Shi Luo et al. “Numerics of the lattice Boltzmann method: Effects of collision models on the lattice Boltzmann simulations”. In: *Physical Review E* 83.5 (2011), p. 056710.

[69] M Mahdaoui et al. “Laminar flow in circular tube with internal solidification of a binary mixture”. In: *Energy* 78 (2014), pp. 713–719.

[70] Donald Marinus McEligot. *Effect of large temperature gradients on turbulent flow of gases in the downstream region of tubes*. Stanford University, 1963.

[71] Patrick McMurtry. “Length and Time Scales in Turbulent Flows”. Lecture Notes. Salt Lake City, Jan. 2001.

[72] D. Mehta et al. “Large eddy simulation with energy-conserving schemes and the smagorinsky model: A note on accuracy and computational efficiency”. In: (2019). (Visited on 03/16/2023).

[73] Josua P Meyer et al. “Heat transfer coefficients of laminar, transitional, quasi-turbulent and turbulent flow in circular tubes”. In: *International Communications in Heat and Mass Transfer* 105 (2019), pp. 84–106.

[74] Johan Meyers and Pierre Sagaut. “On the model coefficients for the standard and the variational multi-scale Smagorinsky model”. en. In: *Journal of Fluid Mechanics* 569 (Dec. 2006). Publisher: Cambridge University Press, pp. 287–319. ISSN: 1469-7645, 0022-1120. DOI: `10.1017/S0022112006002850`. URL: `https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/on-the-model-coefficients-for-the-standard-and-the-variational-multiscale-smagorinsky-model/43F96FCE0CE22EEF4F1B2122F16126F7` (visited on 03/17/2023).

[75] *Molten Salt Reactors*. 2021. URL: https://world-nuclear.org/information-library/current-and-future-generation/molten-salt-reactors.aspx.

[76] Robert D Moser, John Kim, and Nagi N Mansour. “Direct numerical simulation of turbulent channel flow up to Re τ= 590”. In: *Physics of fluids* 11.4 (1999), pp. 943–945.

[77] Eric Phillip Muntz, David P Weaver, and David H Campbell. *Rarefied gas dynamics: theoretical and computational techniques*. Vol. 118. American Institute of aeronautics and astronautics, 1989.

[78] M. Nguyen et al. “Large eddy simulation of a thermal impinging jet using the lattice Boltzmann method”. en. In: *Physics of Fluids* 34.5 (May 2022), p. 055115. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/5.0088410. URL: https://aip.scitation.org/doi/10.1063/5.0088410 (visited on 03/29/2023).

[79] F Nicoud and F Ducros. “Subgrid-Scale Stress Modelling Based on the Square of the Velocity Gradient Tensor”. In: *Flow, Turbulence and Combustion* 62 (1999), pp. 183–200.

[80] Frans TM Nieuwstadt, Bendiks J Boersma, and Jerry Westerweel. *Turbulence: Introduction to Theory and Applications of Turbulent Flows*. Springer International Publishing, 2016.

[81] DR Noble and JR Torczynski. “A lattice-Boltzmann method for partially saturated computational cells”. In: *International Journal of Modern Physics C* 9.08 (1998), pp. 1189–1201.

[82] Numba. *Reference Manual: Deviations from Python Semantics*. URL: https://numba.readthedocs.io/en/stable/reference/pysemantics.html#global-and-closure-variables.

[83] *Numba for CUDA GPUs*. 2024. URL: https://numba.readthedocs.io/en/stable/cuda/index.html.

[84] NVIDIA Corporation. *CUDA C++ Programming Guide*. Version 12.3. NVIDIA Corporation. 2024. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#programming-model.

[85] *NVIDIA TESLA V100 GPU ARCHITECTURE*. Tech. rep. NVIDIA, 2017.

[86] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[87] Jiaxing Qi, Harald Klimach, and Sabine Roller. “Implementation of the compact interpolation within the octree based Lattice Boltzmann solver Musubi”. In: *Computers & Mathematics with Applications* 78.4 (2019), pp. 1131–1141.

[88] J. N. Reddy. *An Introduction to Continuum Mechanics*. Cambrdige University Press, 2013.

[89] J. N. Reddy. *Principles of Continuum Mechanics: A Study of Conservation Principles with Applications*. Cambridge University Press, 2018. DOI: 10.1017/CBO9780511763212.

[90] Feng Ren, Baowei Song, and Haibao Hu. “Lattice Boltzmann simulations of turbulent channel flow and heat transport by incorporating the Vreman model”. In: *Applied Thermal Engineering* 129 (2018), pp. 463–471.

[91] Juan Reyes Barraza. “A generalised lattice Boltzmann method with block-structured adaptive mesh refinement”. PhD thesis. University of Southampton, 2021.

[92] M Rohde, D Kandhai, and J J Derksen. “A generic, mass conservative local grid re nement technique for lattice-Boltzmann schemes”. In: *International Journal for Numerical Methods in Fluids* 51 (2006), pp. 439–468.

[93] Pierre Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media, 2005.

[94] Alexander Schukmann et al. “Analysis of Hierarchical Grid Refinement Techniques for the Lattice Boltzmann Method by Numerical Experiments”. In: *Fluids* 8.3 (2023), p. 103.

[95] J. Kenneth Shultis and Richard E. Faw. *Fundamentals of Nuclear Science and Engineering*. Marcel Dekker, Inc., 2002.

[96] J. Smagorinsky. “General Circulation Experiments with the Primitive Equations: I. The Basic Experiment”. In: *Monthly Weather Review* 91.3 (Mar. 1963). Publisher: American Meteorological Society Section: Monthly Weather Review, pp. 99–164. ISSN: 1520-0493, 0027-0644. (Visited on 04/10/2023).

[97] *Solar Power Molten Salt*. URL: https://www.yara.com/industrial-nitrogen/solar-power-molten-salt/.

[98] J. A. Somers. “Direct simulation of fluid flow with cellular automata and the lattice-Boltzmann equation”. In: *Applied Scientific Research* 51.1-2 (June 1993). Publisher: Springer, pp. 127–133. ISSN: 00036994. DOI: 10.1007/BF01082526. URL: https://link.springer.com/article/10.1007/BF01082526 (visited on 09/13/2022).

[99] O Erik Strack and Benjamin K Cook. “Three-dimensional immersed boundary conditions for moving solids in the lattice-Boltzmann method”. In: *International Journal for Numerical Methods in Fluids* 55.2 (2007), pp. 103–125.

[100] Dawid Taler and Jan Taler. “Simple heat transfer correlations for turbulent tube flow”. In: *E3S Web of conferences*. Vol. 13. EDP Sciences. 2017, p. 02008.

[101] *The Concept of the Molten Salt Fast Reactor*. URL: https://samosafer.eu/project/concept/.

[102] Jonas Tölke. “Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA”. In: *computing and Visualization in Science* 13.1 (2010), p. 29.

[103] Nhat-Phuong Tran, Myungho Lee, Sugwon Hong, et al. “Performance optimization of 3D lattice Boltzmann flow solver on a GPU”. In: *Scientific Programming* 2017 (2017).

[104] *unittest — Unit testing framework*. Python Software Foundation. 2024. URL: https://docs.python.org/3/library/unittest.html.

[105] Daniel Van Bemmelen. “Influence of turbulence on the internal conductivity and total electrical resistance of a carbon black suspension inside a Semi-Solid Flow Battery”. MA thesis. Delft University of Technology, 2023.

[106] S Van Buren, A Cárdenas Miranda, and W Polifke. “Large eddy simulation of enhanced heat transfer in pulsatile turbulent channel flow”. In: *International Journal of Heat and Mass Transfer* 144 (2019), p. 118585.

[107] Christoph Van Treeck et al. “Extension of a hybrid thermal LBE scheme for large-eddy simulations of turbulent convective flows”. In: *Computers & Fluids* 35.8-9 (2006), pp. 863–871.

[108] J.H.S. Van Winden. “Developing a Hybrid Lattice Boltzmann Finite Difference Model for Phase Change in the Molten Salt Reactor”. MA thesis. Delft University of Technology, 2021.

[109] J.H.S. Van Winden. “Investigating Adaptive Mesh Refinement Criteria for a Double-Distribution FMLB Scheme in Melting and Solidifaction Processes”. MA thesis. Delft University of Technology, 2022.

[110] Vaughan R Voller, M Cross, and NC Markatos. “An enthalpy method for convection/diffusion phase change”. In: *International journal for numerical methods in engineering* 24.1 (1987), pp. 271–284.

[111] A. W. Vreman. “An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications”. en. In: *Physics of Fluids* 16.10 (Oct. 2004), pp. 3670–3681. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/1.1785131. URL: http://aip.scitation.org/doi/10.1063/1.1785131 (visited on 03/17/2023).

[112] Bert Vreman, Bernard Geurts, and Hans Kuerten. “Comparison of numerical schemes in large-eddy simulation of the temporal mixing layer”. In: *International Journal for Numerical Methods in Fluids* 22.4 (Feb. 1996), pp. 297–311. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/(SICI)1097-0363(19960229)22:4<297::AID-FLD361>3.0.CO;2-X. URL: https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-0363(19960229)22:4%3C297::AID-FLD361%3E3.0.CO;2-X (visited on 04/12/2023).

[113] Min Wang et al. “A novel algorithm of immersed moving boundary scheme for fluid–particle interactions in DEM–LBM”. In: *Computer Methods in Applied Mechanics and Engineering* 346 (2019), pp. 109–125.

[114] Xian Wang et al. “Direct numerical simulation and large eddy simulation on a turbulent wall-bounded flow using lattice Boltzmann method and multiple GPUs”. In: *Mathematical Problems in Engineering* 2014 (2014).

[115] *Water - Prandtl Number vs. Temperature and Pressure*. URL: `https://www.engineeringtoolbox.com/water-steam-Prandtl-number-d_2059.html`.

[116] Mathias Weickert et al. “Investigation of the LES WALE turbulence model within the lattice Boltzmann framework”. In: *Computers & Mathematics with Applications* 59.7 (2010), pp. 2200–2214.

[117] Philipp Wellinger et al. “Analysis of turbulence structures and the validity of the linear Boussinesq hypothesis for an infinite tube bundle”. In: *International Journal of Heat and Fluid Flow* 91 (2021), p. 108779.

[118] M Grae Worster. “Convection in mushy layers”. In: *Annual Review of Fluid Mechanics* 29.1 (1997), pp. 91–122.

[119] Mees Wortelboer. “Investigating GPU-accelerated Double Distribution Function Lattice Boltzmann Schemes for Heat Transfer and Phase Change in Turbulent Flows”. MA thesis. Delft University of Technology, 2023.

[120] Hong Wu, Jiao Wang, and Zhi Tao. “Passive heat transfer in a turbulent channel flow simulation using large eddy simulation based on the lattice Boltzmann method framework”. In: *International journal of heat and fluid flow* 32.6 (2011), pp. 1111–1119.

[121] Heng Xiao and Paola Cinnella. “Quantification of model uncertainty in RANS simulations: A review”. In: *Progress in Aerospace Sciences* 108 (2019), pp. 1–31.

[122] Qingang Xiong et al. “Efficient 3D DNS of gas–solid flows on Fermi GPGPU”. In: *Computers & Fluids* 70 (2012), pp. 86–94.

[123] Ao Xu and Bo-Tao Li. “Multi-GPU thermal lattice Boltzmann simulations using OpenACC and MPI”. In: *International Journal of Heat and Mass Transfer* 201 (2023), p. 123649.

[124] Dazhi Yu, Renwei Mei, and Wei Shyy. “A multi-block lattice Boltzmann method for viscous fluid flows”. In: *International journal for numerical methods in fluids* 39.2 (2002), pp. 99–120.

[125] Rui Zhang et al. “Large-eddy simulation of wall-bounded turbulent flow with high-order discrete unified gas-kinetic scheme”. In: *Advances in Aerodynamics* 2 (2020), pp. 1–27.

[126] Pengbo Zhao et al. “Modeling the mushy zone during the melting process under Neumann boundary condition using the improved enthalpy-porosity method”. en. In: *Numerical Heat Transfer, Part A: Applications* 78.8 (July 2020). ISSN: 1040-7782. DOI: `10.1080/10407782.2020.1793540`. URL: `https://www-tandfonline-com.tudelft.idm.oclc.org/doi/epdf/10.1080/10407782.2020.1793540?needAccess=true&role=button` (visited on 04/16/2023).

[127] Yang Zhiyin. “Large-eddy simulation: Past, present and the future”. In: *Chinese Journal of Aeronautics* 28.1 (Feb. 2015), pp. 11–24.

[128] De-yu Zhong, Guang-qian Wang, and Ming-Xi Zhang. “Kinetic Equation for Stochastic Vector Bundles”. In: *arXiv preprint arXiv:2211.16754* (2022).

[129] Congshan Zhuo and Chengwen Zhong. “LES-based filter-matrix lattice Boltzmann model for simulating fully developed turbulent channel flow”. In: *International Journal of Computational Fluid Dynamics* 30.7-10 (Nov. 2016). Publisher: Taylor & Francis, pp. 543–553. ISSN: 1061-8562. DOI: `10.1080/10618562.2016.1254777`. URL: `https://doi.org/10.1080/10618562.2016.1254777`.

[130] Congshan Zhuo and Chengwen Zhong. “LES-based filter-matrix lattice Boltzmann model for simulating turbulent natural convection in a square cavity”. In: 42 (Aug. 2013), pp. 10–22. ISSN: 0142727X. DOI: `10.1016/j.ijheatfluidflow.2013.03.013`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0142727X13000702`.

[131] Congshan Zhuo, Chengwen Zhong, and Jun Cao. “Filter-matrix lattice Boltzmann model for incompressible thermal flows”. In: *Physical Review E* 85.4 (Apr. 2012), p. 046703. ISSN: 1539-3755. DOI: `10.1103/PhysRevE.85.046703`. URL: `https://link.aps.org/doi/10.1103/PhysRevE.85.046703`.

A
Temperature Fluctuation in DDF-LBM

As was mentioned in Sec. 6.2, a larger relative temperature or enthalpy difference resulted in a reduced relative fluctuation of the local temperature field. To illustrate the reason behind this result, we consider a simplified system with two adjacent nodes in a D3Q7 scheme (see Fig. A.1). Both nodes have an equal sensible enthalpy $h_0 = 500$ lJ/lm, initialized with the equilibrium distribution. The nodes have different velocities, such that:

$$\boldsymbol{u_1} = \begin{pmatrix} u_1 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{u_2} = \begin{pmatrix} 0 \\ u_2 \\ 0 \end{pmatrix}, \tag{A.1}$$

where

$$\boldsymbol{x}_1 = (x, y, z), \quad \boldsymbol{x}_2 = (x, y-1, z). \tag{A.2}$$

Furthermore, it is assumed that all boundaries of the two-node system satisfy Neumann conditions, such that streaming from the boundaries does not change the initial system's properties.

We simplify by assuming a liquid fraction of $f_l = 0$ everywhere on the domain. The enthalpy equilibrium distribution of Eq. 3.15 then reduces to

$$g_i^{\text{eq}} = \begin{cases} \omega_i h & i = 0 \\ \omega_i h \left[1 + \frac{\boldsymbol{c_i} \cdot \boldsymbol{u}}{c_s^2}\right] & i \neq 0 \end{cases}, \tag{A.3}$$

We can now insert the velocities of Eq. A.1 into the equilibrium distribution of Eq. A.3 to obtain the following expressions for the thermal distributions at nodes 1 and 2:

$$g^1 = \begin{pmatrix} \omega_0 h \\ \omega_1 h(1 + \frac{u_1}{c_s^2}) \\ \omega_2 h(1 - \frac{u_1}{c_s^2}) \\ \omega_3 h \\ \omega_4 h \\ \omega_5 h \\ \omega_6 h \end{pmatrix}, \quad g^2 = \begin{pmatrix} \omega_0 h \\ \omega_1 h \\ \omega_2 h \\ \omega_3 h(1 + \frac{u_2}{c_s^2}) \\ \omega_4 h(1 - \frac{u_2}{c_s^2}) \\ \omega_5 h \\ \omega_6 h \end{pmatrix}. \tag{A.4}$$

We will now perform a streaming step. Because Neumann boundary conditions are assumed, the only new information that flows into node 1 comes from node 2 and vice versa. After streaming, we therefore obtain the following distributions for node 1 and 2:

$$g^1_{post} = \begin{pmatrix} \omega_0 h \\ \omega_1 h(1 + \frac{u_1}{c_s^2}) \\ \omega_2 h(1 - \frac{u_1}{c_s^2}) \\ \omega_3 h(1 + \frac{u_2}{c_s^2}) \\ \omega_4 h \\ \omega_5 h \\ \omega_6 h \end{pmatrix}, \quad g^2_{post} = \begin{pmatrix} \omega_0 h \\ \omega_1 h \\ \omega_2 h \\ \omega_3 h(1 + \frac{u_2}{c_s^2}) \\ \omega_4 h \\ \omega_5 h \\ \omega_6 h \end{pmatrix}. \tag{A.5}$$

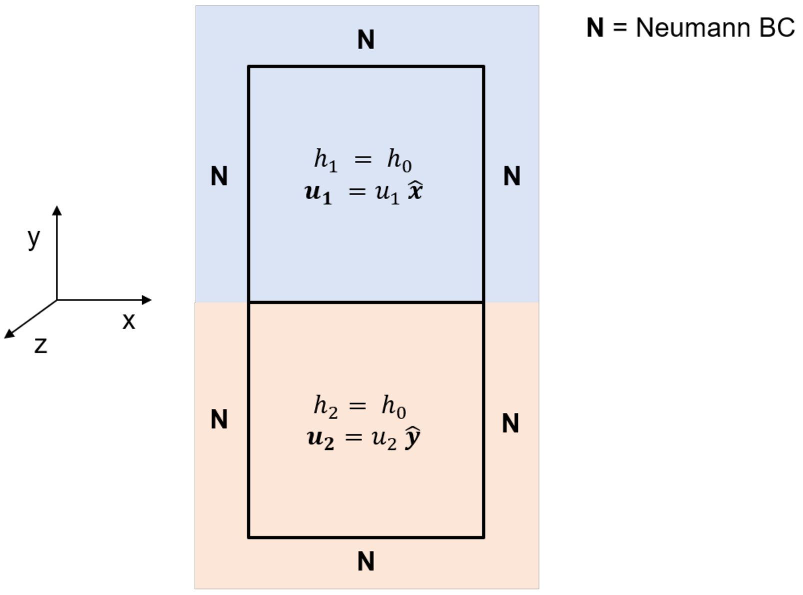

N = Neumann BC
N
N
N
N
N
N
$h_1 = h_0$
$\boldsymbol{u_1} = u_1\,\hat{\boldsymbol{x}}$
$h_2 = h_0$
$\boldsymbol{u_2} = u_2\,\hat{\boldsymbol{y}}$
y
x
z

Figure A.1: Initial situation of two-node system with constant sensible enthalpy $h_1 = h_2 = h_0$. The nodes have different velocities $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$. Neumann conditions are assumed to simulate adjacent nodes with the same initial distribution functions.

Now, the post-stream enthalpy of the two nodes is calculated by summing the components of the distribution functions. This leads to

$$h_{1,post} = \sum_i g^1_{i,post} = \sum_i \omega_i h + \frac{u_2}{c_s^2} = h_0 + \frac{u_2}{c_s^2}, \tag{A.6}$$

$$h_{2,post} = \sum_i g^2_{i,post} = \sum_i \omega_i h + \frac{u_1}{c_s^2} = h_0 + \frac{u_1}{c_s^2}. \tag{A.7}$$

It is found that the enthalpies in node 1 and 2 are increased by $\frac{u_2}{c_s^2}$ and $\frac{u_1}{c_s^2}$, respectively, which is the equivalent of a temperature fluctuation. Such fluctuations occur when the two adjacent nodes have different velocities along the line that connects them, or equivalently, in regions with high vorticity. This explains why fluctuations were observed after the first few streaming steps of the simulations in Sec. 6.2. Although the grid was initialized with a constant enthalpy, the presence of the turbulent flow field caused significant fluctuations in the thermal field. The fact that vorticity is a source of fluctuation also explains why the fluctuations in Figs. 6.2-6.3 resembled the eddy shapes.