Developing a GPU-Accelerated Filter-Matrix Lattice Boltzmann Multiphysics Tool for the Transient Thermal-Hydraulics and Neutronics of a Molten Salt Fast Reactor Core

Master Thesis

Thomas Pijls

Delft University of Technology

TU Delft

Developing a GPU-Accelerated Filter-Matrix Lattice Boltzmann Multiphysics Tool for the Transient Thermal-Hydraulics and Neutronics of a Molten Salt Fast Reactor Core

Master Thesis

by

Thomas Pijls

Student Name	Student Number
Thomas Pijls	4869354

to obtain the degree of Master of Science at the Delft University of Technology,
to be defended publicly on Friday June 27, 2025 at 14:00.

Thesis committee:	dr.ir. M. Rohde	TU Delft, Supervisor
	prof.dr.ir. Jan Leen Kloosterman	TU Delft
	dr. Erik van der Kolk	TU Delft
Program:	MSc Applied Physics	
Faculty:	Faculty of Applied Sciences, Delft	

TU Delft

Highlights

The most important contributions from this research project are:

- *The development of a novel filter-matrix lattice Boltzmann method (FM-LBM) algorithm for the simulation of the thermal-hydraulics, neutronics, and precursor transport in a MSFR reactor core. This algorithm shows for the first time that the full multiphysics can be simulated in 1 single FM-LBM algorithm.*
- *Increasing the computational performance of the FM-LBM algorithm using GPU-acceleration. The algorithm shows a substantial computational performance increase compared to CPU implementations, however it under performs compared to similar GPU-accelerated LBM algorithms in literature.*
- *The successful implementation of the predictor-corrector quasi-static method in the FM-LBM algorithm to simulate transient MSFR reactor core behaviour. The results agree well with the established Tiberga benchmark.*

Abstract

This thesis presents the development of a novel GPU-accelerated multiphysics simulation framework for modeling the transient behavior of a Molten Salt Fast Reactor (MSFR) core. The MSFR, a Generation IV nuclear reactor design, operates with a liquid fuel mixture dissolved in molten salt, allowing for higher fuel utilization, improved inherent safety, and the potential for continuous reprocessing. However, the use of a circulating liquid fuel introduces a complex coupling between thermal-hydraulics, neutronics, and delayed neutron precursor transport, which traditional reactor simulation tools are not well-suited to handle. In this research, a simulation tool is developed specifically to simulate the coupled multiphysics of a molten salt fast reactor core. The equations describing fluid flow, temperature, neutron flux, and precursor transport will be solved using the lattice Boltzmann method (LBM) with the novel filter-matrix (FM) collision operator.

The FM-LBM formulation used in this work provides a more stable numerical scheme than classical single- or multi-relaxation time LBM operators. It effectively filters out non-physical higher-order moments introduced during discretization, enabling accurate simulations at high Prandtl and Schmidt numbers. This is essential for simulating the low-diffusivity, low-viscosity characteristics of molten salts and the slow diffusion of delayed neutron precursors. For the neutronics, the multigroup neutron diffusion equation is implemented using the filter-matrix approach. A two-domain approach with different timescales is adopted to simulate the short timescale neutronics separate from the longer timescale thermal-hydraulics and precursor transport. To model transient reactor behavior efficiently, the predictor-corrector quasi-static method (PCQSM) is integrated into the LBM framework.

To ensure practical usability, the entire simulation framework is implemented in the Julia programming language with the CUDA backend for GPU acceleration. This allows the simulation tool to leverage the massive parallelism of modern GPUs, significantly reducing computation time without compromising accuracy. Benchmarking against the established Tiberga benchmark suite demonstrates that the tool achieves excellent agreement with reference solutions for steady-state cases and that it is able to reproduce the trends of the transient case.

The results of this thesis show that the developed GPU-accelerated FM-LBM tool is capable of accurately capturing the coupled thermal-hydraulics, neutronics, and precursor transport of a simplified MSFR core for both steady-state and transient cases. This work offers a foundation for future research into more detailed three-dimensional geometries, extended physics (Doppler shift, freezing and melting of salt, turbulence modeling), and the implementation of the neutron transport equation.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviation	Definition
ADE	advection-diffusion equation
FM-LBM	filter-matrix lattice Boltzmann method
GPU	graphical processing unit
LBM	lattice Boltzmann method
MSFR	molten salt fast reactor
MSR	molten salt reactor
NDE	neutron transport equation
NTE	neutron transport equation

Symbol	Definition	Unit
c_s	lattice speed of sound	$-$
$\boldsymbol{c}_i$	discrete lattice velocity	$\mathrm{m\,s^{-1}}$
C	precursor density	$\mathrm{m^{-3}}$
C_p	specific heat capacity	$\mathrm{J\,m^{-3}\,K^{-1}}$
D_g	neutron diffusion constant	m
D_p	precursor diffusion constant	$\mathrm{m^2\,s^{-1}}$
E	energy	J
$\boldsymbol{f}$	bodyforce	$\mathrm{N\,kg^{-1}\,m^{-3}}$
f_i	particle density distribution function	$-$
h	enthalpy	$\mathrm{J\,m^{-3}}$
h_i	enthalpy distribution function	$-$
k	multiplication factor	$-$
$n_{i,g}$	neutron distribution function	$-$
p	pressure	Pa
$p_{i,d}$	precursor concentration distribution function	$-$
P	power	W
q	heat source or sink	W
$\boldsymbol{r}$	space coordinate	m
t	time	s
T	temperature	K
$\boldsymbol{u}$	velocity	$\mathrm{m\,s^{-1}}$
v	velocity magnitude	$\mathrm{m\,s^{-1}}$

Greek symbol	Definition	Unit
α	thermal diffusivity	$\mathrm{m^2\,s^{-1}}$
α_k	momentum solution vector	$-$
β	delayed neutron fraction	$-$
β_k	temperature solution vector	$-$
β_{th}	thermal expansion coefficient	$\mathrm{K^{-1}}$
γ	volumetric heat transfer coefficient	$\mathrm{W\,m^{-3}\,K^{-1}}$
$\gamma_{k,g}$	neutron solution vector	$-$
$\delta_{k,d}$	delayed precursor solution vector	$-$
κ	thermal conductivity	$\mathrm{W\,m^{-1}\,K^{-1}}$

Greek symbol	Definition	Unit
λ	decay constant	s^{-1}
ν	kinematic viscosity	$\mathrm{m}^2\,\mathrm{s}^{-1}$
ν	average number of neutrons produced per fission	$-$
$\boldsymbol{\xi}$	particle velocity	$\mathrm{m}\,\mathrm{s}^{-1}$
ρ	density	$\mathrm{kg}\,\mathrm{m}^{-3}$
ρ	reactivity	$-$
σ	microscopic nuclear cross-section	m^2
$\boldsymbol{\sigma}$	stress tensor	Pa
Σ	macroscopic nuclear cross-section	m^{-1}
ϕ	scalar neutron flux	$\mathrm{m}^{-2}\,\mathrm{s}^{-1}$
φ	angular neutron flux	$\mathrm{m}^{-2}\,\mathrm{s}^{-1}$
χ	neutron spectrum	$-$
ω_i	weight function	$-$
Ω	collision operator	$-$
$\hat{\boldsymbol{\Omega}}$	angular direction	$-$

Subscript	Definition
adj	adjacent
d	delayed precursor family
eff	effective
f	fission
g	neutron energy group
N	neutronics domain
ref	reference
s	scattering
t	total removal
TH	thermal-hydraulics domain
tot	total

Superscript	Definition
d	delayed
eq	equilibrium
p	prompt

Symbol	Dimensionless number
Da	Damköhler
Gr	Grashof
Ma	Mach
Pr	Prandtl
Ra	Rayleigh
Re	Reynolds
Sc	Schmidt

1 Introduction

With the last two years being the warmest years on record (global air temperature anomalies of $+1.17\,^{\circ}\mathrm{C}$ and $+1.29\,^{\circ}\mathrm{C}$ compared to pre-industrial levels for 2023 and 2024, respectively [34]), it has become evident that the Earth's climate is warming due to human activity. A threshold of $+1.5\,^{\circ}\mathrm{C}$ is widely accepted as the upper limit of global warming beyond which we will risk severe and irreversible effects on Earth's climate [8]. Although many factors play a role in global warming, greenhouse gas emissions are responsible for the majority. Globally, electricity production accounts for 40% of global greenhouse gas emissions [3]. Although electricity production is already moving towards carbon-free sources such as wind and solar power, their weather- and environmental-dependent output endangers the stability of the power supply. Nuclear power is becoming a more attractive option for reliable, sustainable energy production, since it is a carbon-free, on-demand power source.

To address current and past concerns about nuclear power generation, the Generation IV International Forum (GIF) was established in 2001 as a collaborative research initiative with the aim of developing nuclear reactor technologies with improved reliability, sustainability, and safety [15]. Six reactor designs have been selected by the GIF for further research - one of which is the Molten Salt Reactor, the subject of this research.

1.1. The Molten Salt Fast Reactor

The Molten Salt Reactor (MSR) is a family of nuclear fission reactors characterized by the use of a molten salt which functions as both the fuel and the coolant of the system [16, 31]. Because of the use of a liquid fuel, molten salt reactors offer improved safety, sustainability, and waste management compared to conventional solid fuel reactors. Several MSR designs have been proposed over the years, varying in salt composition (fluoride or chloride salts) and fuel use (uranium, plutonium, and/or thorium) to address the purpose of the reactor (energy production, high neutron flux, medical isotope production, breeding, etc.) [25, 31]. In this research, the Molten Salt Fast Reactor (MSFR) design will be studied.

As the name implies, the MSFR operates neutrons in the fast spectrum. Combined with the addition of fertile thorium isotopes to the salt mixture, this allows the MSFR to breed fissile uranium and the fast spectrum promotes actinide burnup [2]. Currently, the MSFR is still in the conceptual phase and numerical studies are performed to refine the design. The reference MSFR used in these simulations is a $3\ \mathrm{GW}_{th}$ reactor, using a salt mixture composed of 77.5 mol% lithium fluoride (LiF), 20 mol% thorium fluoride ($\mathrm{ThF_4}$), and 2.5 mol% uranium fluoride ($\mathrm{UF_4}$) [2, 17, 30]. The reactor is split in three circuits: the fuel circuit, the intermediate circuit, and the power conversion system [17, 30]. The fuel circuit, shown in figure 1.1, is regarded as reactor core.

The core shown in figure 1.1 is a single compact cylinder that measures $2.25\ \mathrm{m}$ high and $2.25\ \mathrm{m}$ in diameter. The fuel salt occupies a volume of $18\ \mathrm{m}^3$ and operates at a maximum temperature of $750\,^{\circ}\mathrm{C}$ [2]. The inner part of the core contains the core cavity (green region), where most of the fission occurs, as well as a fertile thorium blanket (red region) and reflectors (blue region). In the outer ring, 16 equally

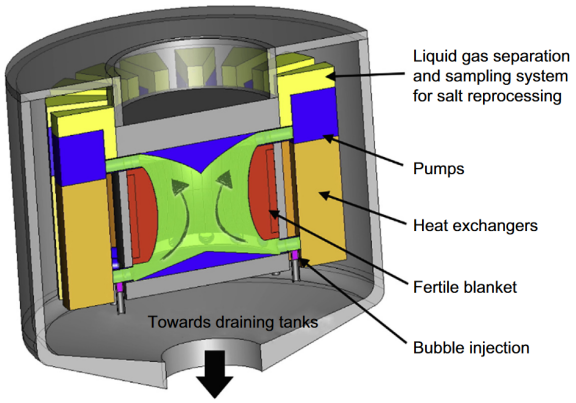


Figure 1.1: Schematic representation of the reference MSFR fuel circuit [2].

spaced pumps (blue region) and heat exchangers (orange region) are placed. These pumps and heat exchangers facilitate in the heat transfer from the core to the intermediate circuit. The salt circulates in roughly 3 to 4 seconds through the reactor [2, 30]. At the bottom of the reactor (pink region), helium bubbles are injected to capture insoluble gaseous and metallic fission products. These bubbles are captured at the top (yellow region) with some salt for reprocessing. This allows for continuous adjustments to the molten salt mixture without the need for reactor shutdowns [25, 31]. In addition, a passive safety system is incorporated, using a freeze plug at the bottom of the reactor core. This freeze plug is continuously cooled. In case of emergency, such as power loss or excessive reactor temperatures, the freeze plug will melt and the molten salt will drain into subcritical drainage tanks, where the fission reaction stops and the salt can gradually cool down [25, 36, 41]. The reactor core is enclosed by thick neutron reflectors and a $20\ \mathrm{cm}$ thick layer of boron carbide ($\mathrm{B_4C}$) to shield the outside from neutrons [2, 31].

In addition to the continuous reprocessing of fuel salt, and the passive salt-draining mechanism, the MSFR offers several other advantages. The MSFR exhibits a strong negative temperature feedback effect [25, 31] of $-5\ \mathrm{pcm\,K^{-1}}$ [2]. This is due to the strong density feedback effect, arising from the expansion of the salt, which has an immediate effect on the cross sections since there is no heat transfer delay between the coolant and the fuel. The Doppler feedback also contributes to the negative temperature feedback. Therefore, the MSFR can entirely be controlled through temperature regulation and helium bubbling, eliminating the need for control rods [2, 25, 35]. Additional safety and sustainability advantages include the following:

- The molten salt has a high boiling point and a low vapor pressure, allowing it to operate at low pressures. This reduces the risk of pipe ruptures and other high-pressure induced failures [25, 35].
- Continuous circulation and reprocessing of fuel leads to a more complete burnup and therefore to a more efficient fuel usage [31, 35].
- Next to the more efficient fuel usage offered by continuous processing, long-lived actinides remain longer in the reactor allowing them to burnup completely, significantly reducing the long-term radioactivity of the nuclear waste. The nuclear waste produced by MSFRs is only dangerous for a few centuries [9, 20]. Nuclear waste produced by conventional reactors can also be processed by MSFRs, reducing its dangerous levels of radioactivity from tens of thousands of years to to the same few centuries. This makes the MSFR a potential solution to the long-lived nuclear waste [9, 25].
- Thorium is approximately four times more abundant than uranium. Furthermore, while only 0.7% of all uranium is fissile $^{235}\mathrm{U}$, thorium is almost 100% made up of fertile $^{232}\mathrm{Th}$. Thorium is also already produced as a by-product of rare-earth mining, making fuel acquisition easier and more sustainable [9, 25].

The safety and sustainability advantages of the MSFR outlined in the previous paragraphs make it an

attractive technology for future nuclear power generation. However, the MSFR also comes with some significant challenges that need to be overcome. One of these challenges is the simulation of the coupled neutronics and thermal-hydraulics behaviour of the reactor. Because in MSFRs fuel and coolant are combined and fuel is liquid rather than solid, these physical fields have a much stronger interaction than in conventional solid-fuel reactors. Codes developed to simulate the multiphysics of conventional reactor designs fail to account for the stronger coupling and additional processes like precursor flow, necessitating the development of new, MSFR-specific, codes. Numerical studies have been performed, like [35], creating a simulation tool that combines the thermal-hydraulics and neutronics in a single model. The goal of this research is to contribute by developing a new simulation tool for MSFR reactor cores, where the thermal-hydraulics and neutronics are simulated in a one GPU-accelerated lattice Boltzmann model.

1.2. Existing Literature

In this research, a coupled thermal-hydraulics neutronics, also called multiphysics, code will be developed to simulate the multiphysics behaviour of the MSFR reactor core. This tool consists of three components: a lattice Boltzmann thermal-hydraulics code, a lattice Boltzmann neutronics code, and a coupling between these two codes. These multiphysics simulations are computationally expensive. The lattice Boltzmann method is well suited to be parallelized, so we will study if and how the lattice Boltzmann method can be parallelized on a graphical processing unit (GPU).

1.2.1. LBM for Thermal Fluid Simulation

The lattice Boltzmann method (LBM) is a numerical technique developed in the 1980s to simulate the flow of gasses and liquids. The flow is simulated by streaming a distribution function over discrete directions and colliding it at lattice points. He et al. showed that by implementing a double distribution function (DDF) flow and temperature can be simulated simultaneously [21]. Chatterjee added to the DDF by simulating enthalpy rather than temperature, therefore solving the energy equation [6]. Wang et al. expanded on the DDF approach by introducing a separate distribution function for the precursor density to simulate the precursor advection-diffusion equation [42]. These studies show that the momentum, energy, and precursor advection-diffusion equations can all be solved simultaneously in the same LBM framework, allowing for the simulation of the thermal-hydraulics using the LBM. In the book *The Lattice Boltzmann Method: Principles and Practice* by Krüger et al. a detailed overview of the fundamentals of the use of the LBM to simulate thermal fluids is given [23].

The collisions of the distribution function are modeled using a collision operator. In most studies, the single or multiple relaxation time collision operators are used; however, these operators suffer from instabilities or require the optimization of a large number of non-physical relaxation times. This research uses the filter-matrix lattice Boltzmann method (FM-LBM) collision operator, which was first introduced by Somers [32]. The FM-LBM has an improved stability by filtering out the non-physical terms that are introduced by the discretization of the governing equations. Zhuo et al. showed that FM-LBM can be used to simulate both flow and temperature in 2D and 3D simulations [52, 55]. Entes derived a FM-LBM collision operator analogous to the temperature collision operator to successfully model precursor flow in a thermal-fluid [12].

1.2.2. LBM for Neutronics Simulation

The LBM has also been employed to model the neutron transport equation and neutron diffusion equations. The multigroup neutron diffusion equation has been modeled similarly to other physical fields, by introducing a neutron distribution function for each group [42, 45]. Wang et al. showed that the multigroup SP_3 neutron transport equation can be simulated using the LBM by using two distribution functions for each energy group [43, 46]. Both neutronics models were in good agreement with neutronics benchmarks, showing that neutronics can be modeled using the LBM. However, in al these studies the single relaxation time collision operator was used. To improve the stability of the neutronics simulation, the neutron diffusion equation will be used and a FM-LBM collision operator will be derived analogous to other fields.

Computing transient neutronics problems comes with some difficulties, since the neutronics change on very short timescales. This requires the LBM timestep to become very small, which is computa-

tionally expensive and introduces instabilities. To circumvent this, the predictor-corrector quasi-static method (PCQSM) has been developed to accurately and efficiently solve transient neutronics problems. Developed by Dulla et al., the PCQSM factorizes the scalar neutron flux in a slow changing shape function and fast changing amplitude function [11]. The shape function is solved over big timesteps using a steady-solver, while the amplitude function is solved over small timesteps using the point-kinetics equations. This approach allows the LBM timesteps to remain relatively large, while the fast changing neutronics are captured by the amplitude function. Wang et al. [44] implemented the PCQSM in a neutron diffusion LBM solver which showed good agreement with the neutronics benchmarks used. In this research, the shape function was solved using the LBM and the amplitude function using the point kinetics equations.

1.2.3. GPU Accelerated LBM Simulation Techniques

The most expensive operation of the LBM algorithm is the collision, which is a local operation that only requires the information on one node. This lends the LBM well to parallelization, making it suitable for GPU parallelization. Li et al. showed how significant speed-ups can be achieved by parallelizing the LBM on the OpenGL graphics API. Further advancements have been made on GPU-accelerated LBM algorithms by Habich [19] and Tölke [38] who demonstrated how the use of shared memory, quickly accessible short lived memory on the GPU, can be used to accelerate the computations. Delbosc et al. [7] and Tran et al. [39] showed that by optimizing the data structuring the memory coalescence is improved and further speed-ups are achieved. Wang et al. [44] showed that even greater speed-ups can be reached by using a multi-GPU approach, with the greatest speed-ups observed for larger grids.

1.2.4. Multiphysics Simulation Tools for MSFRs

Several studies have developed multiphysics simulation tools for MSFR reactor cores. Wang et al. [42] developed a unified thermal-hydraulics and neutronics simulation tool using the LBM framework, where disitribution functions for the precursor density and neutron flux were introduced. This simulation showed how the multiphysics of a MSFR core can successfully be simulated in a LBM-only code. Since all fields are simulated using the LBM framework, data exchange between the fields is relatively straightforward, allowing for the easy implementation of couplings between fields. Other studies have developed similar multiphysics tools, however these tools often use software packages such as COMSOL [4] and OpenFOAM [14, 22] for their simulations.

1.3. Research Goals

The goal of this thesis is to develop a multiphysics tool dedicated to molten salt fast reactor cores that simulates the thermal-hydraulics and neutronics in a single LBM code base that is able to perform steady-state as well as transient simulations. This is done by developing a GPU-accelerated FM-LBM algorithms for the thermal-hydraulics as well as the neutronics in the Julia-CUDA framework, as well as a coupling between these two solvers. The following research questions will be answered during the development process:

1. ***Steady-state FM-LBM model development:*** *How can the Julia-CUDA framework be used to develop a GPU-accelerated FM-LBM algorithm that accurately models the steady-state thermal-hydraulics, precursor transport, and neutronics in a MSFR reactor core?*
    - *How can the DDF approach be used to model the velocity, temperature, and precursor fields in a single LBM algorithm?*
    - *How can the neutron diffusion equation be accurately modeled using the FM-LBM algorithm? How can the neutronics solver be coupled to the thermal-hydraulics solver to simulate accurate behaviour?*
    - *How can the flexibility of the Julia-CUDA framework be used to implement the GPU optimization techniques from the literature to improve the performance of the LBM algorithm?*
    - *How does the steady-state FM-LBM model compare to established steady-state benchmark studies?*
2. ***Transient FM-LBM model development:*** *How does the steady-state FM-LBM algorithm needs to be changed to accurately model transient behaviour of a MSFR reactor core?*

- *How does the thermal-hydraulics FM-LBM solver need to be altered to simulate transient situations?*
- *How does the neutronics FM-LBM solver need to be altered to simulate transient situations? Will the implementation of the predictor-corrector quasi-static method be sufficient, or are additional methods required?*
- *How does the transient FM-LBM model compare to established transient benchmark studies?*

3. ***Computational performance:*** *What is the computational performance of the multiphysics simulation tool?*
   - *Can we achieve significant computational performance improvements of the GPU-accelerated simulation tool compared to other codes using a serial implementation?*
   - *How does the GPU-accelerated simulation tool compare to other GPU-accelerated tools from the literature?*

1.4. Thesis Outline

This thesis is structured as follows. Chapter 2 covers the theoretical background of this project, discussing the principles of thermal-hydraulics, nuclear reactor physics, the neutron transport and diffusion equations, the interactions between the physical fields, kinetic theory, and GPU programming. Chapter 3 discusses the numerical methods used, first explaining the fundamentals of the LBM, as well as the FM-LBM algorithm for each physical field and the implementation of boundary conditions. Afterwards, the lattice conversion, scheme to solve the coupled system, as well as the implementation of the PCQSM will be discussed. Lastly, the GPU-acceleration of the FM-LBM algorithm will be outlined. Chapter 4, 5, and 6 compare the results simulation tool against the Tiberga benchmark case, which is used to validate the multiphysics tool. Chapter 4 covers the validation of the steady-state, single, uncoupled physics, chapter 5 covers the validation of the steady-state, coupled physics, and chapter 6 covers the validation of the transient coupled physics simulations. Chapter 7 concludes the report, summarizes the findings and giving recommendations for future research.

2

Theory

In the core of a molten salt reactor, the liquid salt acts as both a coolant and a fuel. This leads to a stronger coupling between the thermal-hydraulics and neutronics compared to conventional, solid-fuel reactors. In this chapter, the relevant physics and GPU theory will be discussed for this research. In section 2.1 the physics describing thermal fluids will be presented. Section 2.2 will explain the general theory of nuclear reactors while section 2.3 will discuss the physical equations describing neutron transport. The interactions between fields will be discussed in section 2.4 and kinetic theory will be presented in section 2.5. Lastly, section 2.6 will present the fundamentals of GPU programming.

2.1. Thermal-Hydraulics

Thermal-hydraulics concerns the flow and heat transfer in a liquid. In this section, the physical formulas describing flow and heat transfer through a liquid will be introduced, as well as the relevant dimenionsless numbers and an introduction to kinetic theory.

2.1.1. Fluid Dynamics

The continuum approximation is used to mathematically describe the behaviour of fluids. The continuum approximation is based on the assumption that the fluid can be described as a continuous mass distribution [29]. This means that the discrete nature of atoms and their interactions can be ignored. Under the continuum approximation, the fluid is described by macroscopic variables such as density, velocity, and pressure. Consequently, conservation laws can be derived for isolated physical systems in continuum physics.

In continuum physics, mass cannot be created or destroyed. This means that a change in mass in a control volume can only occur due to mass flowing in or out of the volume. This leads to the first conservation law in continuum physics, the so called continuity equation

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

where ρ denotes the density of the control volume and $\boldsymbol{u}$ the fluid velocity [29]. Similarly, the momentum inside a control volume can only change due to momentum flowing in or out, stresses at the boundary, or a body force acting on the control volume. This yields the Cauchy momentum equation, the second conservation law

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} = \frac{1}{\rho}\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{f} \tag{2.2}$$

where $\boldsymbol{\sigma}$ denotes the Cauchy stress tensor and $\boldsymbol{f}$ denotes the bodyforce. When dealing with Newtonian fluids, the Cauchy stress tensor can be simplified since the viscosity is independent of the applied shear stresses. The Cauchy momentum equation can be further simplified for incompressible flows, allowing

the material derivative ($\frac{\mathrm{D}\rho}{\mathrm{D}t}$) to be set to zero. Applying these assumptions, we arrive at the Navier-Stokes Equations (NSE) for incompressible Newtonian fluids

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{f} \tag{2.3}$$

where ν denotes the kinematic viscosity of the fluid. The NSE forms a complete set of equations to describe incompressible Newtonian fluids [29].

2.1.2. Temperature Dynamics

Similarly to mass and momentum, a conservation law can be formulated for energy. The amount of energy in a control volume can only change as a result of in or out flow of energy (by convection and diffusion) or by the production or destruction of energy. By applying Fourier's law of thermal conduction the heat equation can be derived

$$\rho C_p \frac{\partial T}{\partial t} + \rho C_p \, \boldsymbol{u} \cdot \boldsymbol{\nabla} T = \boldsymbol{\nabla} \cdot \kappa \boldsymbol{\nabla} T + q \tag{2.4}$$

where κ denotes the thermal conductivity, C_p the specific heat capacity, and q the heat source or sink term. The terms of the heat equation describe, from left to right, the rate of change of temperature, convective transport, diffusive transport, and heat production or destruction. For constant thermal conductivity, κ can be taken out of the gradient of the first term on the left-hand side.

To accommodate the modeling of freezing effects in further research, it is more convenient to model the heat in terms of enthalpy rather than temperature. For systems with negligible heat capacity thermal variations, the heat equation can easily be rewritten to the enthalpy equation by defining the enthalpy $h = \rho C_p T$ and the thermal diffusivity $\alpha = \kappa / \rho C_p$, resulting in:

$$\frac{\partial h}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} h = \alpha \nabla^2 h + q \tag{2.5}$$

In this research, heat transport is modeled using the enthalpy equation.

2.1.3. Dimensionless Numbers

In the study of fluid dynamics, the behaviour of a system is largely determined by the ratio of different physical processes (e.g. momentum transport by convection versus viscosity) rather than their absolute physical size. These ratios can be derived by performing a dimensional analysis and give a fundamental insight into the behaviour of a system. A dimensional analysis will be performed on the NSE from equation 2.3 by introducing the following dimensionless variables

$$\tilde{\boldsymbol{r}} = \frac{\boldsymbol{r}}{L}, \qquad \tilde{t} = \frac{t}{T}, \qquad \tilde{\rho} = \frac{\rho}{\rho^\star}, \qquad \tilde{\nu} = \frac{\nu}{\nu^\star}, \qquad \tilde{\boldsymbol{u}} = \frac{\boldsymbol{u}}{U} = \frac{\boldsymbol{u}T}{L}, \qquad \text{etc.} \tag{2.6}$$

where L, T, $\rho^\star$, $\nu^\star$, and U are the characteristic length, time, density, viscosity, and velocity scales of the system. With these dimensionless variables, the NSE can be rewritten to

$$\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + \left(\tilde{\boldsymbol{u}} \cdot \tilde{\boldsymbol{\nabla}}\right)\tilde{\boldsymbol{u}} = \frac{1}{\tilde{\rho}}\tilde{\boldsymbol{\nabla}}\tilde{p} + \frac{1}{\mathrm{Re}}\tilde{\nu}\tilde{\boldsymbol{\nabla}}^2\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{f}} \tag{2.7}$$

where the tilde notation is used for all dimensionless variables and operators. The dimensionless Reynolds number is also introduced which is defined as

$$\mathrm{Re} = \frac{UL}{\nu^\star} \tag{2.8}$$

The relevance of the Reynolds number can be demonstrated when we take a closer look at the dimensionless NSE. When the Reynolds number is low, the viscous term (second term on the right side)

dominates over the convective term (second term on the left side). Since the viscous term is linear, at low Reynolds number the flow is smooth and stable, which is also known as laminar flow. For high Reynolds number, the non-linear convective term dominates, resulting in unstable and chaotic flow, which is also known as turbulence.

This shows the importance of dimensionless numbers and how they can be applied in practice. For other equations and systems different dimensionless numbers can be derived, the most important ones for this research are outlined in the table below.

Table 2.1: Dimensionless numbers with their definitions and physical interpretation relevant to this research.

Dimensionless number	Definition	Physical interpretation
Reynolds	$\mathrm{Re} = \frac{UL}{\nu}$	Ratio between inertial and viscous forces acting on the fluid. Determines if the flow is laminar or turbulent.
Prandtl	$\mathrm{Pr} = \frac{\nu}{\alpha}$	Ratio between the viscosity and thermal conductivity of a fluid.
Schmidt	$\mathrm{Sc} = \frac{\nu}{D}$	Ratio between the viscosity and mass diffusion of a fluid.
Rayleigh	$\mathrm{Ra} = \frac{\beta_{th} L^3 g \Delta T}{\nu \alpha}$	Ratio between the buoyancy forces and viscous and thermal diffusion effects.

2.2. Nuclear Reactor Physics

A nuclear reactor is an apparatus that can sustain nuclear fission [10]. Nuclear fission is a reaction where an atomic nucleus splits into two or more smaller nuclei. Fission releases energy and also often gamma rays and neutrons. Fission can occur spontaneously, however this rate is often slow. Fission can also be induced by hitting nuclei with neutrons. When a nucleus absorbs a neutron a heavy, unstable nucleus is created which will undergo fission. This is the basic principle used in a nuclear reactor core to induce nuclear fission and sustain a fission chain reaction.

The chance that a neutron interacts with a nucleus is characterized by the microscopic nuclear cross section, σ_x. For bulk materials the chance is described by the macroscopic cross section Σ_x, which is the product of the microscopic nuclear cross section σ_x and the isotope number density N. In the rest of this report, the term 'nuclear cross section' refers to the macroscopic nuclear cross section. The nuclear cross section depends on the medium, neutron energy (kinetic energy), and temperature. There are several neutron-nucleus interactions possible, such as scattering, fission, and absorption.

Fission is a type of absorption that results in an unstable nucleus that undergoes fission. Whether absorption occurs highly depends on the neutron velocity. If the neutron energy matches an energy level in the nucleus, absorption is greatly increased, resulting in so called resonance peaks. Due to thermal motion of the nucleus, these resonance peaks broaden and decrease in magnitude with increasing temperature, this is called the Doppler effect. Temperature changes will also influence the nuclear cross section due to isotope number density changes due to volume expansion or shrinkage.

In the reactor core, the neutrons produced in previous fission reactions are used to induce new fission reactions. For safe reactor operations, one wants to track the increase or decrease of neutrons over time, for which the multiplication factor k is used. The multiplication factor k is defined as the number of neutrons produced in the current reactor period divided by the number of neutrons produced in the previous reactor period. The reactor period is defined as the time between fission events. When $k = 1$ the reactor core is critical and the number of neutrons in the core will be constant. A reactor core is subcritical when $k < 1$ and supercritical when $k > 1$. For safe reactor operations, the number of neutrons in the core should be constant, so all nuclear reactors are designed to maintain a critical multiplication factor. Since small deviations from criticality will results in big power changes due to the short timescale of nuclear fission, deviations are measured by the reactivity ρ given by

$$\rho = \frac{k-1}{k} \tag{2.9}$$

which is usually measured in pcm (per-cent-mille, 10^{-5}). For instance, a reactivity of $500\ \mathrm{pcm}$ corresponds to a multiplication factor of 1.005.

The number of neutrons emitted from a fission reaction vary and their energy varies as well. The average number of neutrons emitted per fission event is represented by ν, and the energy spectrum by $\chi(E)$. Most neutrons appear instantaneously ($< 10^{-14}$ s after fission occurs. However, some neutrons are produced by radioactive fission products. Their emission rate varies from milliseconds to minutes. The instantaneous neutrons are called prompt neutrons while the neutrons emitted by the fission products are called delayed neutrons. When a nuclear reactor reaches criticality on only prompt neutrons, the reactor is prompt critical. This makes the reactor hard to control, since the reactor period is in the order of 10^{-14} s, meaning that a small change in reactivity will quickly get amplified before an external reaction (for example, pulling out/dropping in a control rod) can come into effect. Due to this, all nuclear reactors are designed to avoid prompt criticality and use delayed neutrons to reach criticality. The inclusion of delayed neutrons lengthens the reactor period to seconds, enabling practical reaction times for operators.

To analyze the delayed precursors, they are often grouped into families with similar half-life. The fraction of neutrons that appear in delayed precursor family d is represented by β_d. The total number of neutrons that appear in the delayed precursors are represented by $\beta_{tot} = \sum_d \beta_d$. The average half-life of each precursor family is given by λ_d. In molten salt reactors, the delayed precursors are dissolved in the salt. This means that the flow of the salt will affect the neutron distribution, since the delayed precursors will emit their neutrons at a different place than where the delayed precursors were created.

2.3. Neutron Transport and Diffusion

Understanding and predicting the behavior and number of neutrons is essential to control a nuclear reactor. The neutron transport equation (NTE) gives the most complete description of neutrons. The NTE balances the mechanisms of neutron creation and removal for neutrons with energy E, traveling in direction $\mathbf{\Omega}$, located in an infinitesimal volume element $\boldsymbol{r}$ at a time t [10]. The NTE is given by

$$\frac{1}{v}\frac{\partial \varphi}{\partial t} + \mathbf{\Omega}\cdot\boldsymbol{\nabla}\varphi + \Sigma_t(\boldsymbol{r},E)\varphi(\boldsymbol{r},E,\mathbf{\Omega},t) = \int_{4\pi}\mathrm{d}\mathbf{\Omega}'\int_0^\infty \mathrm{d}E'\,\Sigma_s(E'\to E,\mathbf{\Omega}'\to\mathbf{\Omega})\varphi(\boldsymbol{r},E',\mathbf{\Omega}',t) + S_f(\boldsymbol{r},E,\mathbf{\Omega},t) \tag{2.10}$$

where φ is the angular neutron flux, a measure of the number of neutrons crossing a unit of area per unit of time. The first term on the left is the rate of change of number of neutrons, the second term is the leakage of neutrons into or out of the volume element, and the third term the removal of neutrons with energy E in the volume element, either by absorption or collision. The first term on the right is the scattering of neutrons with all energies and directions to energy E and direction $\mathbf{\Omega}$ and the second term is the production of neutrons by fission. The fission term is given by

$$S_f(\boldsymbol{r},E,\mathbf{\Omega},t) = \frac{\chi(E)}{4\pi}\int_{4\pi}\mathrm{d}\mathbf{\Omega}'\int_0^\infty \mathrm{d}E'\,\nu(E')\Sigma_f(E')\varphi(\boldsymbol{r},E',\mathbf{\Omega}',t) \tag{2.11}$$

The fission term computes the total number of neutrons that are produced by fission from neutrons of all energies and directions. The prefactor gives the fraction of neutrons traveling with energy E and direction $\mathbf{\Omega}$.

Equation 2.10 must be solved for all possible energies and directions, which results in an infinite number of coupled equations. To solve the NTE, the neutron energies are discretised into energy bands. Neutrons are grouped by their energy bands, resulting in so called energy groups, denoted by E_g. For each energy group share the same neutronics parameters such as cross-sections and diffusion constants.

The NTE can be further simplified by discretising the angular directions of the neutrons. This is called a discrete ordinate treatment of angle and gives rise to the S_N equations, where N represents the number of dicretised angular directions. Alternatively, the angular directions can be expanded using spherical

harmonics. In one-dimension, this expansion corresponds to an expansion in Legendre polynomials in $\cos\theta$, resulting in the P_N equations [10].

The NTE gives the most complete description of neutron transport, however it is also computationally expensive, even with the use of aforementioned discretisations [10]. A less computationaly intensive equation can be derived by sacrificing some accuracy. By combining the P_1 approximation with the assumption that neutron sources are isotropic and that the neutron current density changes much slower compared to the neutron collision frequency, the neutron diffusion equation (NDE) can be derived from the NTE. In the NDE, the neutron diffusion coefficient D_g is introduces, which depends on the total cross-section, the scattering cross-section and the average scattering cosine $\bar{\mu}_0$. Additionally the angular neutron flux φ is replaced by the scalar neutron flux ϕ. The NDE reduces the number of equations significantly and describes most media very well. In regions with large neutron density gradients, like sources and boundaries, the NDE become less accurate. Considering discretised energy groups, the multi-group NDE is written for each energy group g as

$$\frac{1}{v_g}\frac{\partial \phi_g}{\partial t} - \boldsymbol{\nabla}\cdot(D_g\boldsymbol{\nabla}\phi_g) = -\Sigma_{t,g}\phi_g + \chi_g\sum_{g'}(\nu\Sigma_f)_{g'}\phi_{g'} + \sum_{g'}\Sigma_{s,g'\to g}\phi_{g'} \tag{2.12}$$

Here the first term on the left side is the rate of neutron density change. The second term is the diffusion term. The first term on the right side is the removal of neutrons, the second term the production due to fission, and the third term the scattering of neutrons from groups g' to group g.

A vacuum boundary condition is enforced by setting the incoming neutron flux to zero. In the NTE this can easily be implemented since the angular neutron flux incorporates a direction. However, for the NDE this condition is harder to implement since the scalar neutron flux does not carry a direction. It can be derived that a vacuum boundary condition can be enforced for the NDE by requiring the scalar flux to be zero at an extrapolated boundary $\tilde{x}_{boundary,g}$. This extrapolated boundary is proportional to the neutron diffusion constant

$$\tilde{x}_{boundary,g} = x_{boundary,g} + 2.1312D_g \tag{2.13}$$

A visualization of the diffusion approximation of the vacuum boundary condition is shown in figure 2.1.

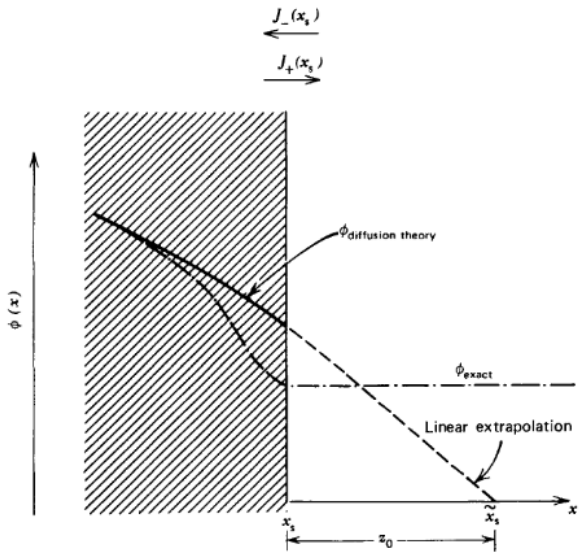


Figure 2.1: The exact and diffusion approximation of the neutron flux near a vacuum boundary [10].

Up to this point only prompt neutrons were considered. However, as was stated earlier, the delayed neutrons also play an important role. To describe the delayed neutrons, a physical description of the

delayed precursors is needed. The advection-diffusion equation of the concentration of precursors of family d is given by

$$\frac{\partial C_d}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u} C_d) - \boldsymbol{\nabla} \cdot (D_p \boldsymbol{\nabla} C_d) = -\lambda_d C_d + \beta_d \sum_g (\nu \Sigma_f)_g \phi_g \tag{2.14}$$

Here the first term on the left side is the rate of change of precursor concentration. The second term is the convective transport and the third term the diffusive transport of precursors. The first term on the right side is the natural decay of precursors while the second term represents the production of precursors due to fission. Since the decay of precursors produces neutrons, an extra source term needs to be added to equation 2.12

$$\frac{1}{v_g}\frac{\partial \phi_g}{\partial t} - \boldsymbol{\nabla} \cdot (D_g \boldsymbol{\nabla} \phi_g) = -\Sigma_{t,g}\phi_g + \chi_g^p (1-\beta_{tot}) \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \chi_g^d \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g' \to g} \phi_{g'} \tag{2.15}$$

Where the fourth term on the right side is the neutron production due to precursor decay. The second term on the right has been adjusted to account for the fission neutron loss to precursor production ($\beta_{tot} = \sum_d \beta_d$).

2.3.1. Steady-State Neutronics: Reactor Criticality Calculation

For safe operation, reactors are designed to be critical. Criticality implies that the number of neutrons does not change over time, so in equations 2.14 and 2.15 the time-dependent term can be eliminated

$$\boldsymbol{\nabla} \cdot (\boldsymbol{u} C_d) - \boldsymbol{\nabla} \cdot (D_p \boldsymbol{\nabla} C_d) = -\lambda_d C_d + \beta_d \sum_g (\nu \Sigma_f)_g \phi_g \tag{2.16}$$

$$-\boldsymbol{\nabla} \cdot (D_g \boldsymbol{\nabla} \phi_g) = -\Sigma_{t,g}\phi_g + \chi_g^p (1-\beta_{tot}) \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \chi_g^d \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g' \to g} \phi_{g'} \tag{2.17}$$

However, these equations only have a solution when the neutronics parameters (Σ_t, χ^p, β_d, etc.) and geometric dimensions are chosen for criticality. However, often one wants to assess if the chosen parameters and dimensions will results in criticality. For this reason, equations 2.16 and 2.17 are studied as an eigenvalue problem

$$\boldsymbol{\nabla} \cdot (\boldsymbol{u} C_d) - \boldsymbol{\nabla} \cdot (D_p \boldsymbol{\nabla} C_d) = -\lambda_d C_d + \frac{\beta_d}{k_{\mathit{eff}}} \sum_g (\nu \Sigma_f)_g \phi_g \tag{2.18}$$

$$-\boldsymbol{\nabla} \cdot (D_g \boldsymbol{\nabla} \phi_g) = -\Sigma_{t,g}\phi_g + \frac{\chi_g^p (1-\beta_{tot})}{k_{\mathit{eff}}} \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \chi_g^d \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g' \to g} \phi_{g'} \tag{2.19}$$

where the eigenvalue k_{eff} is incorporated into the fission terms and represents the multiplication factor of the reactor. The fluxes ϕ_g and concentrations found are the eigenvectors of the system.

2.3.2. Power Method for k-Eigenvalue Problem

To solve the system of equations described by equations 2.18 and 2.19 and find k_{eff}, C_d, and ϕ_g the system is rewritten to matrix form:

$$\mathbf{M}\gamma = \frac{1}{k_{\mathit{eff}}} \mathbf{F}\gamma \tag{2.20}$$

where matrix $\mathbf{M}$ can be identified as the destruction operator and matrix $\mathbf{F}$ as the fission operator. γ is the eigenvector. For a 2 neutron group and single precursor family system, these matrices and eigenvector look the following [10]

$$\boldsymbol{\gamma} = [C_1, \phi_1, \phi_2]^T \tag{2.21}$$

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{\nabla}\cdot\mathbf{u} - \boldsymbol{\nabla}\cdot D_p\boldsymbol{\nabla} - \lambda_1 & 0 & 0 \\ \chi_1^d\lambda_1 & -\boldsymbol{\nabla}\cdot D_1\boldsymbol{\nabla} + \Sigma_{t,1} - \Sigma_{s,1\to1} & 0 \\ \chi_2^d\lambda_1 & \Sigma_{s,1\to2} & -\boldsymbol{\nabla}\cdot D_2\boldsymbol{\nabla} + \Sigma_{t,2} - \Sigma_{s,2\to2} \end{bmatrix} \tag{2.22}$$

$$\mathbf{F} = \begin{bmatrix} 0 & \beta_1\nu_1\Sigma_{f,1} & \beta_1\nu_2\Sigma_{f,2} \\ 0 & \chi_1^p(1-\beta_1)\nu_1\Sigma_{f,1} & \chi_1^p(1-\beta_1)\nu_2\Sigma_{f,2} \\ 0 & \chi_2^p(1-\beta_1)\nu_1\Sigma_{f,1} & \chi_2^p(1-\beta_1)\nu_2\Sigma_{f,2} \end{bmatrix} \tag{2.23}$$

These matrices can be rewritten in a more general block matrix form

$$\begin{bmatrix} A_{CC}(\boldsymbol{u}) & 0 \\ A_{\phi C} & A_{\phi\phi} \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \boldsymbol{\phi} \end{bmatrix} = \frac{1}{k_{\mathit{eff}}} \begin{bmatrix} 0 & A_{C\phi}^F \\ 0 & A_{\phi\phi}^F \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \boldsymbol{\phi} \end{bmatrix} \tag{2.24}$$

where A_{CC} is an operator only acting on and contributing to precursors concentrations, $A_{\phi\phi}$ an operator only acting on and contributing to neutron fluxes, and $A_{\phi C}$ acting on precursor concentrations and contributing to neutron fluxes. Similarly, $A_{C\phi}^F$ is a fission operator acting on neutron fluxes and contributing to precursor concentrations and $A_{\phi\phi}^F$ acting on and contributing to neutron fluxes. The $\mathbf{M}$ matrix on the left can be identified as a lower triangular matrix. The power method can be used to find the largest eigenvalue and accompanying eigenvector for this kind of matrix systems.

The power method is an iterative method which starts with an initial guess for the effective multiplication factor and fission source at $l = 0$. From these initial guesses k_{eff}^0 and S^0 a new value for the eigenvector $\boldsymbol{\gamma}$ is found using

$$\mathbf{M}\boldsymbol{\gamma}^{(l+1)} = \frac{1}{k_{\mathit{eff}}^{(l)}} S^{(l)} \tag{2.25}$$

The new fission source and effective multiplication value are then determined using

$$S^{(l+1)} = \mathbf{F}\boldsymbol{\gamma}^{(l+1)} \tag{2.26}$$

$$k_{\mathit{eff}}^{(l+1)} = k_{\mathit{eff}}^{(l)} \frac{\int S^{(l+1)}\,\mathrm{d}V}{\int S^{(l)}\,\mathrm{d}V} \tag{2.27}$$

This scheme is repeated until the difference $(k_{\mathit{eff}}^{(l+1)} - k_{\mathit{eff}}^{(l)})$ is deemed small enough.

2.3.3. Transient Behaviour: the Predictor-Corrector Quasi-Static Method

To study transient reactor core problems, equations 2.14 and 2.15 have to be solved. To solve this system of equations, the neutron flux is factorized in a slowly changing neutron *shape* function $\tilde{\phi}(\boldsymbol{r}, t)$ and a fast changing neutron *amplitude* function $n(t)$ [10, 11]. This factorization is not an approximation and results in the following expression for the neutron flux

$$\phi_g(\boldsymbol{r}, t) = \tilde{\phi}_g(\boldsymbol{r}, t)\, n(t) \tag{2.28}$$

Using this factorization, the fast changing neutron amplitude function is solved for small timescales, while the neutron shape functions is solved for large timescales. This is called a quasi-static method, since the shape function is assumed to be constant over small timescales.

Using this factorization and some assumptions, point-kinetics (PK) equations can be derived from the neutron diffusion equation and precursors advection-diffusion equation describing the transient

behaviour of the reactor core. In the following derivation, the operators that act on the neutron flux are shortened as shown below.

$$M_g\phi_g = -\boldsymbol{\nabla}\cdot(D_g\boldsymbol{\nabla}\phi_g) + \Sigma_t\phi_g - \sum_{g'}\Sigma_{s,g'\to g}\phi_{g'} \tag{2.29}$$

$$F_g^P\phi_g = \chi_g^p(1-\beta_{tot})\sum_{g'}(\nu\Sigma_f)_{g'}\phi_{g'} \tag{2.30}$$

$$F_{d,g}^D\phi_g = \chi_g^d\beta_d\sum_{g'}(\nu\Sigma_f)_{g'}\phi_{g'} \tag{2.31}$$

$$F_g^D\phi_g = \sum_d F_{d,g}^D\phi_g \tag{2.32}$$

$$F_g\phi_g = \big(F_g^P + F_g^D\big)\phi_g \tag{2.33}$$

$$S_g^D = \chi_g^d\sum_d\lambda_d C_d \tag{2.34}$$

By applying the neutron flux factorization of equation 2.28 to the neutron diffusion equation from equation 2.15 and using the previously defined operators, one arrives at the equation

$$\Phi_g^*\frac{1}{v_g}\tilde{\phi}_g\frac{\partial n}{\partial t} + \Phi_g^*\frac{1}{v_g}n\frac{\partial\tilde{\phi}_g}{\partial t} = \Phi_g^*\big(F_g^P - M_g\big)\tilde{\phi}_g n + \Phi_g^* S^D \tag{2.35}$$

Here, the conjugate neutron flux Φ_g^* has been introduced. The conjugate neutron flux is used as a weighing function to ensure the uniqueness of the factorization of equation 2.28 [11]. The conjugate neutron flux is defined as the solution to the steady-state neutron diffusion equation from equation 2.19 at the initial time. The conjugate neutron flux is normalised using the condition

$$\left\langle\Phi_g^*, \frac{1}{v_g}\tilde{\phi}_g\right\rangle = 1 \tag{2.36}$$

Continuing the derivation, equation 2.35 is integrated over the entire domain, where $\langle x, y\rangle$ denotes the volume integration of x and y. The second term on the left-hand side is neglected, since the neutron shape function is constant over small timescales. This results in

$$\left\langle\Phi_g^*, \frac{1}{v_g}\tilde{\phi}_g\right\rangle\frac{\partial n}{\partial t} = \left\langle\Phi_g^*, \big(F_g^P - M_g\big)\tilde{\phi}_g\right\rangle n + \left\langle\Phi_g^*, S_g^D\right\rangle \tag{2.37}$$

Applying the same steps to the precursor advection diffusion equation from equation 2.14, assuming no convection or diffusion, results in [44]

$$\frac{\partial\left\langle\Phi_g^*, \chi_g^d C_d\right\rangle}{\partial t} = \left\langle\Phi_g^*, F_{d,g}^D\tilde{\phi}_g\right\rangle n - \lambda_d\left\langle\Phi_g^*, \chi_g^d C_d\right\rangle \tag{2.38}$$

The assumption of no convection and diffusion is justified because the timescales for convection and diffusion are generally much larger than the timescales for precursor concentration change. We now introduce the definition

$$m_d = \frac{\sum_g\left\langle\Phi_g^*, \chi_g^d C_d\right\rangle}{\sum_g\left\langle\Phi_g^*, \frac{1}{v_g}\tilde{\phi}_g\right\rangle} \tag{2.39}$$

By dividing equations 2.35 and 2.37 by $\left\langle\Phi_g^*, \frac{1}{v_g}\tilde{\phi}_g\right\rangle$ and summing over all neutron groups, we arrive at

$$\frac{\partial n}{\partial t} = \frac{\left\langle \Phi_g^*, \left(F_g^P - M_g\right)\tilde{\phi}\right\rangle}{\left\langle \Phi_g^*, \frac{1}{v_g}\tilde{\phi}_g\right\rangle} n + \sum_d \lambda_d m_d \tag{2.40}$$

$$\frac{\partial m_d}{\partial t} = \frac{\left\langle \Phi_g^*, F_{d,g}^D \tilde{\phi}_g\right\rangle}{\left\langle \Phi_g^*, \frac{1}{v_g}\tilde{\phi}_g\right\rangle} n - \lambda_d m_d \tag{2.41}$$

These equation are a form of PK equations, where the PK parameters are expressed as a function of the slowly changing neutron shape function. The equations are rewritten to [44]

$$\frac{\mathrm{d}n(t)}{\mathrm{d}t} = \frac{\rho(t) - \beta_{tot}(t)}{\Lambda(t)} n(t) + \sum_d \lambda_d m_d(t) \tag{2.42}$$

$$\frac{\mathrm{d}m_d(t)}{\mathrm{d}t} = \frac{\beta_d(t)}{\Lambda(t)} n(t) - \lambda_d m_d(t) \tag{2.43}$$

Where the PK parameters are defined as [44]

$$\rho(t) = \frac{\sum_g \left\langle \Phi_g^*, (F_g - M_g)\tilde{\phi}_g\right\rangle}{\sum_g \left\langle \Phi_g^*, F_g\tilde{\phi}_g\right\rangle} \tag{2.44}$$

$$\beta_d(t) = \frac{\sum_g \left\langle \Phi_g^*, F_{d,g}^D\tilde{\phi}_g\right\rangle}{\sum_g \left\langle \Phi_g^*, F_g\tilde{\phi}_g\right\rangle} \tag{2.45}$$

$$\beta_{tot}(t) = \sum_d \beta_d \tag{2.46}$$

$$\Lambda(t) = \frac{\sum_g \left\langle \Phi_g^*, \frac{1}{v_g}\tilde{\phi}_g\right\rangle}{\sum_g \left\langle \Phi_g^*, F_g\tilde{\phi}_g\right\rangle} \tag{2.47}$$

2.4. Interaction Between Fields

In the prior sections, the physics of each field has been discussed separately. However, in a molten salt reactor these four fields are strongly coupled to each other through a number of different mechanisms, which makes the thermal-hydraulics-neutronics of a MSFR a multiphysics problem. In this section all interactions are summarized and figure 2.2 shows a schematic of the four fields and their interactions.

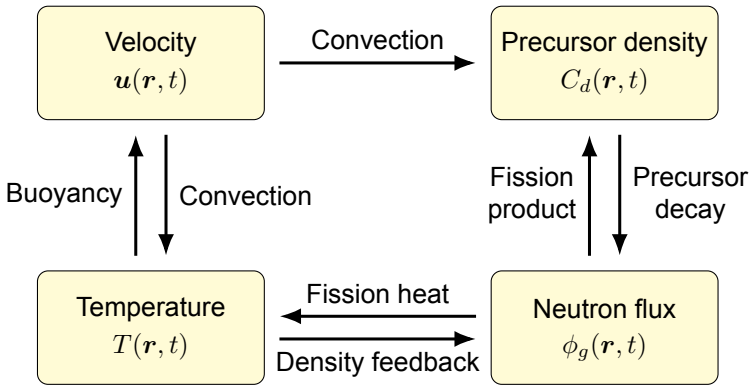


Figure 2.2: Schematic of the interactions between the four fields describing the thermal-hydraulics and neutronics of a MSFR.

Firstly, the velocity interacts with the temperature and precursor density fields through convection. The precursor density and neutron flux interact with each other through fission production and precursor decay.

The temperature has a momentum feedback through the buoyancy force. Since the salt is modeled as an incompressible fluid, the Boussinesq approximation is used. In the Boussinesq approximation, it is assumed that density differences can be neglected for all heat, mass, and momentum terms except for the buoyancy force. Density variations of the molten salt are modeled using a linear temperature dependence on a reference temperature [33].

$$\rho(T) = \rho(T_{ref})(1 - \beta_{th}(T - T_{ref})) \tag{2.48}$$

where β_{th} denotes the thermal expansion coefficient in units of K^{-1}. This results in the following temperature-dependent buoyancy force.

$$\boldsymbol{f}_{buoyancy} = -\beta_{th}\,\rho(T_{ref})(T - T_{ref})\boldsymbol{g} \tag{2.49}$$

Besides neutrons and precursors, fission also produces heat, coupling the neutron flux to the temperature. The fission heat is modeled as a heat source q where each fission event produces $E_{fission}$ of energy.

$$q = E_{fission} \sum_{g} \Sigma_{f,g}\phi_g \tag{2.50}$$

Lastly, the temperature has a feedback effect on the neutronics through two mechanisms: salt expansion and Doppler broadening. Due to temperature changes, the salt will expand or shrink, thereby changing the number density (number of atoms per volume) of the salt. Since the macroscopic cross section is the product of the microscopic cross section and the number density, the macroscopic cross section and neutron diffusion cross section will also change through the expansion (or shrinkage) of the salt. Additionally, a rise in temperature will broaden the absorption spectrum of the salt due to the thermal motion of the atoms. This is called Doppler broadening. In this research only the salt expansion feedback will be taken into account, the equations for the change in macroscopic cross section and neutron diffusion constant are defined as

$$\Sigma_{x,g}(T) = \Sigma_{x,g}(T_{ref})\frac{\rho(T)}{\rho(T_{ref})} \tag{2.51}$$

$$D_g(T) = D_g(T_{ref})\frac{\rho(T_{ref})}{\rho(T)} \tag{2.52}$$

Which using equation 2.48 can be rewritten to

$$\Sigma_{x,g}(T) = \Sigma_{x,g}(T_{ref})(1 - \beta_{th}(T - T_{ref})) \tag{2.53}$$

$$D_g(T) = D_g(T_{ref})\frac{1}{1 - \beta_{th}(T - T_{ref})} \tag{2.54}$$

2.5. Kinetic Theory

To understand the lattice Boltzmann method, which will be discussed in the next chapter, knowledge of kinetic theory is required. Like the NSE, kinetic theory is used to describe the behaviour of fluids. However, while the NSE describes fluids from a macroscopic perspective by giving equations for density and fluid velocity, kinetic theory describes fluid behaviour from a mesoscopic perspective. Kinetic theory tracks the distribution of particles in a gas, which on a macroscopic scale leads to the NSE. The particle distribution is given by $f(\boldsymbol{r}, \boldsymbol{\xi}, t)$ and has 7 degrees of freedom: 3 spatial, 1 temporal, and 3 in velocity space (denoted by $\boldsymbol{\xi}$). While the distribution function itself does not represent any physical quantities its integrals, called moments, do. The following moments are defined for the particle distribution function $f(\boldsymbol{r}, \boldsymbol{\xi}, t)$ [23]:

$$\rho(\boldsymbol{r}, t) = \int f(\boldsymbol{r}, \boldsymbol{\xi}, t)\,\mathrm{d}\boldsymbol{\xi} \tag{2.55}$$

$$\rho(\boldsymbol{r}, t)\boldsymbol{u}(\boldsymbol{r}, t) = \int \boldsymbol{\xi}\, f(\boldsymbol{r}, \boldsymbol{\xi}, t)\,\mathrm{d}\boldsymbol{\xi} \tag{2.56}$$

Where $\boldsymbol{u}(\boldsymbol{r}, t)$ is the mean velocity of the gas. To study the evolution of the particle distribution function over time, the time derivative is taken

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \left(\frac{\partial f}{\partial t}\right)\frac{\mathrm{d}t}{\mathrm{d}t} + \left(\frac{\partial f}{\partial \boldsymbol{r}}\right)\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} + \left(\frac{\partial f}{\partial \boldsymbol{\xi}}\right)\frac{\mathrm{d}\boldsymbol{\xi}}{\mathrm{d}t} \tag{2.57}$$

This equation can be simplified by recognizing some derivatives. In the first term on the right-hand side we can rewrite $\mathrm{d}t\,/\,\mathrm{d}t = 1$. In the second term, $\mathrm{d}\boldsymbol{r}\,/\,\mathrm{d}t = \boldsymbol{\xi}$ can be identified. Lastly, the velocity derivative $\mathrm{d}\boldsymbol{\xi}\,/\,\mathrm{d}t$ can be rewritten as the bodyforce $\boldsymbol{f}$. By replacing the total differential $\mathrm{d}f\,/\,\mathrm{d}t$ by $\Omega(f)$, we arrive at the Boltzmann equation

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \boldsymbol{\nabla} f + \boldsymbol{f} \cdot \boldsymbol{\nabla}_{\boldsymbol{\xi}} f = \Omega(f) \tag{2.58}$$

Here, the first two terms represent the advection of the distribution function with particle velocity $\boldsymbol{\xi}$. The third term represents the forces acting on the distribution function. On the right-hand side is a source term which represents the redistribution of f due to collisions. Therefore, $\Omega(f)$ is called the collision operator.

The redistribution of particles should conserve mass and momentum for a infinitesimal volume at $\boldsymbol{r}$. Hence, a collision operator should obey the following moments

$$\int \Omega(f)\,\mathrm{d}\boldsymbol{\xi} = 0 \tag{2.59}$$

$$\int \boldsymbol{\xi}\,\Omega(f)\,\mathrm{d}\boldsymbol{\xi} = \mathbf{0} \tag{2.60}$$

Additionally, the collision operator should result in the equilibrium distribution function $f^{eq}(\boldsymbol{r}, \boldsymbol{\xi}, t)$ as time goes to infinity. The equilibrium distribution is isotropic in velocity space around the mean gas velocity $\boldsymbol{u}$ [23].

Through Chapman-Enskog analysis, it can be shown that the Boltzmann equation leads to the NSE on a macroscopic scale. The Chapman-Enskog analysis decomposes the particle distribution function into an equilibrium and non-equilibrium part. The particle distribution functions are then expanded through a perturbation and terms of similar order are grouped together. Using the moments defined in equations 2.55 and 2.56 the same conversation laws of section 2.1.1 are found. For a detailed description of the Chapman-Enskog analysis, the reader is referred to [23].

2.6. Parallel GPU Programming

Simulating thermal fluid flow using the NSE is notoriously computationally expensive [13]. Therefore, this research also focuses on speeding up the computational procedure of the simulation. The simulation method used in this research, the lattice Boltzmann method (LBM) is a method where the most numerical intensive operation only uses information at a local (nodal) level. Since no global information is required for this operation, this operation can be parallelized, which has the potential to significantly reduce computation times. The operations of the LBM are small enough that they can be executed on a graphical processing unit (GPU). A GPU contains thousands of small cores that are individually less powerful than CPU that can execute massive tasks in parallel. In this section, the fundamentals of GPU programming will be outlined to show the advantages of parallelization for certain computation tasks.

2.6.1. CUDA Programming Language

In this research, the Compute Unified Device Architecture (CUDA) programming language, developed by NVIDIA, is used to write algorithms for the GPU. CUDA is developed by NVIDIA and allows developers to write programs that are directly executed on the GPU [27]. The functions that are executed on the GPU cores, called kernels, should be defined to run independently to allow for parallel execution. Since the GPU is controlled by the CPU, the general procedure for parallel programming using CUDA is:

1. Transfer input variables from CPU memory to GPU memory
2. Allocate GPU memory for simulation output
3. Run the simulation by executing kernels on the GPU
4. Transfer the simulation results from GPU memory to CPU memory

Several challenges related to GPUs are encountered when using CUDA for parallel programming. Firstly, the cores of a GPU are less powerful than those of a CPU core. Therefore, the complexity of the kernel functions should be minimized as much as possible. This is best done by adopting a modular code design, where each kernel function has a single, well-defined task. The use of single-precision floating-point number is also recommended, since this halves the computational burden compared to double-precision floating-point numbers.

Secondly, CUDA only supports a limited set of operations for the kernel function. Most importantly, vector operations like matrix multiplication are not built-in and need to be written out using loops. This makes kernel programming more laborious, however it also allows for greater control and optimization of these complex operations.

Lastly, CUDA assumes that calculations and memory addresses accessed by kernels are independent and do not overlap. When the independence is not maintained, race conditions may arise, where multiple kernels on different cores access and modify the same memory location simultaneously. These

situations do not result in CUDA generating an error, however the results become distorted, making identifying and debugging race conditions challenging. Race conditions can be avoided by using features such as atomic operations and synchronization between kernel executions.

2.6.2. GPU Hardware Architecture and Memory Hierarchy

NVIDIA GPUs are organized in a three-layered architecture to optimize memory handling and processing efficiency. This layered architecture has implications for the software abstractions used in CUDA and needs to be taken into account for optimal GPU usage.

The layered structure of the GPU is shown in figure 2.3. The first layer consists of all the cores of the GPU. The intermediate layer consists of streaming multiprocessors (SMs), which are a group of cores responsible for the parallel execution of tasks. Each SM contains a warp scheduler which is responsible for scheduling the tasks that will be discussed in more detail in a later section. The last layer is the individual core, which executes kernels scheduled by the SM.

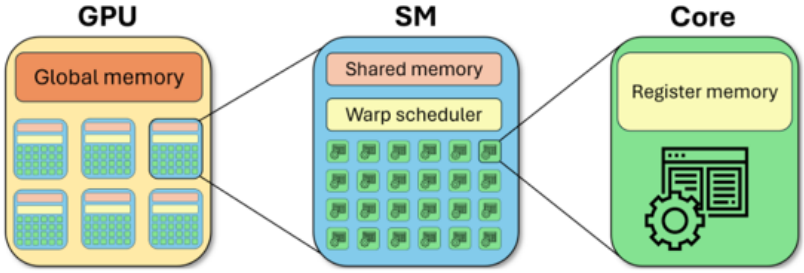


Figure 2.3: The layered hardware architecture of a NVIDIA GPU.

Each layer also has an associated memory, as can be seen in figure 2.3. The first layer contains the global memory, which has a large size, exists for the whole program time, and is accessible by all cores for reading and writing. However, this comes with the downside that reading and writing to global memory is slow. The shared memory is accessible by all cores in a SM and exists for the lifetime of the SM task. The shared memory is relatively small; however, it has a quick access speed. The last type of memory is the register memory, which is only accessible by a single core. The register memory is the smallest type of memory and the most quick to access. However, its lifetime is short: it exists only for the execution time of a kernel. When developing GPU code, it is essential to consider optimal storage locations. For instance, variables that are only required for kernel computation can best be stored in register memory, while constants and final results that need to be transferred back to the GPU can best be stored in global memory. An overview of the types of memory and their characteristics is shown in table 2.2.

Table 2.2: Overview of the types of memory of a NVIDIA GPU. For each type of memory their accessibility, access speed, size, and lifetime is shown.

Memory type	Accessibility	Access speed	Size	Lifetime
Global memory	All cores + CPU	Slow	Large	Program time
Shared memory	Cores in SM	Fast	Small	SM task lifetime
Register memory	Single core	Fastest	Very small	Kernel lifetime

2.6.3. CUDA Software Abstractions

In the CUDA programming language, four layers of software abstractions exist to control the task division of the GPU. These layers are illustrated in figure 2.4. The lowest layer is the thread, the smallest computation unit of a program, usually performing operations on a single lattice point. Warps, the second layer, are a group of 32 threads and are scheduled to a single SM. A collection of warps forms a block, the third layer. The number of warps in a block can be specified, thus allowing for the optimized use of a GPU. All blocks together make up the grid, the top layer of the CUDA abstractions.

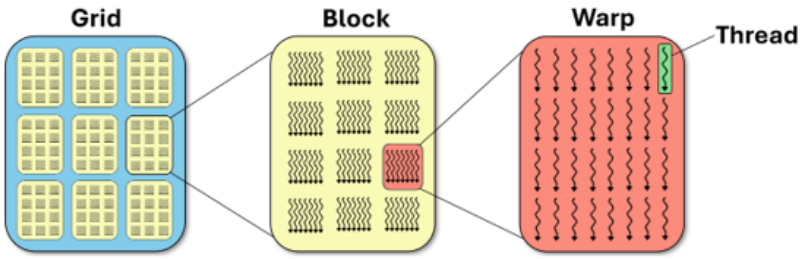


Figure 2.4: Overview of the software abstractions used in CUDA. The number of threads in a warp is fixed at 32, however CUDA allows to specify the number of blocks and the number of threads per block.

To conclude, the task division using CUDA can namely be controlled by specifying the number of threads per block. Since 32 threads will always be collected in a warp and be scheduled to a SM as a warp, for optimal use the threads per block should always be chosen in multiples of 32.

3

Numerical Method

In this chapter the numerical techniques used in this research will be discussed. The four physical fields (momentum, enthalpy, neutronics, and precursors) will all be simulated using the lattice Boltzmann method (LBM). The fundamentals of the LBM and the filter-matrix algorithm used will be explained in sections 3.1 and 3.2, respectively. A detailed description of boundary condition handling will be given in section 3.3. The conversion of physical units to lattice units will be discussed in section 3.4. In section 3.5 the solving scheme for the coupled system is presented. Section 3.6 presents the implementation of the predictor-corrector quasi-static method, previously discussed in section 2.3.3, to solve transient neutronics. Lastly, the GPU acceleration of the FM-LBM algorithm is discussed in section 3.7 including several optimization techniques.

3.1. Fundamentals of the Lattice Boltzmann Method

As discussed in section 2.5, the Boltzmann equation gives a mesoscopic description of fluid behaviour. However, the Boltzmann equation is notorious for being even harder to solve analytically than the Navier-Stokes equations (NSE). To solve the Boltzmann equation numerically, it will be discretized, leading to the lattice Boltzmann method (LBM). The LBM is surprisingly easy to solve numerically and, as it turns out, well suited for GPU parallelization.

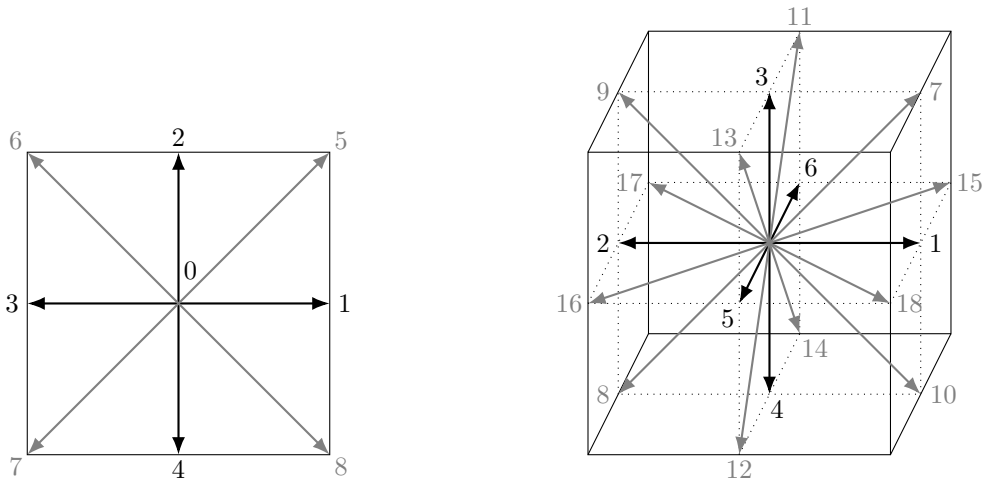
The Boltzmann equation is discretized in time, space, and velocity. The time is discretized in timesteps Δt. The space is discretized by dividing the domain into lattice points that are spaced Δx apart. The velocity discretization derives from the time and space discretization: between each time step, the distribution function can flow to neighbouring lattice points. This results in a discretized velocity for vertical and horizontal movements of $\Delta x/\Delta t$, for square diagonals $\sqrt{2}\,\Delta x/\Delta t$, and for cubic diagonals $\sqrt{3}\,\Delta x/\Delta t$. The discretized velocities are denoted by $\boldsymbol{c}_i = [c_{ix}, c_{iy}, c_{iz}]^\intercal$, where i is the direction of the flow. In the LBM, the distribution function is discretized along the discretized velocities: $f(\boldsymbol{r}, \boldsymbol{\xi}, t) \rightarrow f_i(\boldsymbol{r}, t)$.

Each lattice velocity $\boldsymbol{c}_i$ also has an accompanying weight ω_i. Together, these velocity-weight sets form a *lattice velocity set*. One can choose which (diagonal) movements are allowed in a set, resulting in the $\mathrm{D}d\mathrm{Q}q$ notation for lattice velocity sets, where d is the number of spatial dimensions and q the number of velocities. In figure 3.1 the most common velocity sets for 2 and 3 dimensional domains are shown. Note that all of these sets contain a rest velocity $\boldsymbol{c}_0 = [0, 0(, 0)]^\intercal$.

By applying this discretization to the Boltzmann equation from equation 2.58, the lattice Boltzmann equation (LBE) is found

$$f_i(\boldsymbol{r} + \boldsymbol{c}_i \Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \Omega_i(\boldsymbol{r}, t) \tag{3.1}$$

This expresses that during a timestep Δt particles $f_i(\boldsymbol{r}, t)$ move with velocity $\boldsymbol{c}_i$ to a neighbouring point $\boldsymbol{r} + \boldsymbol{c}_i \Delta t$, while being affected by the collision operator $\Omega_i(\boldsymbol{r}, t)$.



(a) Two 2-dimensional lattice velocity sets in the LBM; $\mathrm{D2Q5}$ in black and $\mathrm{D2Q9}$ in black and gray [23].

(b) Two 3-dimensional lattice velocity sets in the LBM; $\mathrm{D3Q7}$ in black and $\mathrm{D3Q19}$ in black and gray. The rest velocity ($\boldsymbol{c}_0 = [0, 0, 0]^\intercal$ is not shown [23].

Figure 3.1: Common lattice velocity sets for 2 and 3 dimensional LBM simulations.

From this discretization naturally follows the simulation procedure. During each timestep Δt, there is first a collision step, where the new distribution $f_i^*(\boldsymbol{r}, t)$ at each node is determined using the previous distribution $f_i(\boldsymbol{r}, t)$. Next, the streaming step follows, where the distributions $f_i^*(\boldsymbol{r}, t)$ stream with their velocity $\boldsymbol{c}_i$ to the neighbouring nodes $(\boldsymbol{r}+\boldsymbol{c}_i\Delta t, t+\Delta t)$. This procedure is schematically shown in figure 3.2.

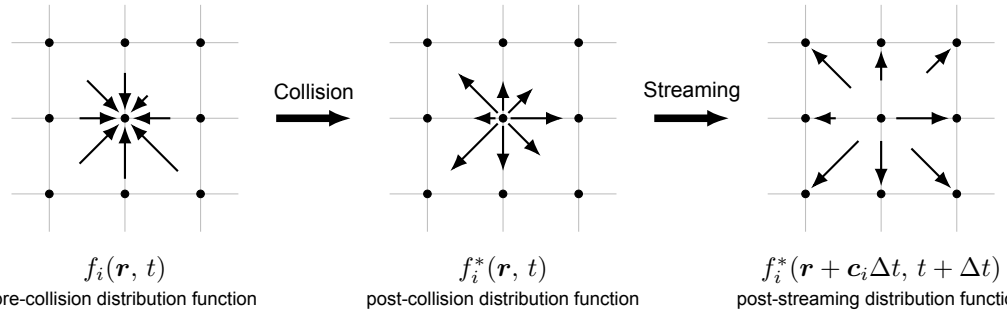


Figure 3.2: Schematic overview of the collision and streaming step for the distribution function f_i.

As with the Boltzmann equation, the distribution function of the LBE does not describe a physical quantity; however, its moments do. Since the LBE is the discretized version of the Boltzmann equation, the integrals have been replaced by sums and the following moments are defined for the velocity distribution function $f_i(\boldsymbol{r}, t)$ [23].

$$\rho = \sum_i f_i, \qquad \rho\boldsymbol{u} = \sum_i \boldsymbol{c}_i f_i \tag{3.2}$$

There is some freedom in choosing the discretized collision operator $\Omega_i(f)$ introduced in equation 3.1. The collision operator must satisfy the moments described in equations 2.59 and 2.60 and in the limit it should relax the distribution function to the equilibrium distribution given by the Maxwell-Boltzmann distribution. For the LBM, the particle density equilibrium function is given by

$$f_i^{eq} = \rho\omega_i\left(1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2}\right) \tag{3.3}$$

Where ρ and $\boldsymbol{u}$ denote the macroscopic fluid density and velocity. ω_i are the weights for each velocity $\boldsymbol{c}_i$. c_s is the lattice speed of sound. Through Chapman-Enskog analysis it can be shown that this equilibrium distribution results in the force-free NSE [23].

The most simple collision operator is the Bhatnager-Gross-Krook (BGK) operator, which relaxes the distribution towards equilibrium using a single relaxation parameter τ, which is defined as

$$\Omega_{BGK}(f) = -\frac{1}{\tau}(f_i - f_i^{eq}) \tag{3.4}$$

The BGK operator is simple to implement and can simulate a large range of fluid behaviours well. However, the BGK operator also has its downsides. The value of the relaxation time τ is directly coupled to the viscosity of the fluid, and for stable simulations the relaxation time is limited to a small range of values, thereby limiting the viscosity range of the simulations. Therefore, more sophisticated methods have been developed, such as the two-relaxation times operator (TRT) [18], the multi-relaxation times operator (MRT) [5], and the filter-matrix lattice Boltzmann method (FM-LBM) operator [23, 55]. In this research the FM-LBM operator is used, which will be discussed in the next section.

3.2. The Filter-Matrix Lattice Boltzmann Method

The filter-matrix lattice Boltzmann method (FM-LBM) operator can be derived from the LBE by shifting the spatial and temporal dimensions by half a grid and time spacing, resulting in the staggered grid approach [23]. Using the staggered approach, equation 3.1 is rewritten to

$$f_i\left(\boldsymbol{r} + \frac{\Delta t}{2}\boldsymbol{c}_i, t + \frac{\Delta t}{2}\right) = f_i\left(\boldsymbol{r} - \frac{\Delta t}{2}\boldsymbol{c}_i, t - \frac{\Delta t}{2}\right) + \Delta t\, \Omega_i(f) \tag{3.5}$$

Applying a Taylor expansion around $f_i(\boldsymbol{r}, t)$ and combining this with the staggered LBE in equation 3.5 results in

$$f_i\left(\boldsymbol{r} \pm \frac{\Delta t}{2}\boldsymbol{c}_i, t \pm \frac{\Delta t}{2}\right) = f_i(\boldsymbol{r}, t) \pm \frac{\Delta t}{2}\Omega_i(f) + \mathcal{O}(\Delta t^2) \tag{3.6}$$

By applying the Chapman-Enskog analysis to equation 3.6, expressions for the particle distributions f_i and collision operator Ω_i in terms of macroscopic variables are found. These expressions ensure that the LBE simulates incompressible NSE flow and are given by

$$f_i(\boldsymbol{r}, t) = \rho\omega_i\left[1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{1}{2}\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{c_s^2}\right) - \nu\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{\nabla})(\boldsymbol{c}_i \cdot \boldsymbol{u})}{c_s^4} - \frac{\boldsymbol{\nabla} \cdot \boldsymbol{u}}{c_s^2}\right)\right] \tag{3.7}$$

$$\Omega_i(f) = \frac{\rho\omega_i}{c_s^2}\left[(\boldsymbol{c}_i \cdot \boldsymbol{\nabla})(\boldsymbol{c}_i \cdot \boldsymbol{u}) - c_s^2\, \boldsymbol{\nabla} \cdot \boldsymbol{u} + \boldsymbol{c}_i \cdot \boldsymbol{f}\right] \tag{3.8}$$

Here, ν denotes the kinematic viscosity and $\boldsymbol{f}$ the body force acting on the fluid. By substituting the expression for the particle distribution and collision operator into equation 3.6 one arrives at

$$\begin{aligned} f_i\left(\boldsymbol{r} \pm \frac{\Delta t}{2}\boldsymbol{c}_i, t \pm \frac{\Delta t}{2}\right) = \rho\omega_i\Bigg[1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{1}{2}\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{c_s^2}\right) - \nu\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{\nabla})(\boldsymbol{c}_i \cdot \boldsymbol{u})}{c_s^4} - \frac{\boldsymbol{\nabla} \cdot \boldsymbol{u}}{c_s^2}\right) \\ \pm \frac{\Delta t}{2}\left(\frac{(\boldsymbol{c}_i \cdot \boldsymbol{\nabla})(\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} - \boldsymbol{\nabla} \cdot \boldsymbol{u} + \frac{\boldsymbol{c}_i \cdot \boldsymbol{f}}{c_s^2}\right)\Bigg] \end{aligned} \tag{3.9}$$

The equation above can also be written more concisely as a matrix multiplication, by introducing the filter matrix E_{ik} and the solution vector $\alpha_k^{\pm}(\boldsymbol{r}, t)$

$$f_i\left(\boldsymbol{r} \pm \frac{\Delta t}{2}\boldsymbol{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i E_{ik} \alpha_k^{\pm}(\boldsymbol{r}, t) \tag{3.10}$$

The filter matrix E_{ik} depends on the mesoscopic velocities $\boldsymbol{c}_i$, while the solution vector $\alpha_k^{\pm}$ is defined in terms of macroscopic quantities. The definition of the filter matrix and the solution vector depends on the lattice velocity set used and physical quantity simulated. In later sections definitions for the filter matrix and solution vector will be given. Equation 3.10 can also be inverted by introducing the inverted matrix E_{ki}, defined as $\omega_i E_{ki} = (E_{ki})^{-1}$, resulting in

$$\alpha_k^{\pm}(\boldsymbol{r}, t) = \sum_i E_{ki} f_i\left(\boldsymbol{r} \pm \frac{\Delta t}{2}\boldsymbol{c}_i, t \pm \frac{\Delta t}{2}\right) \tag{3.11}$$

The FMLBM is a powerful collision operator, since it allows us to transform the non-physical distribution function to the physical solution vector and back using the E_{ik} and E_{ki} filter matrices. The collision algorithm for the FMLBM operator is as follows:

1. The pre-collision distribution function f_i is transformed to the pre-collision vector α_k^- using the filter matrix E_{ik}.
2. The pre-collision solution vector α_k^- is updated to the post-collision vector α_k^+.
3. The post-collision vector α_k^+ is transformed back to the post-collision distribution $f_i^\star$ using the filter matrix E_{ki}.

The exact definition of the solution vector depends on the lattice velocity set used and physical quantity being simulated. As will be shown in later sections, the solution vector is easily transformed from pre- to post-collision, making the FMLBM an easy and intuitive operator. Additionally, due to the lack of a relaxation parameter as is present in the BGK operator, the FMLBM operator can simulate a wider range of fluid behaviour before becoming unstable.

3.2.1. Lattice Velocity Sets

In this research, the $\mathrm{D3Q7}$ and $\mathrm{D3Q19}$ lattice velocity sets are used. For the $\mathrm{D3Q7}$ velocity set, the following weights and filter matrix are defined [50, 49]:

$$\omega_i^7 = \begin{cases} 1/4 & i = 1 \\ 1/8 & i = 2, 3, 4, 5, 6, 7 \end{cases} \tag{3.12}$$

$$E_{ki}^7 = \left[1, c_{ix}, c_{iy}, c_{iz}, 4c_{ix}^2 - 1, 4c_{iy}^2 - 1, 4c_{iz}^2 - 1\right]^\intercal \tag{3.13}$$

The speed of sound for the $\mathrm{D3Q7}$ velocity set, in lattice units (lattice time lt and lattice spacing ls), is $c_s^2 = 1/2\,\mathrm{ls}\,\mathrm{lt}^{-1}$. The speed of sound for the $\mathrm{D3Q19}$ velocity set is $c_s = 1/\sqrt{3}\,\mathrm{ls}\,\mathrm{lt}^{-1}$ [49]. For the $\mathrm{D3Q19}$ velocity set, the following weights and filter matrix are defined [23, 53]:

$$\omega_i^{19} = \begin{cases} 1/3 & i = 1 \\ 1/18 & i = 2, 3, 4, 5, 6, 7 \\ 1/36 & i = 8, 9, \ldots, 18, 19 \end{cases} \tag{3.14}$$

$$E_{ki}^{19} = \begin{bmatrix} 1 \\ c_{ix},\ c_{iy},\ c_{iz} \\ 3c_{ix}^2,\ 3c_{iy}^2,\ 3c_{iz}^2 \\ 3c_{iy}c_{iz},\ 3c_{ix}c_{iz},\ 3c_{ix}c_{iy} \\ 3c_{ix}\left(c_{iy}^2 - c_{iz}^2\right),\ 3c_{iy}\left(c_{iz}^2 - c_{ix}^2\right),\ 3c_{iz}\left(c_{ix}^2 - c_{iy}^2\right) \\ c_{ix}\left(3c_{iy}^2 + 3c_{iz}^2 - 2\right),\ c_{iy}\left(3c_{ix}^2 + 3c_{iz}^2 - 2\right),\ c_{iz}\left(3c_{ix}^2 + 3c_{iy}^2 - 2\right) \\ 3\left(2c_{ix}^2 - c_{iy}^2 - c_{iz}^2\right)\left(|\boldsymbol{c}|^2 - \frac{3}{2}\right) \\ 3\left(c_{iy}^2 - c_{iz}^2\right)\left(|\boldsymbol{c}_i|^2 - \frac{3}{2}\right) \\ 3|\boldsymbol{c}_i|^2\left(|\boldsymbol{c}_i|^2 - 2\right) + 1 \end{bmatrix} \tag{3.15}$$

3.2.2. Filter-Matrix for Momentum Transport

The momentum transport through the fluid is simulated using the particle density distribution function f_i. Momentum is transported according to the NSE defined in equation 2.3. To correctly simulate fluid flow, the $\mathrm{D3Q19}$ scheme is required to capture all complexities of the NSE [23]. Using the Chapman-Enskog analysis, the momentum solution vector $\alpha_k^{\pm}$ is defined as [53]:

$$\alpha_k^{\pm} = \begin{bmatrix} \rho \\ \rho u_x \pm \Delta t f_x/2 \\ \rho u_y \pm \Delta t f_y/2 \\ \rho u_z \pm \Delta t f_z/2 \\ 3\rho u_x^2 + \rho(-6\nu \pm \Delta t)\partial_x u_x + (2-3B)\rho\nu\boldsymbol{\nabla}\cdot\boldsymbol{u} \\ 3\rho u_y^2 + \rho(-6\nu \pm \Delta t)\partial_y u_y + (2-3B)\rho\nu\boldsymbol{\nabla}\cdot\boldsymbol{u} \\ 3\rho u_z^2 + \rho(-6\nu \pm \Delta t)\partial_z u_z + (2-3B)\rho\nu\boldsymbol{\nabla}\cdot\boldsymbol{u} \\ 3\rho u_y u_z + \rho(-3\nu \pm \Delta t/2)(\partial_y u_z + \partial_z u_y) \\ 3\rho u_x u_z + \rho(-3\nu \pm \Delta t/2)(\partial_x u_z + \partial_z u_x) \\ 3\rho u_x u_y + \rho(-3\nu \pm \Delta t/2)(\partial_x u_y + \partial_y u_x) \\ 0,\ k = 11, ..., 16 \\ 0,\ k = 17, 18, 19 \end{bmatrix} \tag{3.16}$$

Here, ρ is the fluid density, $u_{x,y,z}$ the fluid velocity along the Cartesian axes, $f_{x,y,z}$ the bodyforce components along the Cartesian axes, Δt the simulation time step, ν the kinematic viscosity, and B the ratio of the bulk and kinematic viscosities. $\partial_{x,y,z}$ denotes the spatial derivative. The terms $\alpha_{11-16}^{\pm}$ and $\alpha_{17-19}^{\pm}$ correspond to third and fourth order terms respectively. These terms are non-physical artifacts that arise from the discretization of the LBM. These terms are set to zero during the FMLBM collision, to filter out these non-physical terms, enhancing the stability of the simulation.

3.2.3. Filter-Matrix for Heat, Neutron, and Delayed Precursor Transport

The heat, neutron, and delayed precursor transport are all described by advection-diffusion type equations. This allows us to treat them in the same manner to derive their filter-matrix expressions, therefore they are discussed together in this section.

Similarly to momentum transport, distribution functions and solution vectors are defined for heat, neutron, and precursor transport. Heat is modeled by the heat distribution function g_i and the heat solution vector $\beta_k^{\pm}$. Neutrons are modeled by the neutron distribution function $n_{i,g}$ per neutron group g and the neutron solution vector $\gamma_{k,g}^{\pm}$. Delayed precursors are modeled by the precursor distribution function $p_{i,d}$ for delayed precursor family d and the precursor solution vector $\delta_{k,d}^{\pm}$. These distribution functions and solution vectors are related by

$$g_i\left(\boldsymbol{r} \pm \frac{\Delta t}{2}\boldsymbol{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i E_{ik} \beta_k^{\pm}(\boldsymbol{r}, t) \tag{3.17}$$

$$n_{i,g}\left(\boldsymbol{r} \pm \frac{\Delta t}{2}\boldsymbol{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i E_{ik} \gamma_{k,g}^{\pm}(\boldsymbol{r}, t) \tag{3.18}$$

$$p_{i,d}\left(\boldsymbol{r} \pm \frac{\Delta t}{2}\boldsymbol{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i E_{ik} \delta_{k,d}^{\pm}(\boldsymbol{r}, t) \tag{3.19}$$

The advection-diffusion equations model the scalar quantities of enthalpy h, neutron flux ϕ_g, and precursor concentration C_d, respectively. This allows us to define the following moments for the distribution functions

$$h = \sum_i g_i \tag{3.20}$$

$$\phi_g = \sum_i n_{i,g} \tag{3.21}$$

$$C_d = \sum_i p_{i,d} \tag{3.22}$$

The heat transport through the fluid is described by the heat equation from equation 2.5, which, as earlier mentioned, is a type of advection-diffusion equation. For advection-diffusion equations the $\mathrm{D3Q7}$ velocity set is sufficient, so this velocity set is used to model heat, neutron, and delayed precursor transport [23]. This significantly reduces memory usage and therefore computation time.

By performing a Chapman-Enskog analysis on the heat equation Zhuo found the following heat solution vector, expressed in enthalpy h [54]

$$\beta_k^{\pm} = \begin{bmatrix} h \pm \Delta t\, q/2 \\ hu_x + \frac{-8\alpha \pm \Delta t}{8}\partial_x h \\ hu_y + \frac{-8\alpha \pm \Delta t}{8}\partial_y h \\ hu_z + \frac{-8\alpha \pm \Delta t}{8}\partial_z h \\ 0,\ k = 5, 6, 7 \end{bmatrix} \tag{3.23}$$

The heat solution vector is physically interpreted that the first term tracks the enthalpy, the second to fourth terms the convection and diffusion in the Cartesian directions, and the fifth to seventh terms are non-physical higher-order terms, set to zero. The heat source q in the first term is defined as:

$$q = \gamma\left(\frac{h_{ref}}{\rho_{ref} c_p} - \frac{h}{\rho c_p}\right) + E_{fission} \sum_g \Sigma_{f,g} \phi_g \tag{3.24}$$

Where $\rho_{(ref)}$ is the (reference) fluid density and C_p the heat capacity. The first term on the right-hand side is the volumetric heat sink around the reference enthalpy h_{ref} and the second term is heat production due to fission.

In literature, no filter-matrix expression was found for the NDE (equation 2.19) and delayed precursor ADE (equation 2.18). However, since both of these equations are types of advection-diffusion equations, expressions for their solution vectors were derived analogously to the heat solution vector. This resulted in the following expression for the neutron solution vector:

$$\gamma^{\pm}_{k,g} = \begin{bmatrix} \phi_g \pm \Delta t\, S_g/2 \\ \frac{-8D_g \pm \Delta t}{8}\partial_x \phi_g \\ \frac{-8D_g \pm \Delta t}{8}\partial_y \phi_g \\ \frac{-8D_g \pm \Delta t}{8}\partial_z \phi_g \\ 0,\ k = 5,6,7 \end{bmatrix} \tag{3.25}$$

Here, it can be seen that the heat source q is replaced with the neutron source S_g and that the convective terms are omitted from the $\gamma^{\pm}_{2-4,g}$ terms, since the NDE is not affected by convection. D_g is the neutron diffusion constant. The neutron source term is defined as:

$$S_g = -\Sigma_{t,g}\phi_g + \chi^p_g \frac{1-\beta_{tot}}{k_{eff}} \sum_{g'} \left(\nu\Sigma_f\right)_{g'} \phi_{g'} + \chi^d_g \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g'\to g}\phi_{g'} \tag{3.26}$$

Where $\Sigma_{(t,f,s),g}$ is the macroscopic neutron cross section for removal, fission, and scattering, respectively. $\chi^{p,d}_g$ are the prompt and delayed neutron spectra. β_{tot} is the total delayed neutron fraction, k_{eff} the multiplication factor, ν_g the average numbers of neutrons produced per fission event, λ_d the decay constant, and C_d the precursor concentration for precursor family d.

The following expression was derived for the precursor solution vector:

$$\delta^{\pm}_{k,d} = \begin{bmatrix} C_d \pm \Delta t\, S_d/2 \\ C_d u_x + \frac{-8D_p \pm \Delta t}{8}\partial_x C_d \\ C_d u_y + \frac{-8D_p \pm \Delta t}{8}\partial_y C_d \\ C_d u_z + \frac{-8D_p \pm \Delta t}{8}\partial_z C_d \\ 0,\ k = 5,6,7 \end{bmatrix} \tag{3.27}$$

Where the heat source is replaced by the delayed precursor source S_d and the convective terms are present in the second to fourth terms, since the precursors are affected by convection. D_p is the delayed precursor diffusion constant and the delayed precursor source is defined as:

$$S_d = -\lambda_d C_d + \frac{\beta_d}{k_{eff}} \sum_g \left(\nu\Sigma_f\right)_g \phi_g \tag{3.28}$$

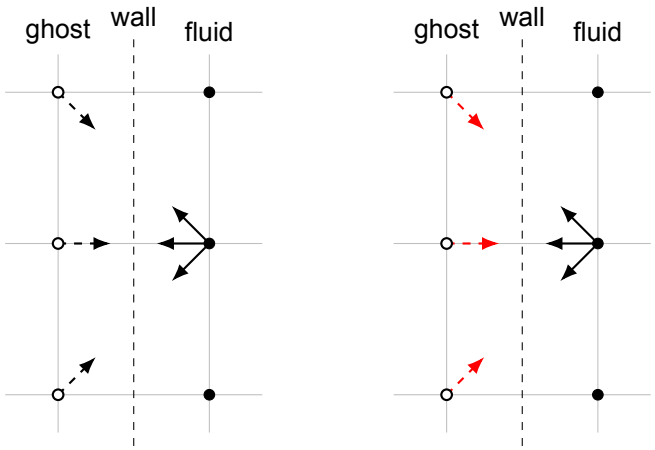
Where β_d is the delayed neutron fraction of delayed precursor family d. The first term on the right-hand side is the decay of precursors, while the second term is the production of precursors by fission.

3.3. Boundary Conditions

During the streaming step of the LBM algorithm, the populations streaming in from outside the domain are unknown. This is were boundary conditions come in: using the boundary conditions, the external populations flowing into the domain are determined.

Computationally, the external populations are programmed by introducing a layer of ghost nodes around the whole lattice. In these ghost nodes, the external populations are stored and updated for each streaming step.

There exists a vast number of techniques to enforce Dirichlet, Neumann, or Robin boundary conditions in the LBM scheme. In this research, the simple and straightforward halfway bounce-back (HBB) and anti bounce-back (ABB) techniques are used to enforce Dirichlet and Neumann boundary conditions for the different physical fields. The HBB and ABB boundary techniques are simple in their application; for both techniques, the outgoing populations 'stream' to a ghost node, where their velocity is reversed. For the ABB technique, the population is also inverted during the reflection. The HBB and ABB techniques are illustrated in figure 3.3 [23].



(a) The halfway bounce-back (HBB) technqiue visualized.

(b) The anti bounce-back (ABB) technique visualized.

Figure 3.3: Visualization of the boundary techniques used in this research. The fluid populations are marked by black, filled arrows, reflected populations are marked by dashed arrows, and inverted populations are marked by red arrows. The illustrations are inspired by [23].

3.3.1. Periodic Boundaries

Periodic boundaries are implemented by copying the distribution function at the edge of the periodic boundary to the ghost node at the other side of the domain. For a domain with N lattice nodes in the x direction, this means that the populations at x_1 are copied to x_{N+1}, and the x_N populations are copied to x_0. This is illustrated in figure 3.4 [23].

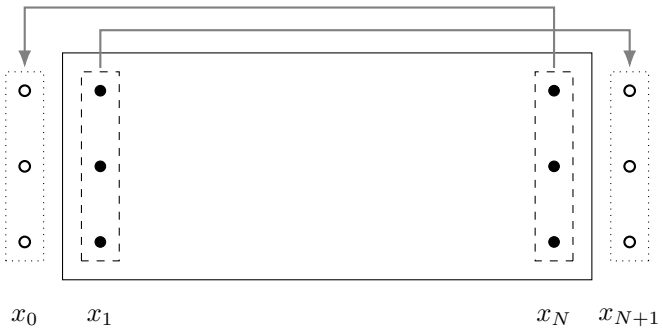


Figure 3.4: Implementation of periodic boundary conditions along the x-axis. Fluid nodes are denoted by filled black circles and ghost nodes are denoted by black, empty circles. Arrows denote the duplication direction. Illustration inspired by [23].

3.3.2. Momentum Boundaries

In this research, the walls of the simulatioN-domain were treated as a no-slip boundary. In the LBM, a no-slip boundary is implemented for the momentum equations using the HBB technique. For stationary walls, this results in [23]:

$$f_j(\boldsymbol{r}_b + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}_b, t) \tag{3.29}$$

where j is the inverse direction of i, such that $\boldsymbol{c}_j = -\boldsymbol{c}_i$. For moving walls, equation 3.29 is modified to [23]:

$$f_j(\boldsymbol{r}_b + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}_b, t) - 2\omega_i\rho_w\frac{\boldsymbol{c}_i \cdot \boldsymbol{u}_w}{c_s^2} \tag{3.30}$$

where ρ_w and $\boldsymbol{u}_w$ are the density and velocity at the wall $\boldsymbol{r}_w = \boldsymbol{r}_b + \frac{1}{2}\boldsymbol{c}_i\Delta t$

3.3.3. Heat, Neutronics, and Delayed Precursor Boundaries

The heat equation, neutron diffusion equation (NDE), and the precursor advection-diffusion equation (ADE), are all types of advection-diffusion equations. In this research, Dirichlet boundary conditions are used for the NDE and Neumann boundary conditions are used for the heat equation and precursor ADE. For advection-diffusion equations, Dirichlet and Neumann boundary conditions are enforced using (modified versions of) the ABB boundary technique. The general expression for the ABB technique, for stationary walls is [51]:

$$f_j(\boldsymbol{r}_b + \boldsymbol{c}_i\Delta t, t + \Delta t) = -f_i(\boldsymbol{r}_b, t) + 2\omega_i C_w \tag{3.31}$$

here again, j is the inverted direction of i. C_w is the value of the physical quantity (enthalpy, neutron flux, or precursor density) prescribed at the wall.

To enforce Dirichlet boundary conditions, C_w is equal to the Dirichlet value at the wall (e.g. the enthalpy at the wall for a fixed temperature boundary). In this research, only neutronics have Dirichlet boundary conditions, namely the vacuum boundary condition for the NDE. The vacuum boundary condition for the NDE prescribes a zero flux at an extrapolated distance $\tilde{x}_b$ from the boundary rather than at the wall, so interpolation is required to determine the neutron flux at the wall. The interpolation is shown in figure 3.5. Here, the wall flux ϕ_w is determined by interpolating between the known neutron flux just inside the domain ϕ_0, located half a lattice spacing from the wall, and the extrapolated neutron flux ϕ_{extr}, spaced at one extrapolated boundary $\tilde{x}_b$ from the wall. The extrapolated neutron flux is by definition equal to zero and the extrapolated boundary length is equal to 2.1312 times the neutron diffusion constant D_g.

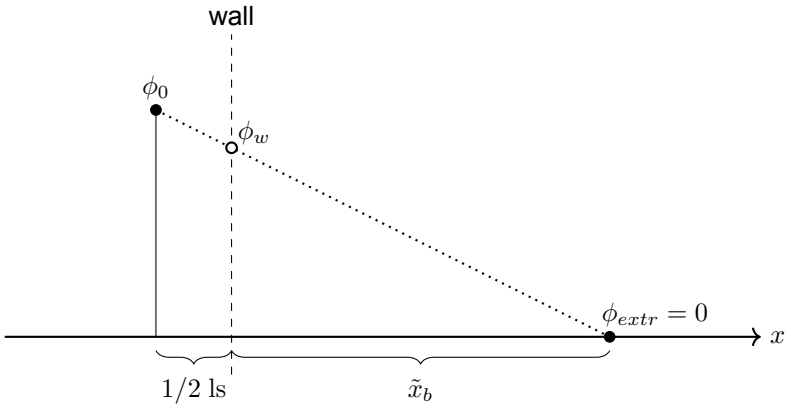


Figure 3.5: Visualization of the linear interpolation to determine the neutron flux at the wall (ϕ_w) for the vacuum boundary condition of the NDE. The neutron flux just inside the domain (ϕ_0), located half a lattice spacing from the wall, and the extrapolated neutron flux (ϕ_{extr}), spaced at one extrapolated boundary length $\tilde{x}_b$ from the wall, are used for the interpolation. The extrapolated boundary length is defined as 2.1312 times the neutron diffusion constant D_g.

Using this interpolation, it can be derived that the neutron flux at the wall ϕ_w is described by:

$$\phi_w = \frac{\tilde{x}_b}{1/2 + \tilde{x}_b}\phi_0 \tag{3.32}$$

Where the extrapolated boundary $\tilde{x}_b$ is expressed in lattice spacings (ls).

The heat and delayed precursor fields both have Neumann boundary conditions of the form $\frac{\partial h}{\partial \boldsymbol{n}} = 0$ and $\frac{\partial C_d}{\partial \boldsymbol{n}} = 0$, where $\boldsymbol{n}$ is the normal of the boundary. Neumann boundary conditions are enforced by setting the physical wall quantity C_w equal to the macroscopic value just inside the domain (C_0). For the enthalpy and precursor fields that translates to setting C_w to the enthalpy just inside the domain (h_0) and delayed precursor concentration just inside the domain ($C_{0,d}$) [51].

For moving walls, a correction to the ABB technique is required. This is only necessary for the physical quantities that are affected by convection (heat and delayed precursors in this research). For walls moving with velocity $\boldsymbol{u}_w$, equation 3.31 is corrected by [51]:

$$f_j(\boldsymbol{r}_b + \boldsymbol{c}_i\Delta t, t + \Delta t) = -f_i(\boldsymbol{r}_b, t) + 2\omega_i C_w \left[1 + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u}_w)^2}{2c_s^4} - \frac{|\boldsymbol{u}_w|^2}{2c_s^2}\right] \tag{3.33}$$

3.4. Lattice Conversion

To standarize computations, the simulation parameters are converted from physical units to lattice units, in a way similar to the dimensionless numbers from section 2.1.3. For instance, the spacing between neighbouring lattice points is defined as one ‘lattice spacing’ (ls), and the time between iterations as one ‘lattice time’ (lt). As with physical units, these lattice units can be combined to express other units such as velocity ($\mathrm{ls\,lt^{-1}}$). When a variable is expressed in lattice units, an asterisk subscript is added (e.g. Q^*). When it is emphasized that a variable is expressed in physical units, a ‘ph’ subscript is added (e.g. Q_{ph}).

When choosing lattice units, the same fluid behaviour needs to be simulated, i.e. the dimensionless numbers should be the same whether they are computed in physical or lattice numbers, i.e.:

$$\mathrm{Re} = \frac{U_{ph}L_{ph}}{\nu_{ph}} = \frac{U^*L^*}{\nu^*} \tag{3.34}$$

This can be achieved by defining the lattice units for the SI base units (time, space, temperature, and mass) and expressing all other units in these lattice units.

To convert from physical units to lattice units and back, conversion factors C_Q are used. The conversion factor is defined as the ratio of the physical value to the lattice value. For example, the conversion factor for space is defined as

$$C_x = \frac{L_{ph}}{L^*} = \frac{L_{ph}}{N} \tag{3.35}$$

where L_{ph} is the physical length of a given axis, and N the number of grid points of that given axis. Conversion factors for non-SI base units can be derived by combining the SI base conversion factors, for example the conversion factor for viscosity:

$$C_\nu = \frac{C_x^2}{C_t} \tag{3.36}$$

3.5. Scheme for Solving the Coupled System

The timescales of the flow, heat, neutronics, and precursor field vary. The timescale of the neutronics field is a few orders of magnitude smaller than the timescales of the flow, heat, and precursor fields, which have comparable timescales. To solve the coupled system, the simulatioN-domain was split in a long-timescale thermal-hydraulics domain (TH-domain), which simulates the flow, heat, and precursors, and a short-timescale neutonics domain (N-domain) simulating the neutronics.

Since the neutronics evolve on a timescale many times smaller than the thermal-hydraulics, in practice the neutronics will converge for each thermal-hydraulics timestep. This is implemented in the solving scheme by letting the N-domain run to convergence for each TH-domain LBM cycle. This solving scheme is thus structured as a thermal-hydraulics solver (the outer iteration) with an embedded neutronics solver (the inner iteration).

Since the two domains are coupled, between LBM cycles there is a data exchange required. Macroscopic quantities such as temperature and precursor concentration are transferred from the TH-domain to the N-domain, and the fission source and heat source is transferred back from the N-domain to the TH-domain. The solving scheme is illustrated in figure 3.6.

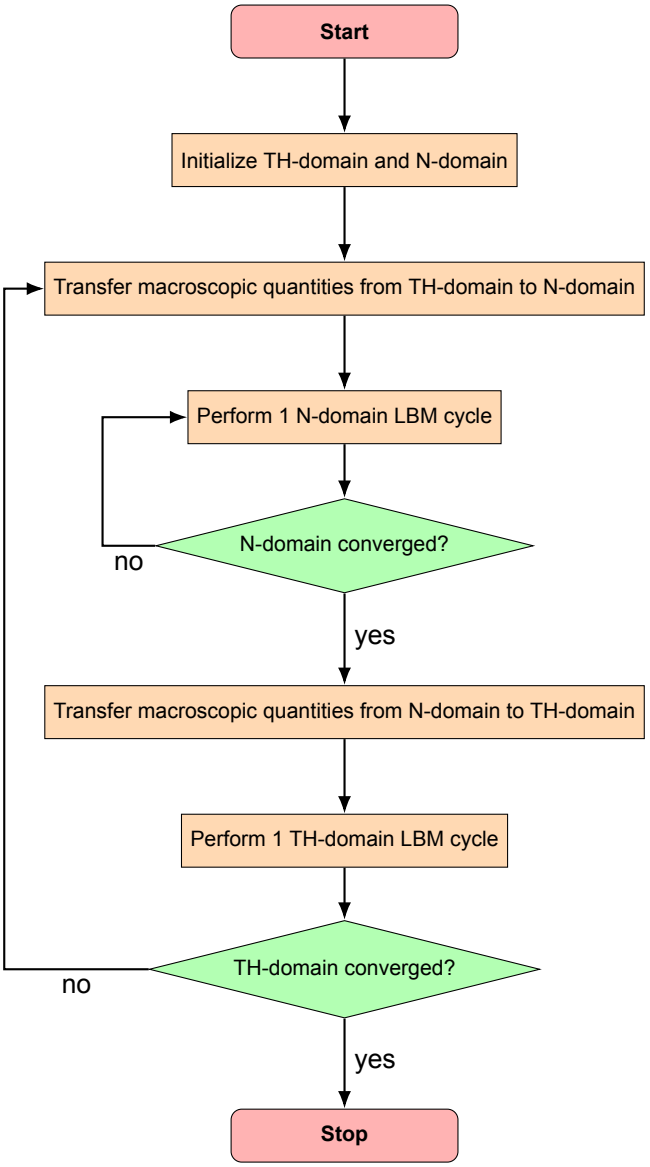


Figure 3.6: General solving scheme used to simulate the coupled multiphysics problem.

3.6. Predictor-Corrector Quasi-Static Method

The transient behaviour of the system is modeled using the predictor-corrector quasi-static method (PCQSM), discussed in more detail in section 2.3.3. The PCQSM is implemented using three timescales: a big timestep Δt_B, a medium timestep Δt_M, and a small timestep Δt_S. The simulation is initialized on a critical, steady state solution. As described in section 2.3.3, the neutron flux is factorized in a slowly changing shape function $\tilde{\phi}(\boldsymbol{r}, t)$ and a fast changing amplitude function $n(t)$, both of which need to be solved.

First, the steady-state LBM algorithm is used to compute the shape function over the big timestep Δt_B. Then, the point-kinetics (PK) parameters ρ, β_d, and Λ are computed over medium timestep Δt_M intervals, where the old and new shape functions are linearly interpolated to determine the shape function at the medium timestep. Lastly, the PK equations in equations 2.42 and 2.43 are solved over small timesteps Δt_S using the forward Euler method, resulting in the following expressions [47]

$$n(t + \Delta t_s) = n(t) + \left[\frac{\rho - \beta_{tot}}{\Lambda} n(t) + \sum_d \lambda_d m_d(t) \right] \Delta t_s \tag{3.37}$$

$$m_d(t + \Delta t_s) = m_d(t) + \left[\frac{\beta_d}{\Lambda} n(t) - \lambda_d m_d(t) \right] \Delta t_s \tag{3.38}$$

The initial values for the PK equations are given by [11]

$$n(0) = 1 \tag{3.39}$$

$$m_d(0) = \frac{\beta_d}{\lambda_d \Lambda} \tag{3.40}$$

When after iterating over all medium and small timesteps the big timestep Δt_B is reached, the neutron flux is computed by multiplying the shape function by the amplitude function. A schematic showing the timescales used in the PCQSM is shown in figure 3.7.

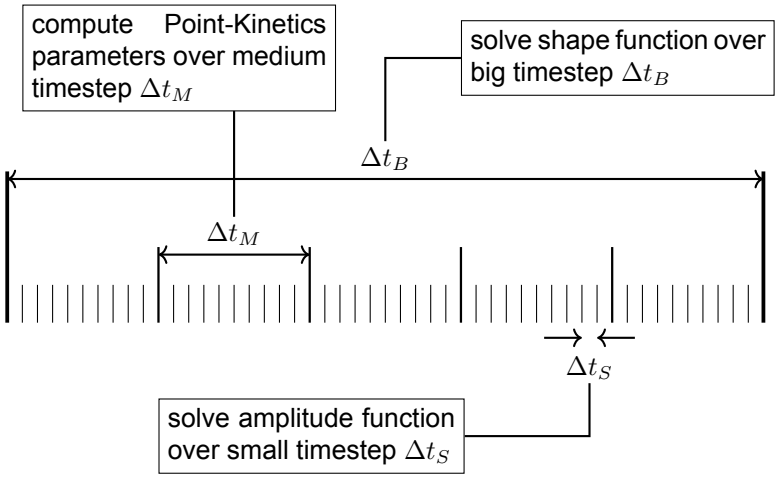


Figure 3.7: Schematic showing the different timescales involved in the PCQSM.

The big timestep Δt_B is chosen to be equal to the thermal-hydraulics timestep, which is in the order of 10^{-3} to 10^{-4} s. The medium and small timesteps Δt_M and Δt_S are chosen sufficiently small such that iteratively solved PK equations yield the same results as for smaller timestep values. The medium and small timestep values are determined on a trial-by-error basis, and are usually 2 to 10 medium timesteps per big timestep, and 10 to 100 small timesteps per medium timestep.

The modified neutronics solving scheme incorporating the PCQSM is shown in figure 3.8. Here, the first block is the same FM-LBM algorithm that is used to compute the steady-state neutronics. All the other blocks are added to compute the amplitude function using the point-kinetics equations.

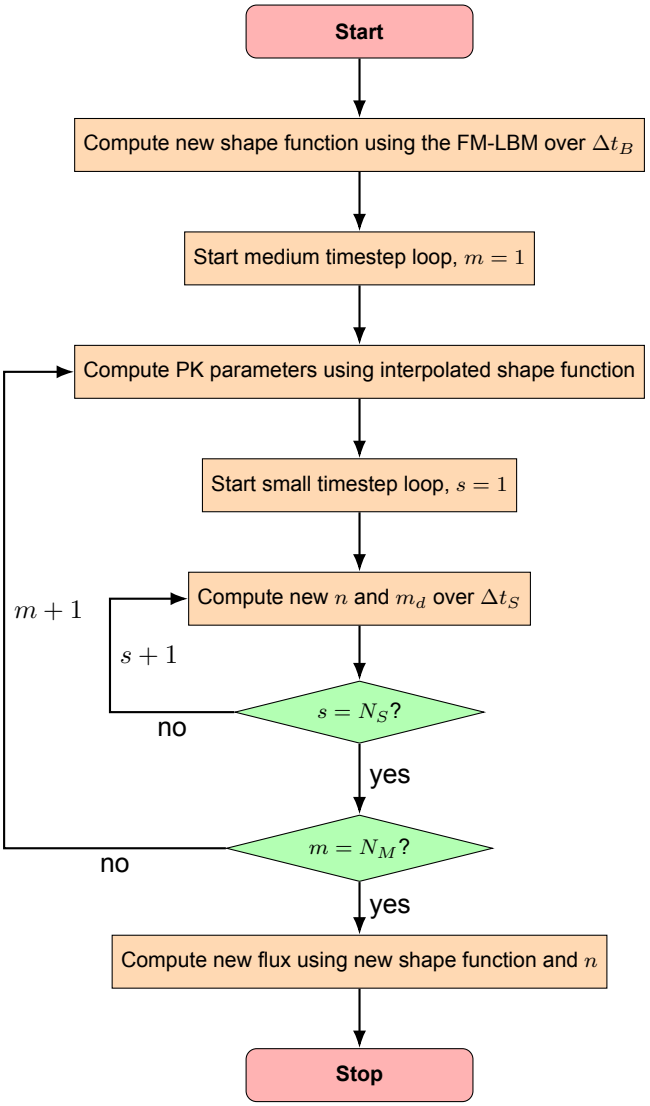


Figure 3.8: Schematic view of the implementation of the PCQSM for the transient neutronics in the code. N_M and N_S are the number of medium and small timesteps, respectively.

3.7. GPU Acceleration

Due to the localized nature of the collision and boundary condition enforcement steps of the FM-LBM, the FM-LBM algorithm is well suited for parallelization. Since the operations are computationally small, the cores of a GPU are sufficient to execute these operations.

3.7.1. Julia-CUDA

In this research, the Julia programming language is used ot parallelize the FM-LBM algorithm on the GPU. Julia uses a just-in-time compiler and high-performing type inference system, giving it computational performance comparable to statically typed languages like C and Java [24]. In Julia, the CUDA package is available, which allows for the writing of GPU kernels and their execution. In Julia-CUDA programming, it is also possible to assign the level of GPU memory to be used, increasing the control and optimization of GPU programming. These facts, combined with the ease of implementation due to the high-level syntax make Julia an ideal programming language for this research.

For the most optimal use of the GPU, the algorithm should be split in small functions - called kernels - that perform only a singular task. For each physical field five kernels are introduced: an initialization kernel, a collision kernel, a boundary condition kernel, a propagation kernel, and a convergence kernel. These kernels are executed at the start of the program (initialization) or in a loop (the rest). A schematic overview of the GPU-accelerated FM-LBM algorithm is shown in figure 3.9.

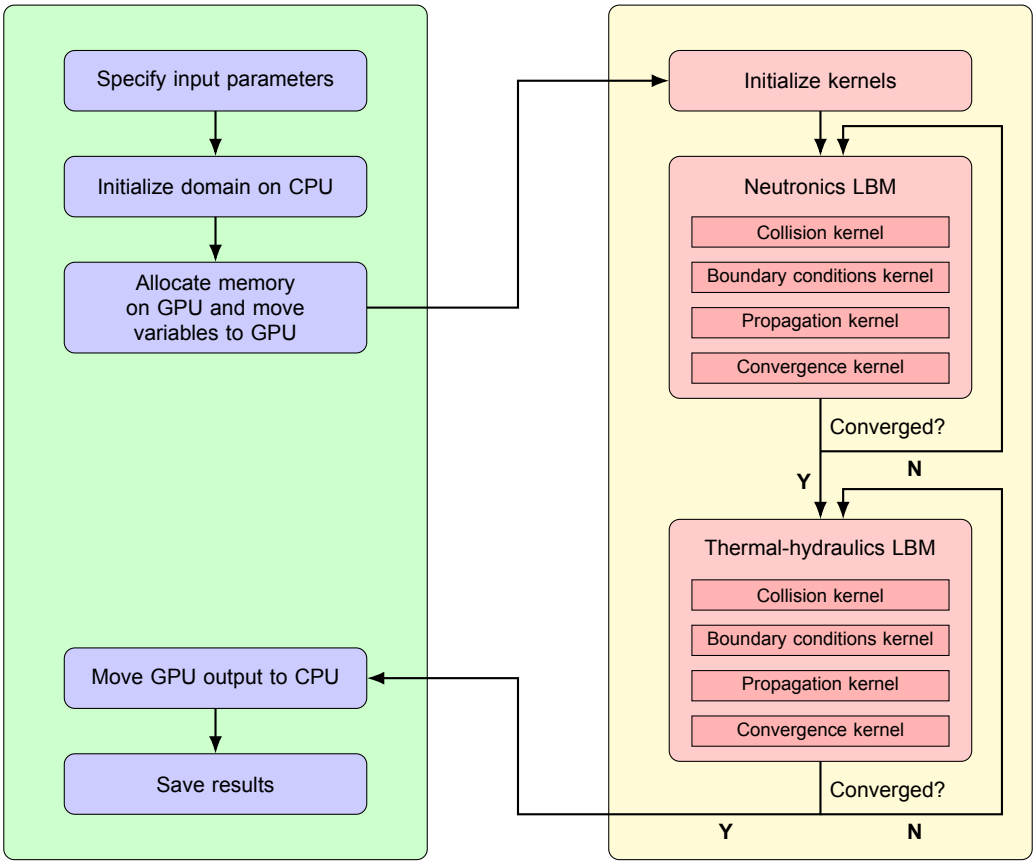


Figure 3.9: Schematic of the GPU-accelerated FM-LBM algorithm. The figure shows which processes take place at the CPU or GPU, and how the processing units communicate.

3.7.2. Race Conditions

One speaks of race conditions when two threads access and modify the same memory address, leading to invalid numerical values. Race conditions rarely result in an error, complicating their detection. The risk of race conditions are the largest during the propagation step of the algorithm, since all the distribution functions stream to the neighboring nodes. To prevent race conditions, an extra temporary

array to store distribution functions is used. During propagation, the distribution function array is read from and the temporary distribution array is written to. After propagation, the temporary distribution array is copied to the distribution array. Due to the use of an extra array this approach uses twice as much memory, however it prevents the occurrence of race conditions.

Race conditions can also occur when a new kernel is executed before the old kernel is finished, for example the boundary condition kernel is executed before the collision kernel is finished. To prevent this from happening, after each kernel launch a synchronization function is called, to force the completion of all kernels before new kernels are launched.

3.7.3. Performance Gains

To optimize GPU usage, several techniques are incorporated into the FM-LBM algorithm to improve its computational efficiency. The two most significant techniques, memory coalescence and shared memory usage, will be discussed in this section.

Memory coalescence is a technique to flatten multidimensional arrays to 1D arrays. 1D arrays are quicker to access and modify, as they incur less computational overhead on the GPU. When flattening a multidimensional array, the order in which the axes is a non-trivial decision. By ordering the axes in such a way that frequently accessed memory locations are located close to each other in the memory, the computational speed can be further improved. In figure 3.10 the two types of flattening of a 2D array are illustrated. Depending on the kernel used, type 1 or type 2 sorting is quicker: for a collision kernel, which uses information over all the directions on a single lattice node, type 2 sorting is quicker; for a propagation kernel, which propagates a direction over the whole lattice at a time, type 1 sorting is quicker. The order of flattening was adapted to the kernels to achieve additional computational speed.

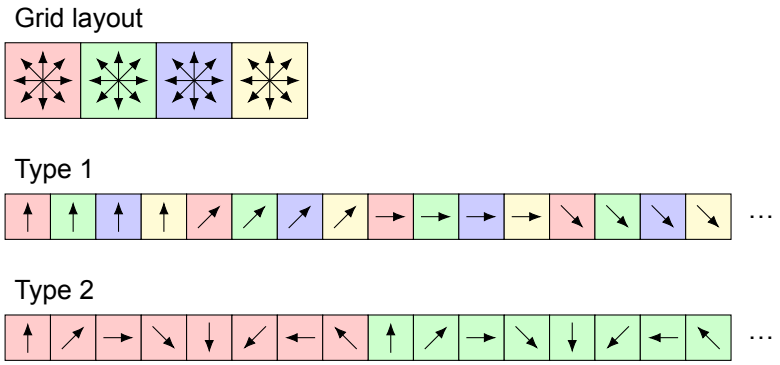


Figure 3.10: Illustration of the two types of flattening for a 2D array. For type 1, the 2D array is flattened over the direction-axis, while for type 2 the 2D array is flattened over the x-axis.

As discussed in section 2.6.2, NVIDIA GPUs have a layered memory hierarchy, where memory closer to individual cores is quicker to access at the cost of smaller memory size and more restrictive access. Julia-CUDA allows low-level management of this memory for each kernel, enabling the assignment of variables to global, shared, or register memory. Register memory was used as much as possible for variables that existed only during the kernel lifetime. Shared memory was more limited; it is quicker to access than global memory; however, due to shared memory only being accessible by threads in the same block, its use to store data spanning multiple lattice nodes was limited. The only use found for shared memory was for the storage of solution vectors of the collision kernel, since these were too large to store in register memory. Global memory usage was limited to storing constants and distribution functions, since these needed to be accessed and updated by all lattice nodes during the simulation.

4

Validation of Single-Physics Models

To validate the accuracy of the numerical methods described in the previous chapter, the multiphysics tool is tested against a benchmark. The Tiberga benchmark was selected to validate the tool, since this benchmark gradually couples the fields and supplies observables for each coupling step, which helps with error identification. The Tiberga benchmark case studies a 2-dimensional lid-driven cavity filled with a molten salt mixture.

Section 4.1 describes the details and structure of the Tiberga benchmark case. The specifics of two previously developed LBM multiphysics tools, which were also validated with the Tiberga benchmark, are discussed in section 4.2. The error quantification methods employed are explained in section 4.3. The results for the single field simulations for momentum, neutronics, and temperature are presented and compared to the benchmark in sections 4.4, 4.5, and 4.6, respectively. Additionally, heatmaps of all observables for each step can be found in appendix B.

All steps from the Tiberga benchmark case in this chapter are steady-state problems. The power method as described in section 2.3.2 is used to compute the neutronics criticality eigenproblem.

4.1. Description of Tiberga Benchmark

The Tiberga benchmark was developed to assess the physics-coupling capabilities of multiphysics MSFR codes [37]. The benchmark compares codes developed during the SAMOFAR project developed by four institutions: Delft University of Technology, Paul Scherrer Institute, Politecnico di Milano, and CNRS-Grenoble.

In the benchmark, a step-by-step approach is taken, where the different physical fields are gradually coupled. In the initial phase (phase 0) the momentum, heat, and neutronics fields are simulated separately until a steady-state solution is found. In the next phase (phase 1) the momentum, heat, and neutronics fields are gradually coupled, until all couplings are into effect and a steady-state solution is found. In the last phase (phase 2) the transient behaviour of the system is studied by perturbing system parameters. The steps in each phase are numbered (i.e. step 1.1, step 1.2, etc.) and for each step the Tiberga benchmark provides the results of observables from the codes developed by the four institutions, against which the results from this research will be compared. The results of the four institutions are labeled by their respective names: 'CNRS' for *Centre national de la recherche scientifique-Grenoble*, 'PoliMi' for *Politecnico di Milano*, 'PSI' for *Paul Scherrer Institute*, and 'TUD' for *Delft University of Technology*. From step 0.2 onwards, the neutron transport equation discretization is also specified if applicable, for example: CNRS-SP1 and CNRS-SP3. The results from this study are labeled by 'LBM', standing for lattice Boltzmann method.

4.1.1. Description of the Molten Salt System

The domain of the problem is a 2-dimensional $2\ \mathrm{m}$ by $2\ \mathrm{m}$ cavity filled with molten salt at an initial temperature of $900\ \mathrm{K}$. Observables of the different steps are compared along the centerlines AA' and BB', which are shown in figure 4.1. A no-slip boundary condition is applied to the flow field, where all

wall except for the top lid are stationary. The top lid moves with a velocity U_{lid} to the right. All walls are adiabatic and a homogeneous Neumann boundary condition is applied to the delayed precursors (i.e. $\frac{\partial C_d}{\partial \boldsymbol{n}} = 0$). The domain is treated as a homogeneous bare reactor, so vacuum boundary conditions are applied to the neutron flux at all boundaries. Salt cooling is modeled using a volumetric heat sink:

$$q'''(\boldsymbol{r}) = \gamma(T_{ref} - T(\boldsymbol{r})) \tag{4.1}$$

where γ is the volumetric heat transfer coefficient, T_{ref} the reference temperature of $900\ \mathrm{K}$, and $T(\boldsymbol{r})$ the local temperature. All steady-state steps (phases 0 and 1) are criticality eigenvalue problems where the reactor power is normalized to the reference power P.

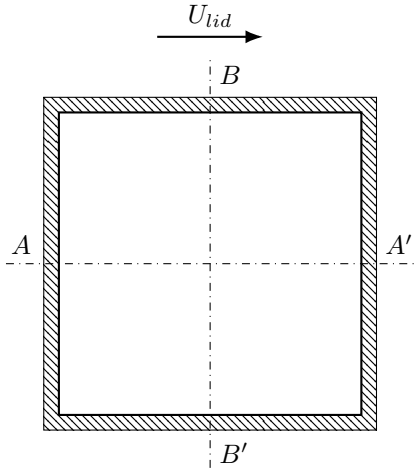


Figure 4.1: Schematic of the benchmark domain. The top lid moves with U_{lid} to the right, all other walls are stationary. The cavity is thermally insulated and surrounded by vacuum. Observables are compared along the AA' and BB' centerlines. Figure is inspired by [37].

The fuel salt used in the benchmark is a $\mathrm{LiF\text{-}BeF_2\text{-}UF_4}$ mixture, the composition of which is shown in table 4.1. Fluid properties like viscosity are taken constant over temperature and space. The fluid properties are shown in table 4.2. The salt flow is considered to be incompressible and laminar, and buoyancy is modeled using the Boussinesq approximation. No neutronics model is prescribed. Turbulence, 3D geometries, and decay heat are all neglected and avoided since other benchmarks exist to study these complexities. These choices were made to make the Tiberga benchmark fairly general and suitable for codes in early stages of development [37].

Table 4.1: Fuel salt composition [37].

Isotope	$^{6}\mathrm{Li}$	$^{7}\mathrm{Li}$	$^{9}\mathrm{Be}$	$^{19}\mathrm{F}$	$^{235}\mathrm{U}$
Atomic fraction (%)	2.11488	26.0836	14.0992	56.3969	1.30545

Table 4.2: Salt thermodynamic and fluid properties [37].

Property	Units	Value
Density	$\mathrm{kg\,m^{-3}}$	2.0×10^{3}
Kinematic viscosity	$\mathrm{m^2\,s^{-1}}$	2.5×10^{-2}
Volumetric heat capacity	$\mathrm{J\,m^{-3}\,K^{-1}}$	6.15×10^{6}
Thermal expansion coefficient	$\mathrm{K^{-1}}$	2.0×10^{-4}
Prandtl number	–	3.075×10^{5}
Schmidt number	–	2.0×10^{8}

The nuclear data was taken from the JEFF-3.1 library at a temperature of $900\,\mathrm{K}$ [1]. The nuclear data for the six neutron groups and eight precursor families was generated using Serpent [26]. The definitions of the neutron groups can be found in table 4.3. The complete set of neutronics and precursor data can be found in appendix A.

Table 4.3: Definition of neutron energy groups [37].

Group number, g	Upper energy bound [MeV]
1	2.000×10^{1}
2	2.231×10^{0}
3	4.979×10^{-1}
4	2.479×10^{-2}
5	5.531×10^{-3}
6	7.485×10^{-4}

4.2. Description of previous LBM studies

Coco Polderman and Tom Entes, two previous students of Martin Rohde's research group, have developed multiphysics tools using the lattice Boltzmann method for the Tiberga benchmark case [28, 12]. Different neutronics implementations and different LBM algorithms were used in these codes. Due to the limitations or advantages offered by these architecture differences, different grid sizes and simulation parameters were used compared to the Tiberga benchmark and this research. Their research offers the opportunity to not only compare the LBM algorithm developed in this research with the Tiberga benchmark, but also with the other LBM algorithms. In this research, only the reactivity (differences) between the LBM codes will be compared. The LBM results will be listed as LBM-Polderman, LBM-Entes, and LBM-Pijls, for the results from Polderman's, Entes', and this research, respectively. The specifics of both LBM codes are discussed below.

Polderman's code simulates both the neutronics and thermal-hydraulics with the LBM. In Polderman's code, the neutronics and thermal-hydraulics were simulated in one domain. The code was written in accelerated Python (numba) and no GPU parallelization was implemented. All simulations were performed on a 200×200 grid. For all four physical fields, the BGK collision operator was used and Dirichlet and Neumann boundary conditions were enforced using the anti bounce-back and halfway bounce-back techniques. For the neutronics, the non-equilibrium extrapolation scheme (NEES) was used. The NEES is a wet-node boundarry technique, meaning that the boundary is set at the node itself instead of halfway between two nodes. Using the NEES, the neutron flux at the ghost node was set to zero, to approximate the extrapolated boundary condition of the vacuum boundary for the neutron diffusion equation. Since the BGK collision operator is limited in the physical values it can stably simulate, Polderman significantly reduced the Prandtl and Schmidt numbers. Step 0.3 was simulated using a Prandtl number of 600, step 1.1 using a Schmidt number of 400, step 1.2 a Prandtl and Schmidt number of 200, step 1.3 a Prandtl and Schmidt number of 300, and step 1.4 a Prandtl and Schmidt number of 100. The transient neutronics were simulated using the PCQSM.

Entes' code simulates the thermal-hydraulics and precursors using the LBM and simulates the neutronics with the Phantom code developed by the TU Delft. This required the exchange of information between the two codes, and also interpolation since the Phantom code used a different grid than the LBM. The Phantom code is written in Fortran and the LBM code was written in Julia. The LBM code was parallelized on the GPU and the simulations were performed on a 501×501 grid. In the LBM code, the filter-matrix collision operator was used, and the boundary conditions were modeled using the anti bounce-back and reflective bounce-back boundary techniques. The use of the filter-matrix collision operator allowed for a greater range of physical parameters that can be stably simulated, so Entes' simulations were done with a Prandtl number of 1000 and a Schmidt number of 1500. The Phantom code simulates the neutron transport equation and uses a S_6 discretization. It can perform both steady-state and transient computations, so no special scheme was used for transient simulations.

4.3. Error Quantification

To compare the simulation results with the benchmark results, an error quantification method is used. For consistent comparisons of the steady-state simulations, the final neutron flux is normalized to the the reference power of $P_{ref} = 1\ \mathrm{GW}$.

The error of the simulated physical quantities Q along the horizontal and vertical centerlines is quantified using the following equation.

$$\varepsilon_Q = \sqrt{\frac{\sum_{i=1}^{N} \left(Q_{sim}(\boldsymbol{r}_i) - Q_{avg}(\boldsymbol{r}_i)\right)^2}{\sum_{i=1}^{N} Q_{avg}^2(\boldsymbol{r}_i)}} \tag{4.2}$$

Where ε_Q denotes the error in physical quantity Q, Q_{sim} the simulation results of the physical quantity, Q_{avg} the mean of all the benchmark studies, N the number of points to evaluate, and $\mathbf{r}_i$ the evaluation point. When the simulation results did not align with the benchmark grid, linear interpolation was used to match the evaluation points.

4.4. Step 0.1: Lid-Driven Cavity Flow

In the first step, the velocity field was simulated. While lid-driven cavity flows have already successfully been simulated using the filter-matrix lattice Boltzmann method, this step is still performed for completeness and to check if the code is implemented properly.

The relevant parameters for this simulation step are listed in table 4.4, given in both physical and lattice units. The physical values were converted to lattice units using the conversion factors described in section 3.4.

Table 4.4: Simulation parameters used for the simulation of step 0.1 of the Tiberga benchmark. The parameters are given in both physical and lattice units, the conversion of which is described in section 3.4.

Parameter	Physical value	Physical unit	Lattice value	Lattice unit
Δt_{TH}	1.0×10^{-3}	s	1.0	lt
Re	40	$-$	40	$-$
L	2.00	m	200	ls
ρ	2000	$\mathrm{kg\,m^{-3}}$	1.0	$\mathrm{lm\,ls^{-3}}$
ν	0.025	$\mathrm{m^2\,s^{-1}}$	0.25	$\mathrm{ls^2\,lt^{-1}}$
U_{lid}	0.5	$\mathrm{m\,s^{-1}}$	0.05	$\mathrm{ls\,lt^{-1}}$

In step 0.1, the momentum is source is a moving top lid, while the other walls are stationary. The observables are the horizontal and vertical velocity components u_x and u_y, measured along the centerlines AA' and BB'. The simulation results of these observables on a 200×200 grid are shown against the benchmark in figures 4.2 and 4.3.

In addition, the spatial convergence of the filter-matrix algorithm was analyzed. This was done by computing the discrepancy of the simulation results with the averaged benchmark results using equation 4.2 for a number of simulation grid sizes, the results of which are shown in figure 4.4.

Figure 4.4 shows that the horizontal velocity component converges to the benchmark results for increasing grid size. In the right plot, the results from the PSI study were omitted, since these showed significant deviations from the other benchmark results, especially for the vertical velocity component along the vertical centerline. However, in both plots the vertical velocity components show a minimum error around $N = 200$, with a slowly increasing error for increasing grid size. However, the overall error of all observables is small, below 1% for grid sizes $N > 100$, and below 0.5% for grid sizes $N > 200$. Due to the small error decrease with increasing grid size, and to save on computing time and memory usage, it was decided to simulate all consecutive steps on 200×200 grid sizes.

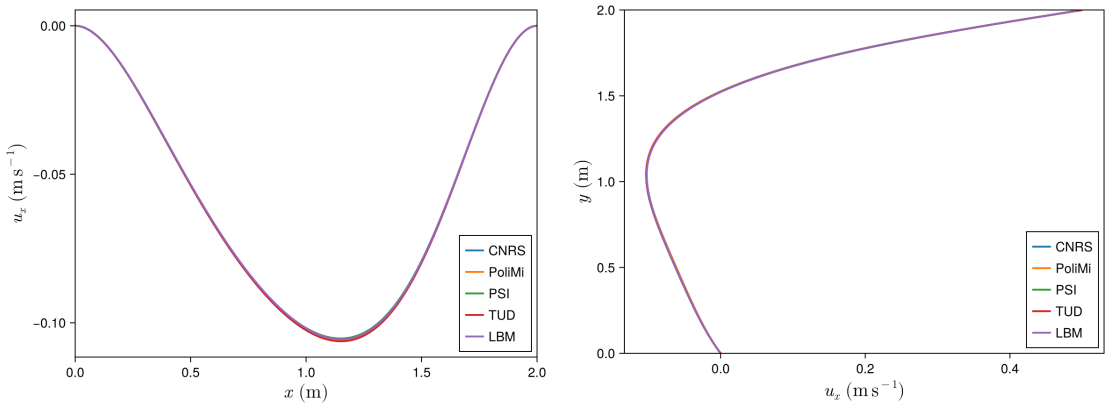


Figure 4.2: Simulation results for the horizontal velocity component for step 0.1 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the AA' (left) and BB' (right) centerlines. The results were obtained using the filter-matrix algorithm on a 200×200 simulation grid.

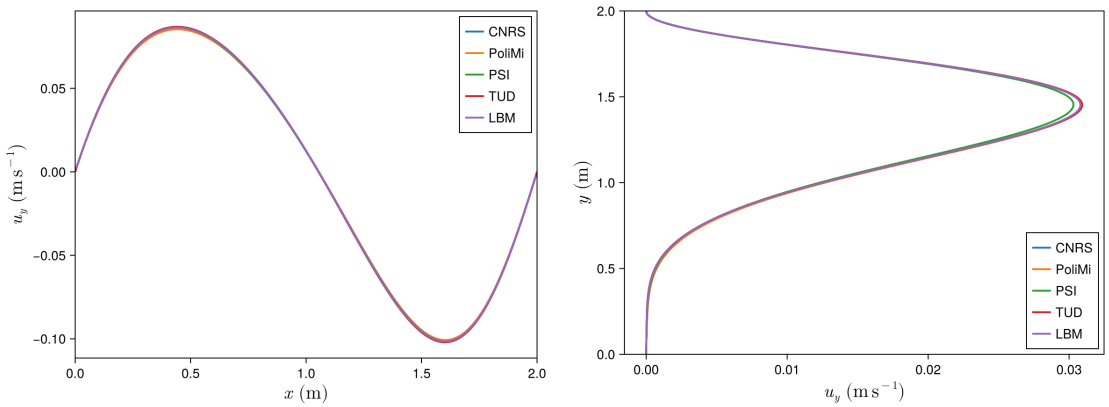


Figure 4.3: Simulation results for the vertical velocity component for step 0.1 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix algorithm on a 200×200 simulation grid.

4.5. Step 0.2: Neutronics

In the second step of the Tiberga benchmark the neutron flux and precursor concentration are simulated in a stationary fluid at a constant temperature of $900\ \mathrm{K}$. The observables of this step are the fission rate density $\int_E \Sigma_f \phi \,\mathrm{d}E$ along the horizontal centerline and the reactivity ρ.

The aim of this step is to verify the neutronics solution in simple, static-fuel conditions. Minor differences in the reactivity are reported due to different neutronics models and approximations adopted and are considered acceptable.

The simulation parameters used are listed in table 4.5.

Table 4.5: Simulation parameters used for the simulation of step 0.2 of the Tiberga benchmark.

Parameter	Physical value	Physical unit	Lattice value	Lattice unit
Δt_N	1.0×10^{-9}	s	1.0	lt
P_{ref}	1.0×10^{9}	W	5.0×10^{-14}	$\mathrm{lm\,ls^2\,lt^{-3}}$
T_{ref}	900	K	1.0	lT

The fission rate density of the simulation is plotted against the benchmark codes in figure 4.5 and the reactivities are reported in table 4.7. In the figure it can be seen that the LBM simulation is in good

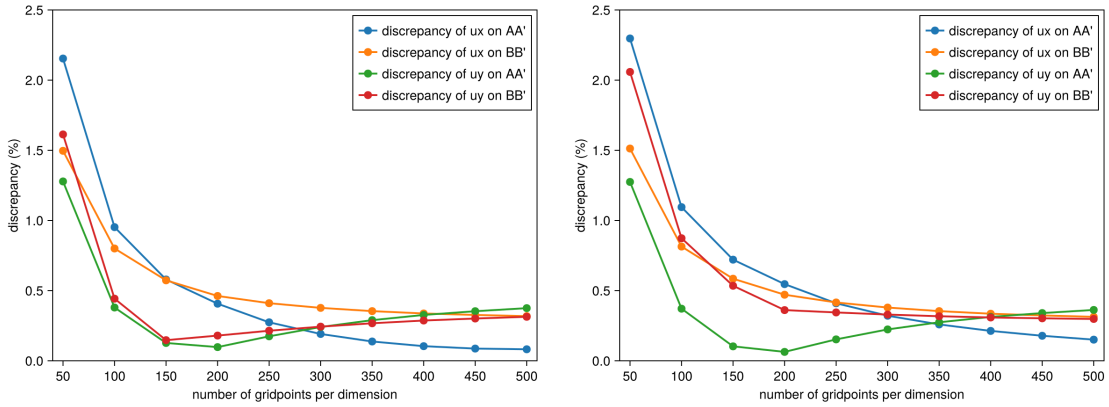


Figure 4.4: Discrepancies of the simulated horizontal and vertical velocity component along the horizontal and vertical centerlines compared to the benchmark results. The left plot includes all benchmark results, while in the right plot the PSI results are omitted since these showed significant deviations from the other benchmark results. The x-axis shows the number of grid point N per dimension. The discrepancies are computed with the average of the benchmark results and is shown in percentages.

agreement with the benchmark codes, which use both the neutron transport equation (CNRS and TUD) and the neutron diffusion equation (PoliMi and PSI). The reactivity of the LBM simulation is also in the same range as the reactivity of the benchmark codes and other LBM codes. These results validate the neutronics filter-matrix method and the neutron flux boundary treatment developed in this research.

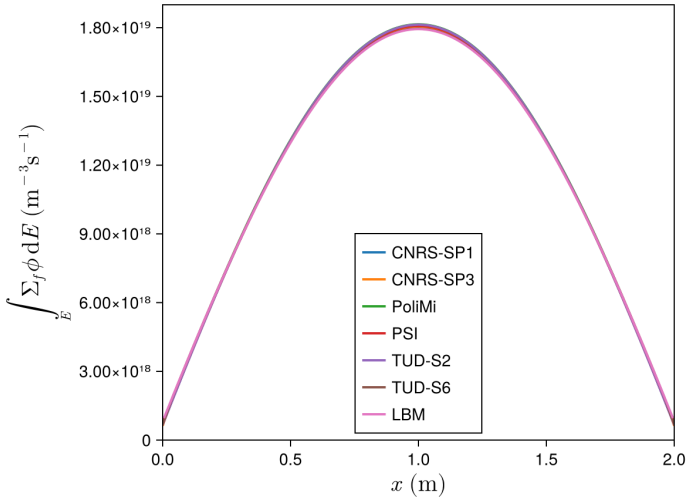


Figure 4.5: Fission rate density along the horizontal centerline of the benchmark codes and the simulation.

Code	ρ (pcm)
CNRS-SP_1	411.3
CNRS-SP_3	353.7
PoliMi	421.2
PSI	411.7
TUD-S_2	482.6
TUD-S_6	578.1
LBM-Polderman	520.6
LBM-Entes	578.1
LBM-Pijls	485.5

Table 4.7: Step 0.2 reactivity for the benchmark codes and the simulation.

4.6. Step 0.3: Temperature

In the third and last step of the single-physics Tiberga benchmark case the temperature is simulated. In this step, the convective and diffusive behavior of the temperature field are assessed, as well as the heat sources and sinks. Thermal convection is modeled by fixing the flow field from step 0.1. The heat source is modeled by using the neutron flux solution from step 0.2, and the heat sink is modeled by the volumetric heat sink. The heat source and sink is described in equation 3.24.

The observables in this step are the temperature distribution along the AA' and BB' centerlines. The simulation parameters are listed in table 4.8.

Table 4.8: Simulation parameters used for the simulation of step 0.3 of the Tiberga benchmark.

Parameter	Physical value	Physical unit	Lattice value	Lattice unit
Δt_{TH}	2.5×10^{-4}	s	1.0	lt
Pr	1200	$-$	1200	$-$
γ	1.0×10^{6}	$\mathrm{W\,m^{-3}\,K^{-1}}$	0.0703	$\mathrm{lm\,ls^{-1}\,lt^{-3}\,lT^{-1}}$
C_p	6.15×10^{6}	$\mathrm{J\,m^{-3}\,K^{-1}}$	1730	$\mathrm{lm\,ls^{-1}\,lt^{-2}\,lT^{-1}}$
T_{ref}	900	K	1.0	lT
α	2.08×10^{-5}	$\mathrm{m^2\,s^{-1}}$	5.21×10^{-5}	$\mathrm{ls^2\,lt^{-1}}$

In the Tiberga benchmark a very high Prandtl number of 3.075×10^5 is prescribed. To simulate such a high Prandlt number, an increased lattice viscosity (ν^*) is required for stability. This, in turn, requires a smaller lattice spacing to keep a constant Reynolds number, greatly increasing computational effort. Additionally, larger grids converge at a slower pace, compounding the computational costs.

Therefore, the simulations will be run at a lower Prandtl number to manage the computational costs. To understand the effect of a reduced Prandtl number, the heat equation in equation 2.4 is non-dimensionalized. The tilde notation indicates non-dimensional variables, as is outlined in section 2.1.3.

$$\mathrm{RePr}\left(\tilde{\boldsymbol{u}} \cdot \tilde{\boldsymbol{\nabla}} \tilde{T}\right) = \tilde{\boldsymbol{\nabla}}^2 \tilde{T} + \frac{1}{L\tau\alpha}\tilde{q} \tag{4.3}$$

Where L and τ are the characteristic length and timescale of the system and α the thermal diffusivity.

From equation 4.3 it can be seen that an increase in the Prandtl number results in convective transport and heat sink effects dominating over diffusive transport. The high Prandtl number in the benchmark implies that diffusion effects can be neglected. However, since the temperature gradients in the benchmark case are relatively small due to the volumetric heat sink being applied over the whole domain, it is expected that at lower Prandtl numbers the diffusive term will be negligible. The only notable temperature gradients are expected to arise from the shape of the power source and by convection. However, beyond a certain Prandtl number these diffusive effects will become negligible as well.

To study the effect of lowering the Prandtl number, different Prandtl numbers were simulated and the error along the vertical and horizontal centerlines with the benchmark results was computed with the help of equation 4.2. The results of these are shown in figure 4.6.

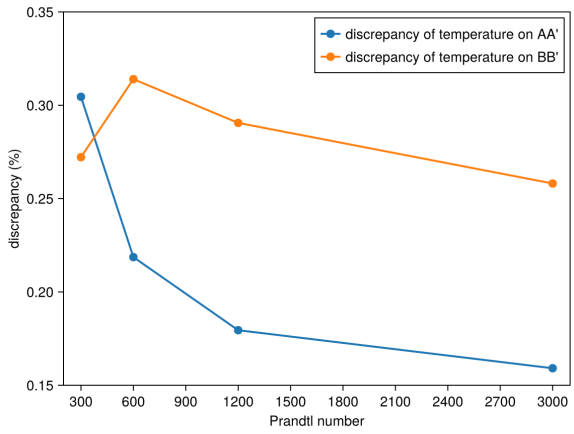


Figure 4.6: Discrepancy of the simulated temperature along the AA' and BB' centerlines compared to the benchmark results. The x-axis shows the Prandtl numbers of the LBM simulation. The discrepancies were computed with the average of the benchmark results and is shown in percentages.

It can be seen that for all Prandtl numbers the error is below 0.32%, which means that a relatively low Prandtl number of 300 is sufficient to neglect diffusive effects. By increasing the Prandtl number, the

error slowly decreases, especially for the temperature along the horizontal (AA') centerline. To get a more detailed view of the error in the temperature simulations, the error was computed per point along the centerlines for the Prandtl numbers. This is shown in figure 4.7. Here, it can be seen that the error is the largest at the boundaries, which is expected since the largest temperature gradients are found here. With increasing Prandtl number, the error at the boundaries decreases, which is expected since diffusive effects are more suppressed.

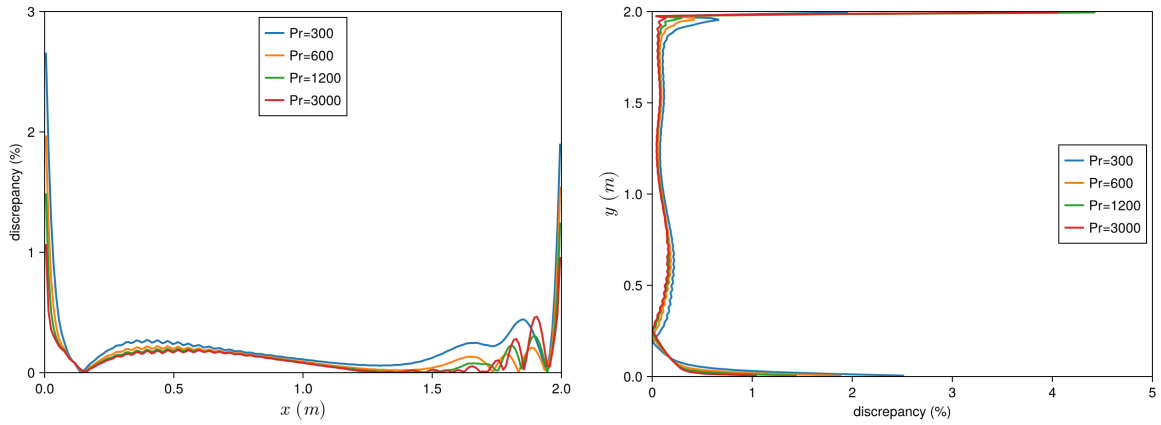


Figure 4.7: Pointwise discrepancy of the LBM simulation along the horizontal (left) and vertical (right) centerlines compared to the benchmark results.

By increasing the Prandtl number, the error in the domain decreases in most regions. Except for the $1.5 - 2.0$ m region along the horizontal centerline, where the error increases with increasing Prandtl number. This is because the fixed flow field of step 0.1 has a higher density in the top-right corner of the domain. This high density is caused by the discontinuity in the wall velocity: the top wall moves to the right, while the right wall is stationary. When simulating the heat transport, this results in a low-temperature region in the top-right corner that travels down due to the strong downward velocity.

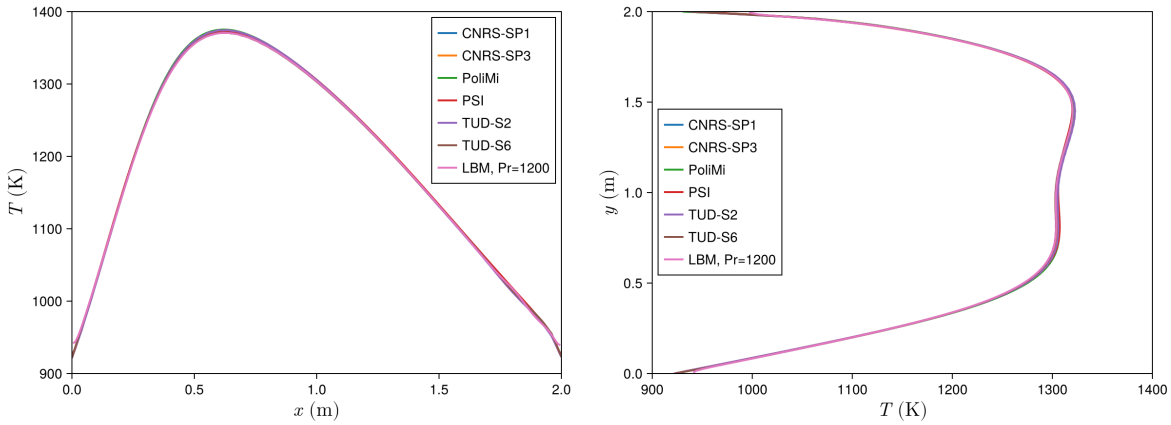


Figure 4.8: Simulation results of the temperature for step 0.3 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix algorithm on a 200×200 grid and with a Prandtl number of 1200.

In figure 4.8 the temperature along the centerlines is shown for a Prandtl number of 1200, and at the rightmost end of the horizontal centerline the effect of the low-temperature region in the corner can be seen as an oscillation in the $1.5 - 2.0$ m region. To increase the Prandtl number, the LBM timestep (Δt) was reduced, which increased the high-density region in the corner and therefore the instability of the temperature along the right wall.

In summary, the overall error decreases as the Prandtl number increases. However, the local error

increases with increasing Prandtl number due to instabilities, in particular along the right wall. Therefore, it was decided to use a Prandtl number of 1200 in further simulations to strike a trade-off between global and local error reduction.

4.7. Performance

To test the computational performance of the GPU-accelerated FM-LBM algorithm, the number of lattice updates per second (LUPS) was computed for the simulations. Due to the large number of updates per second, the computational performance is instead expressed in million lattice updates per second (MLUPS), which is defined as

$$\text{MLUPS} = \frac{N_{grid} N_T}{T} \times 10^{-6} \tag{4.4}$$

where N_{grid} is the number of grid points in the lattice, N_T the number of iterations, and T the simulation time in seconds. A lattice update is defined as the execution of all steps (collision, boundary conditions, propagation) for one LBM iteration at a single lattice point.

In figure 4.9 the computational performance of the previous outlined steps of the single physics phase are shown for increasing grid sizes. Step 0.1 simulated momentum for a lid-driven cavity, step 0.2 simulated the 6 neutron groups and 8 precursor families in a stationary fuel, and step 0.3 simulated the temperature using the flow field from step 0.1 as a convective input and the neutron distribution from step 0.2 as the heat source. All simulations were performed using a NVIDIA A-100 GPU using double-precision floating point numbers. The computational performance of the FM-LBM algorithm developed by Entes for the side-heated 3D cavity problem is shown alongside the results from this study for comparison [12]. In the side-heated 3D cavity problem, flow and thermal transport, including buoyancy, is simulated in a square cavity with a cold left wall, a hot right wall, and adiabatic top, bottom, front, and back walls.

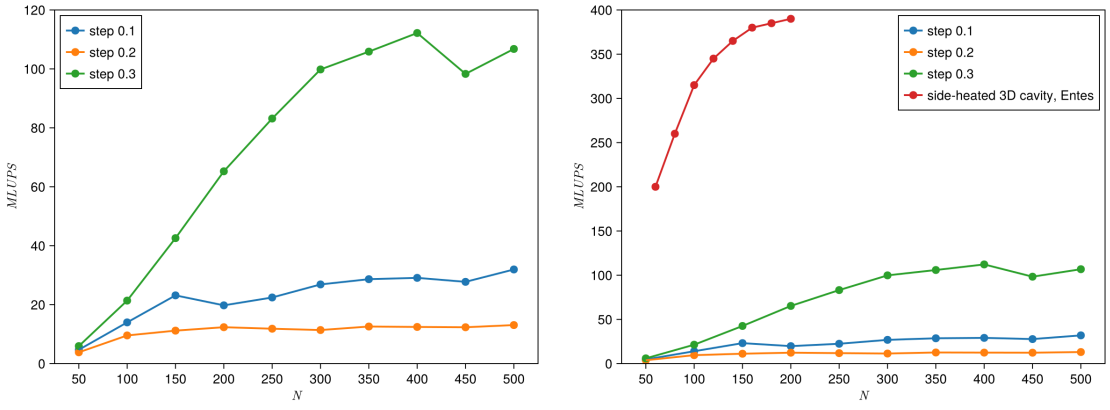


Figure 4.9: In the left figure, the computational performance of the GPU-accelerated FM-LBM algorithms developed for the single physics phase of the Tiberga benchmark for different grid sizes is plotted. In the right figure, the computational performance of the FM-LBM algorithm developed by Entes for the side-heated 3D cavity problem is shown alongside the computational performance in this study [12]. Here, N is the number of grid points along an axis, since the simulations take place in a 2-dimensional square grid $N_{grid} = N^2$.

All three steps in figure 4.9 show an increase in MLUPS as the grid size increases. This is expected since in theory the GPU-accelerated algorithm should not be limited by the grid size due to the parallelization. However, it can be seen that the MLUPS stops increasing or even decreasing with a certain grid size. This can be explained that at these grid sizes the memory access speed starts to play a limiting factor, thereby capping the MLUPS. It can also be seen that the highest maximum number of lattice updates varies greatly between the steps: step 0.1 reports a maximum of 32 MLUPS, step 0.2 of 13 MLUPS and step 0.3 of 112 MLUPS. This finding further reinforces the memory limitation explanation, since step 0.3 is the least memory intensive where one field is simulated using the $\mathrm{D3Q7}$ scheme, while step 0.2 is the most memory intensive simulating 14 fields each using the $\mathrm{D3Q7}$ scheme .

When the computational performance of the GPU-accelerated FM-LBM algorithm is compared to similar studies, it is underperforming. As can be seen in the figure on the right in figure 4.9, Entes reached 390 MLUPS simulating flow and temperature using a GPU-accelerated FM-LBM algorithm in a cavity measuring $200 \times 200 \times 200$ grid points [12]. This is significantly higher than the 112 MLUPS reached for the step 0.3 simulation, which was simulated on a smaller grid ($500^2 = 2.5 \times 10^5$ versus $200^3 = 8.0 \times 10^6$) and only simulated temperature and no flow. Tran et al. reports simulation speeds of 1200 MLUPS simulating only flow using the BGK collision operator [39, 40]. This performance is 38 times higher than the performance reached in step 0.1, which simulates flow using the more complicated FM-LBM collision operator.

Although the GPU-accelerated FM-LBM algorithm shows a substantial increase in computation speed when compared to the CPU FM-LBM algorithm, its computational performance under performs compared to the literature. This can be attributed to the fact that the algorithm was not fully optimized. In future research, parameters such as threads per block, memory coalescence, and the use of shared and register memory can be optimized to achieve greater computational speed improvements.

5

Validation of Steady-State Coupled Models

In this chapter, the single fields are gradually coupled to each other. For each step, the coupling is validated using the Tiberga benchmark case. Firstly, convective transport of the delayed precursors is activated in section 5.1. In section 5.2 the two-way temperature-neutronics coupling is introduced and validated. Section 5.3 studies purely the effects of buoyancy; the top-lid velocity is set to zero in this step. Lastly, the full-coupled problem, with a moving top-lid, is simulated in section 5.4 for several values for the top-lid velocity and reference power. In each step, the results of this research are also compared to the results of the two previous LBM multiphysics tools. Additionally, heatmaps of all observables for each step can be found in appendix B.

5.1. Step 1.1: Circulating Fuel

In the first coupling step of the Tiberga benchmark, the steady-state neutronics are assessed for a moving fluid. The velocity field is fixed on step 0.1 and the temperature field is uniformly fixed at $900\ \mathrm{K}$. The aim of this step is to assess if fuel motion has the correct effect on neutronics, in particular the reactivity loss due to fuel motion.

The observables in this step are the delayed neutron source $\sum_d \lambda_d C_d$ along the horizontal and vertical centerlines and the reactivity change compared to step 0.2: $\rho_{1.1} - \rho_{0.2}$. The simulation parameters are listed in table 5.1.

Table 5.1: Simulation parameters used for the simulation of step 1.1 of the Tiberga benchmark.

Parameter	Physical value	Physical unit	Lattice value	Lattice unit
Δt_{TH}	2.5×10^{-4}	s	1.0	lt
Δt_N	1.0×10^{-9}	s	1.0	lt
Sc	1200	$-$	1200	$-$
D_p	2.08×10^{-5}	$\mathrm{m^2\,s^{-1}}$	5.21×10^{-5}	$\mathrm{ls^2\,lt^{-1}}$

In this step, the same problem as in Step 0.3 was encountered: a very high Schmidt number is prescribed. By the same reasoning as in step 0.3, a very fine grid is required for a stable LBM simulation with such a high Schmidt number, resulting in impractical computation times. Therefore, the Schmidt number will be reduced to arrive at practical computation times. To understand the effect of lowering the Schmidt number, the delayed precursor advection-diffusion equation of equation 2.14 is non-dimensionalized.

$$\mathrm{ReSc}\left(\tilde{\boldsymbol{u}} \cdot \tilde{\boldsymbol{\nabla}} \tilde{C}_d\right) = \tilde{\boldsymbol{\nabla}}^2 \tilde{C}_d - \mathrm{Da}\, \tilde{C}_d \tag{5.1}$$

Here, the tilde notation is used for non-dimensionalized variables and a steady-state system is considered. Da represents the Damköhler number, which is the ratio of the decay rate of the precursors to the precursor diffusion constant. The Damköhler number increases with the Schmidt number.

In equation 5.1 it can be seen that the convective and decay terms start to dominate for high Schmidt numbers. However, since relatively small concentration gradients are encountered in step 1.1, a significantly lower Schmidt number than prescribed by the Tiberga benchmark is expected to be sufficient to simulate a convection-dominated system. To study this, similarly to step 0.3, several Schmidt numbers were simulated and their discrepancy with the benchmark results was computed. The results of this are shown in figure 5.1

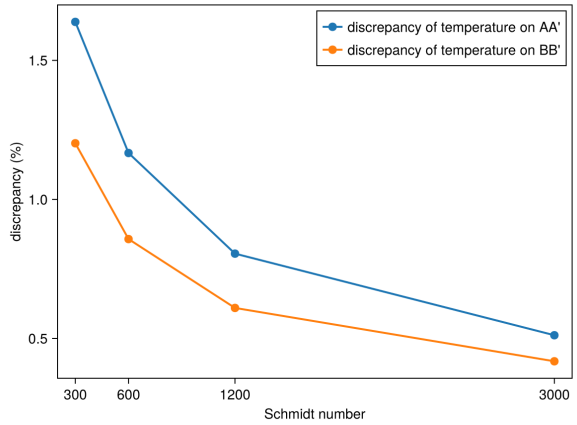


Figure 5.1: Discrepancy of the simulated delayed neutron source along the horizontal and vertical centerlines compared to the benchmark results for phase 1.1 of the Tiberga benchmark case. The x-axis shows the Schmidt numbers of the LBM simulation. The discrepancies were computed with the average of the benchmark results and is shown in percentages.

In figure 5.1 a clear downward trend of the discrepancy can be seen for both the horizontal and vertical centerline. The discrepancy is quite low, reaching around 0.5% for both centerlines at a Schmidt number of 3000. To study the spatial dependence of the discrepancy, the pointwise discrepancy was computed along the horizontal and vertical centerlines and is shown in figure 5.2. In this figure, it can be seen that the discrepancies are the highest at the edges of the domain. This is expected, since the largest concentration gradients are found here. The $1.5 - 2.0$ m region along the horizontal centerline shows the largest non-boundary discrepancy. As in step 0.3, this is also due to the high-density region in the top-right corner of the domain.

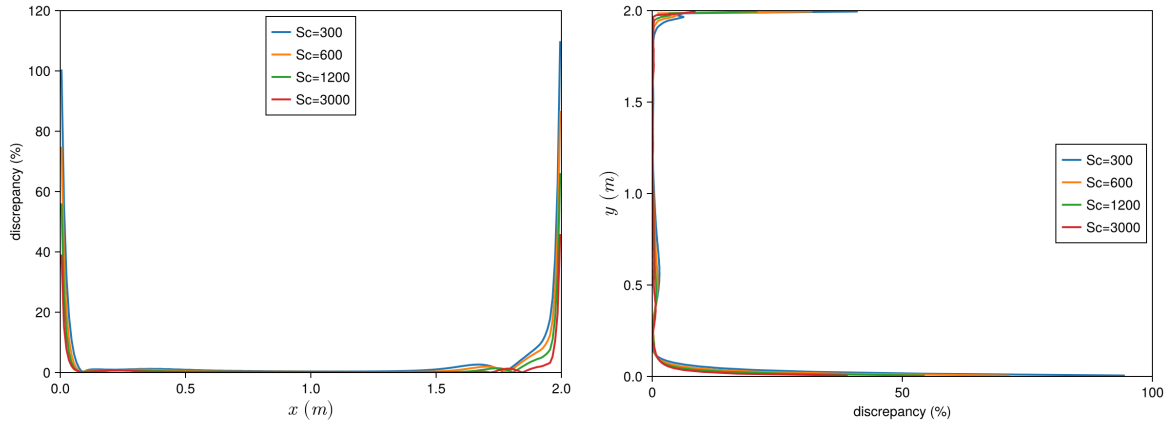


Figure 5.2: Pointwise discrepancy of delayed neutron source of simulation results along the horizontal (left) and vertical (right) centerlines compared to the benchmark results for phase 1.1 of the Tiberga benchmark case.

In figure 5.3 the delayed neutron source simulation results are plotted against the benchmark for a Schmidt number of 1200. The LBM simulation shows a good agreement with the benchmark results. The reactivity difference found in the simulation are listed in table 5.2 next to the benchmark values. The found LBM reactivity difference is slightly higher than the Tiberga benchmark, but is considered satisfactory. Compared to the results from Polderman and Entes the reactivity difference found in this research is even higher, suggesting that the two-domain approach and FM-LBM algorithm for the neutron diffusion equation simulate the neutronics in a different manner compared to the LBM studies by Polderman and Entes.

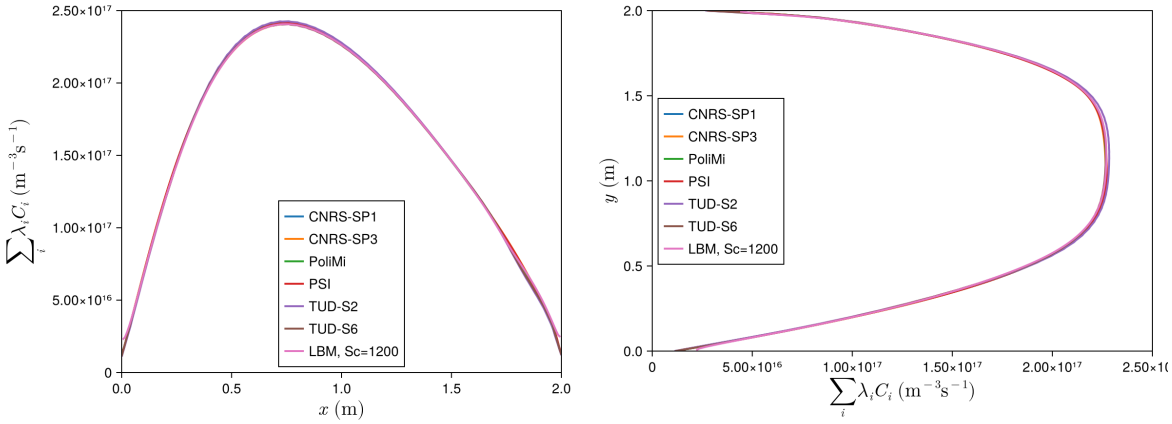


Figure 5.3: Simulation results of the delayed precursor source for step 1.1 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix algorithm on a 200×200 grid with a Schmidt number of 1200.

Table 5.2: The reactivity results for the LBM simulation of step 1.1 of the Tiberga benchmark shown alongside the benchmark results and previous LBM studies results. The effective multiplication factor k_{eff}, reactivity ρ (in per cent mille), and reactivity difference to step 0.2 are shown per column.

Code	k_{eff}	ρ (pcm)	$\rho_{1.1} - \rho_{0.2}$ (pcm)
CNRS-SP_1	1.00350021	348.8	−62.5
CNRS-SP_3	1.00291950	291.1	−62.6
PoliMi	1.00360495	359.2	−62.0
PSI	1.00349920	348.7	−63.0
TUD-S_2	1.00422377	420.6	−62.0
TUD-S_6	1.00520091	517.4	−60.7
LBM-Polderman	1.00457311	455.2	−65.3
LBM-Entes	1.00516126	513.5	−64.7
LBM-Pijls	1.00429923	428.1	−57.4

5.2. Step 1.2: Power Coupling

In step 1.2 of the Tiberga benchmark, the coupling between the neutronics and thermal-hydraulics is investigated for a fixed velocity field. In this step, power coupling is added. The power coupling is two-fold: the temperature field influences the nuclear cross sections due to salt expansion feedback, whereas the neutronics introduce a heat source for the temperature field due to fission. This step focuses on the power coupling, hence complex flow effects like buoyancy are not taken into account. The velocity field is also fixed at the converged field from step 0.1. The reactor power is normalized at 1 GW.

Given from the findings in earlier steps, it is assumed that the simulation will converge for a grid size of 200×200 and a Prandtl and Schmidt number of 1200. These values will be used in this and the next

steps and no further grid and Prandtl or Schmidt number convergence tests will be conducted. The simulation parameters used are the same as in earlier steps, which can be found in tables 4.5, 4.8, and 5.1.

The primary observable in this step is the reactivity change compared to step 1.1. In addition, the temperature field is compared to the benchmark along the vertical and horizontal centerlines, which is shown in figure 5.4, and the change in fission rate density with respect to step 0.2 is compared to the benchmark along the centerlines, which is shown in figure 5.5. The simulation results agree well with the benchmark results, with an average discrepancy for the temperature field of 0.18% along the horizontal and 0.29% along the vertical centerline, and an average discrepancy for the change of the fission rate of 1.3% along the horizontal and 1.4% along the vertical centerline.

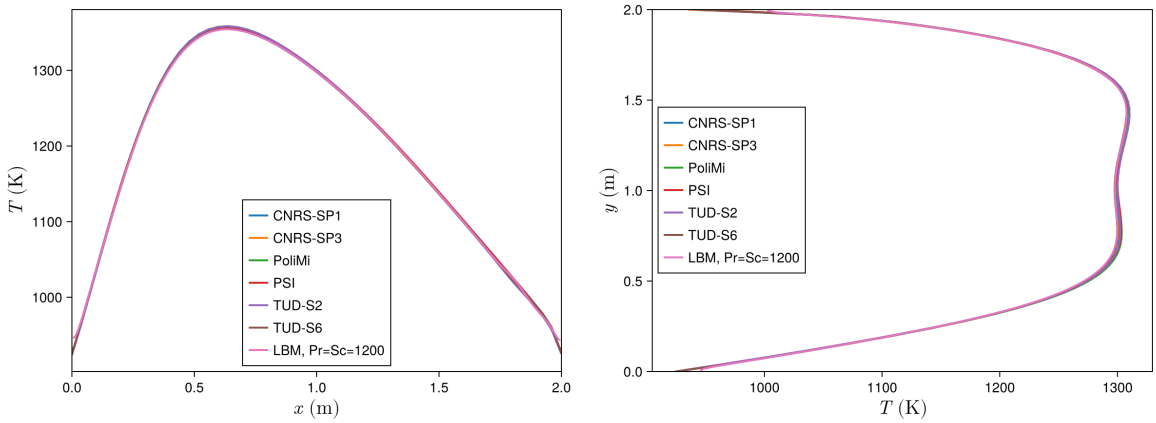


Figure 5.4: Simulation results of the temperature field for step 1.2 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix algorithm on a 200×200 grid with a Prandtl and Schmidt number of 1200.

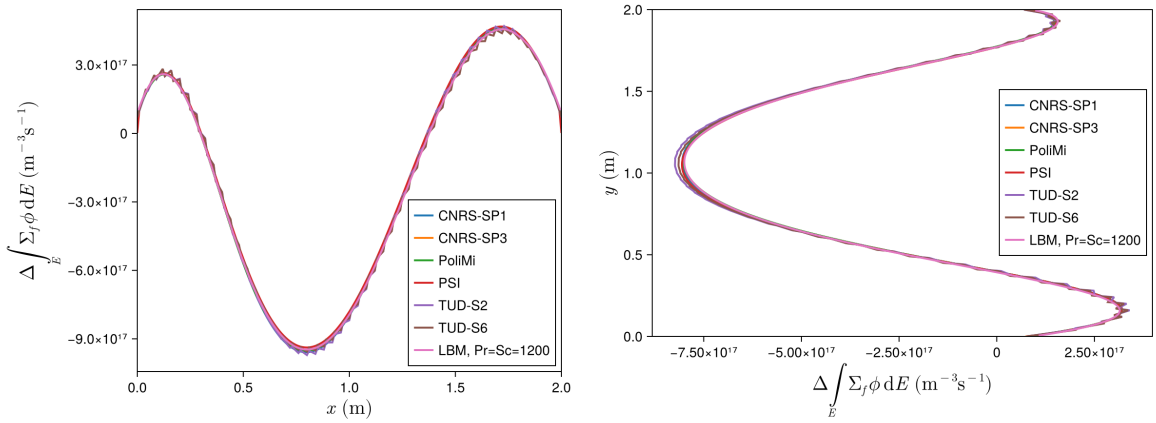


Figure 5.5: Simulation results of the fission rate density change with respect to step 0.2 for step 1.2 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix algorithm on a 200×200 grid and with a Prandtl and Schmidt number of 1200.

The reactivity difference to step 1.1 is shown in table 5.3. The reactivity difference found in the LBM simulation agrees well with those of the other institutions. The LBM reactivity difference is slightly higher than the other diffusion codes (PoliMi and PSI), while it is equal to the TUD-S_6 results. The found reactivity difference agrees well with Entes' results, while Polderman's reactivity difference is slightly lower than the benchmark results. Polderman's results can be attributed to the difference in neutronics modeling, especially the use of the BGK operator and approximation of the vacuum boundary condition.

Table 5.3: The reactivity results for the LBM simulation of step 1.2 of the Tiberga benchmark shown alongside the benchmark results and previous LBM studies results. The effective multiplication factor k_{eff}, reactivity ρ (in per cent mille), and reactivity difference to step 1.1 are shown per column.

Code	k_{eff}	ρ (pcm)	$\rho_{1.2} - \rho_{1.1}$ (pcm)
CNRS-SP_1	0.9920320	-803.2	-1152.0
CNRS-SP_3	0.9914576	-861.6	-1152.7
PoliMi	0.9920458	-801.8	-1161.0
PSI	0.9920035	-806.1	-1154.8
TUD-S_2	0.9928061	-724.6	-1145.2
TUD-S_6	0.9939903	-604.6	-1122.0
LBM-Polderman	0.99288174	-716.9	-1172.1
LBM-Entes	0.9940334	-600.2	-1113.7
LBM-Pijls	0.9931081	-694.0	-1122.0

5.3. Step 1.3: Buoyancy

In step 1.3 of the Tiberga benchmark case, the last coupling, buoyancy is added. The effects of buoyancy are studied under the simplest conditions, so the external momentum source is removed by setting the top lid velocity to zero. The buoyancy force is modeled by the Boussinesq approximation, which is driven by temperature gradients. The goal of this step is to predict the correct velocity field induced by the fission heat source and the correct reactivity change due to the movement of precursors. Discrepancies arising in the results of this step can be attributed to buoyancy effects, since the other couplings have been studied in steps 1.1 and 1.2. This step offers the first opportunity to study the complete multiphysics tool.

Again, the main observable of interest is the reactivity change compared to step 0.2. Along the horizontal and vertical centerlines, the velocity, temperature, and delayed neutron source are observables of interest. The simulation parameters used in this step are identical to those of earlier steps, which are listed in tables 4.4, 4.5, 4.8, and 5.1. The buoyancy specific parameters are shown in table 5.4

Table 5.4

Parameter	Physical value	Physical unit	Lattice value	Lattice unit
Δt_{TH}	2.5×10^{-4}	s	1.0	lt
Δt_N	1.0×10^{-9}	s	1.0	lt
β_{th}	2.0×10^{-4}	K^{-1}	1.80×10^{-1}	lT
g	9.81	$\mathrm{m\,s}^{-2}$	9.81×10^{-4}	$\mathrm{ls\,lt}^{-2}$

The horizontal and vertical velocity components along the centerlines are shown in figures 5.6 and 5.7. Both components agree well with the benchmark codes, showing an average discrepancy of 0.84% and 0.00% for the horizontal velocity component along the AA' and BB' line, and 0.48% and 0.40% for the vertical velocity component along the AA' and BB' line.

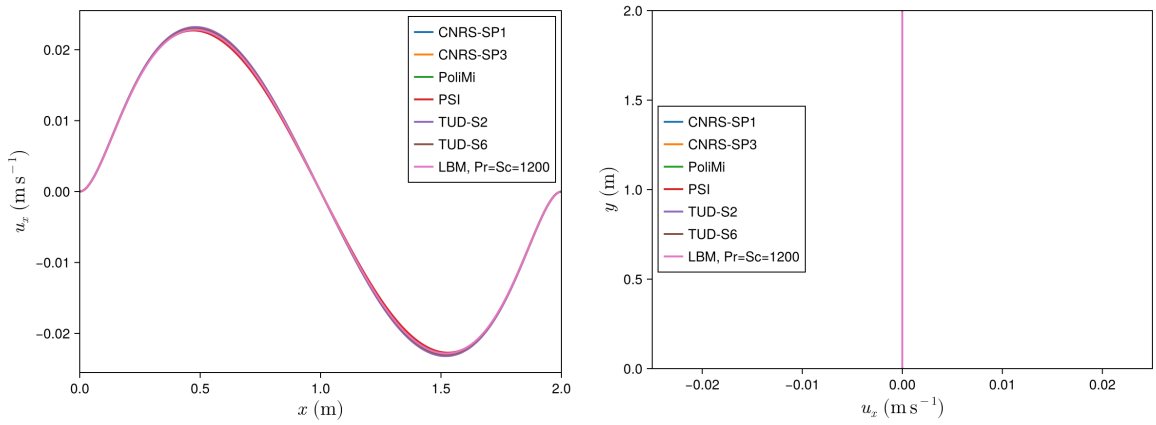


Figure 5.6: Simulation results of the horizontal velocity components for step 1.3 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix-algorithm on a 200×200 grid with a Prandtl and Schmidt number of 1200.

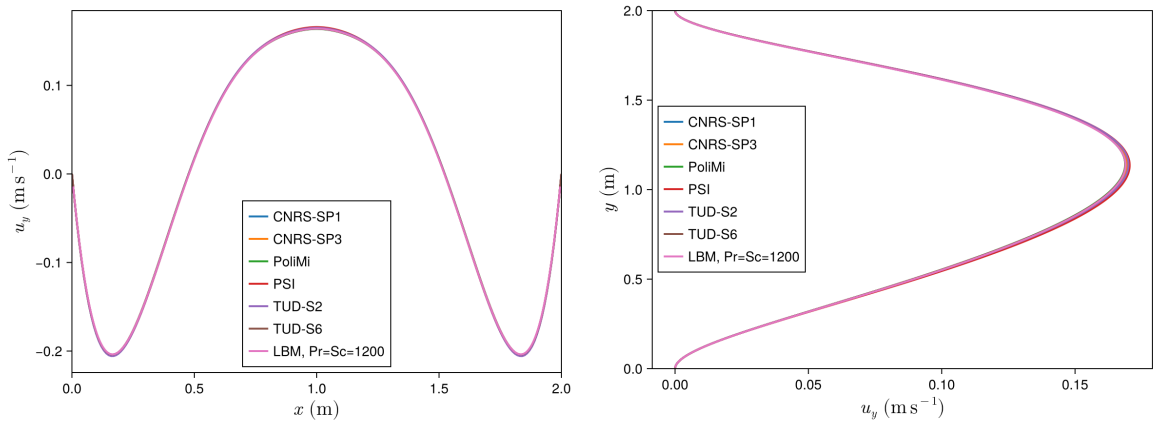


Figure 5.7: Simulation results of the vertical velocity components for step 1.3 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix-algorithm on a 200×200 grid with a Prandtl and Schmidt number of 1200.

The temperature field and delayed neutron source along the centerlines are shown in figure 5.8 and 5.9. In the figures, it can be seen that the temperature and delayed neutron source deviate more from the benchmark results. The LBM simulation temperature results are slightly underestimated compared to the benchmark results, in particular for regions above 1250 K. This suggests that the temperature-neutronics coupling is suppressed in some way by buoyancy. However, the average discrepancy of the temperature is small: 0.41% along the horizontal centerline and 0.41% along the vertical centerline, so these deviations are not considered to play a significant role in the overall reactor behavior. The delayed neutron source agrees better with the benchmark codes; however, it shows a larger average discrepancy: 0.84% along the horizontal centerline and 0.90% along the vertical centerline. This can be attributed to the larger variation of the delayed neutron source shapes among the benchmark codes compared to the temperature shapes. The delayed neutron source shape and magnitude of the LBM simulation fits well within the range of benchmark codes.

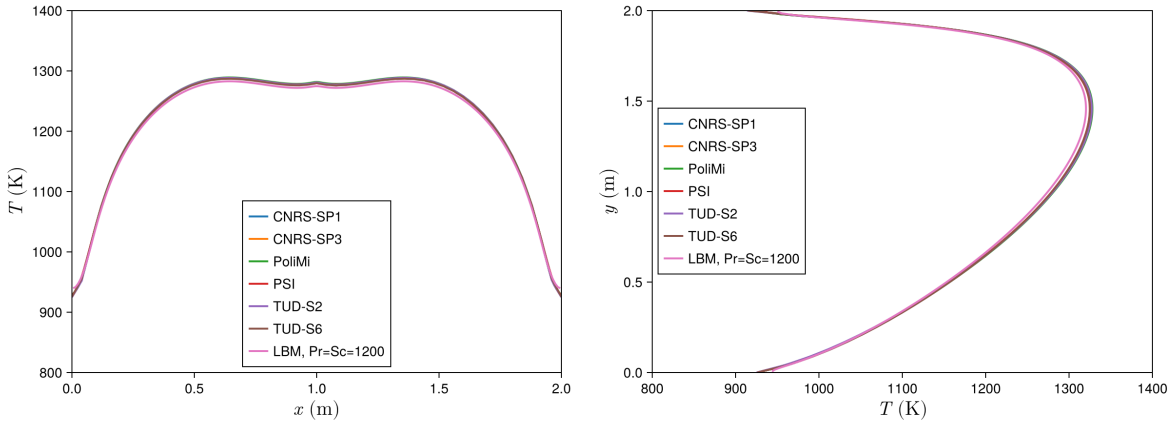


Figure 5.8: Simulation results of the temperature field for step 1.3 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix-algorithm on a 200×200 grid with a Prandtl and Schmidt number of 1200.

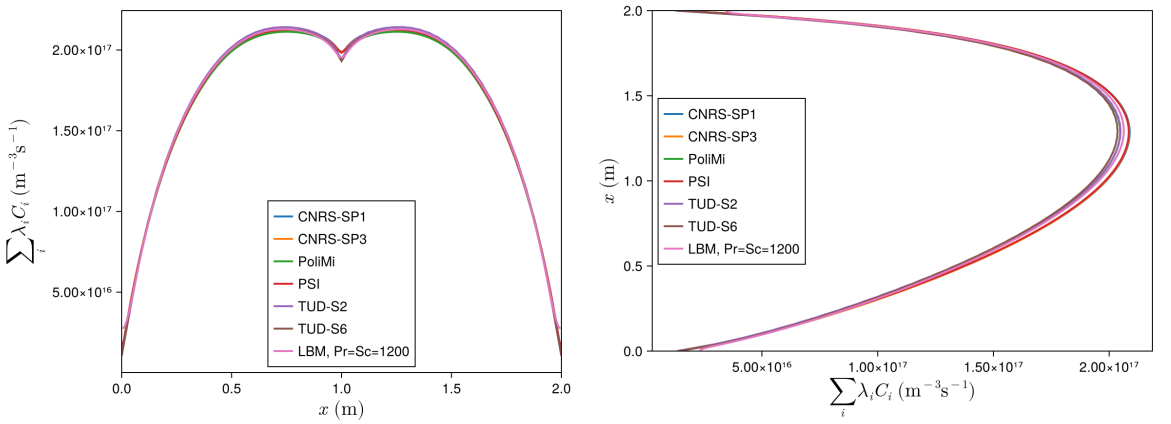


Figure 5.9: Simulation results of the delayed neutron source for step 1.3 of the Tiberga benchmark case. The LBM simulation results are shown alongside the benchmark results along the horizontal (left) and vertical (right) centerlines. The results were obtained using the filter-matrix-algorithm on a 200×200 grid with a Prandtl and Schmidt number of 1200.

The effective multiplication factor, the reactivity, and the reactivity difference to step 0.2 are shown in table 5.5 for the LBM simulation and the benchmark codes. The reactivity difference of the LBM simulation is slightly higher than the benchmark results, which was also observed in earlier steps. The found reactivity difference agrees well with Entes' result, which is also on the high end. Polderman again reports a reactivity difference on the lower end.

Table 5.5: The reactivity results for the LBM simulation of step 1.3 of the Tiberga benchmark shown alongside the benchmark resuls. The effective multiplication factor k_{eff}, reactivity ρ (in per cent mille), and reactivity change compared to step 0.2 are shown per column.

Code	k_{eff}	ρ (pcm)	$\rho_{1.3} - \rho_{0.2}$ (pcm)
CNRS-SP_1	0.9919730	−809.2	−1220.5
CNRS-SP_3	0.9914045	−867.0	−1220.7
PoliMi	0.99200648	−805.8	−1227.0
PSI	0.9919858	−807.9	−1219.6
TUD-S_2	0.9927933	−725.9	−1208.5
TUD-S_6	0.9939735	−606.3	−1184.4

Table 5.5: The reactivity results for the LBM simulation of step 1.3 of the Tiberga benchmark shown alongside the benchmark resuls. The effective multiplication factor k_{eff}, reactivity ρ (in per cent mille), and reactivity change compared to step 0.2 are shown per column.

Code	k_{eff}	ρ (pcm)	$\rho_{1.3} - \rho_{0.2}$ (pcm)
LBM-Polderman	0.99278435	-726.8	-1247.4
LBM-Entes	0.9940039	-603.2	-1181.4
LBM-Pijls	0.9930794	-696.9	-1182.3

5.4. Step 1.4: Full Coupling

In the final step of the steady-state coupling of the Tiberga benchmark case, all couplings and momentum sources are activated. This step is analogous to the previous; however, the top lid is no longer stationary. The observable in this step is the reactivity difference compared to step 0.2. The reactivity difference is computed for a range of top lid velocities and reference powers. Simulations were for a top lid velocity in the range of $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]$ $\mathrm{m\,s^{-1}}$ and a reference power in the range of $[0.2, 0.4, 0.6, 0.8, 1.0]$ GW. Initially, the simulations were performed with a Prandtl and Schmidt number of 1200, however these showed some instabilities and did not converge to a stable value. Therefore, the Prandtl and Schmidt number was lowered to 1000 for stable simulations. The results of the reactivity difference for a limited number of top lid velocities and reference powers are shown in table 5.6, next to the Tiberga benchmark results and other LBM code results.

Table 5.6: Reactivity difference of step 1.4 of the Tiberga benchmark case compared to step 0.2. The benchmark results are shown alongside the LBM simulation results. The reactivity differences are listed for top lid velocities of 0.1, 0.3, and 0.5 $\mathrm{m\,s^{-1}}$ and for reactor powers of 0.2, 0.6, and 1.0 GW.

Code	U_{lid} ($\mathrm{m\,s^{-1}}$)	$\rho_{1.4} - \rho_{0.2}$ (pcm) $P = 0.2$ GW	$P = 0.6$ GW	$P = 1.0$ GW
CNRS-SP_1		-268.5	-738.2	-1219.6
CNRS-SP_3		-268.8	-738.2	-1219.7
PoliMi		-269.0	-734.0	-1225.0
PSI		-270.2	-738.6	-1214.1
TUD-S_2	0.1	-265.8	-730.7	-1207.6
TUD-S_6		-260.1	-716.0	-1183.5
LBM-Polderman		–	–	–
LBM-Entes		–	–	–
LBM-Pijls		-260.7	-722.3	-1195.7
CNRS-SP_1		-269.5	-735.2	-1212.1
CNRS-SP_3		-269.8	-735.3	-1212.4
PoliMi		-278.0	-734.0	-1219.0
PSI		-274.0	-735.6	-1206.8
TUD-S_2	0.3	-269.5	-727.7	-1200.2
TUD-S_6		-263.8	-713.2	-1176.4
LBM-Polderman		–	–	–
LBM-Entes		–	-712.7	-1173.1
LBM-Pijls		-264.1	-719.2	-1188.6
CNRS-SP_1		-276.5	-732.9	-1204.8
CNRS-SP_3		-276.8	-733.0	-1205.2
PoliMi		-284.0	-737.0	-1214.0
PSI		-278.1	-733.1	-1199.8
TUD-S_2	0.5	-273.1	-725.2	-1193.0
TUD-S_6		-267.5	-710.8	-1169.7
LBM-Polderman		–	–	-1085.3
LBM-Entes		–	-708.9	-1164.8

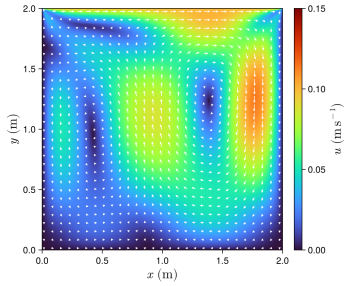
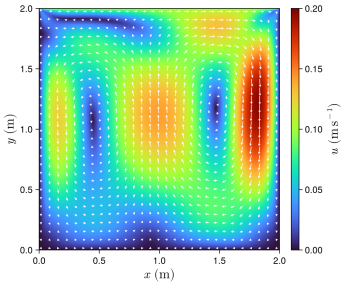
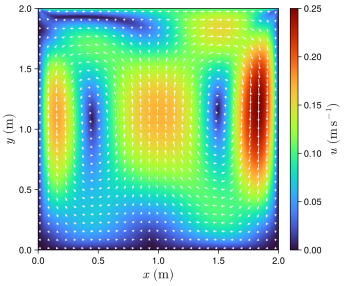
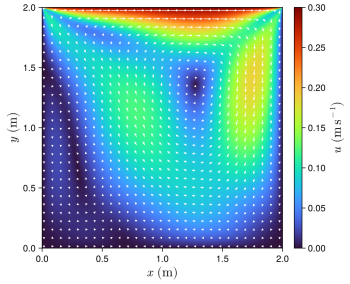
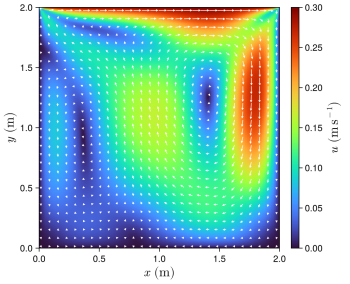
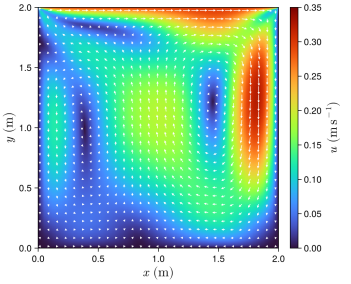
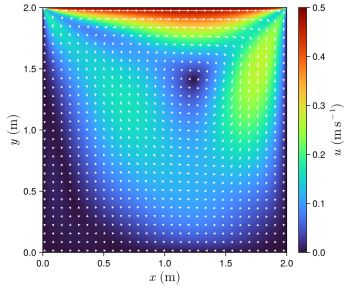
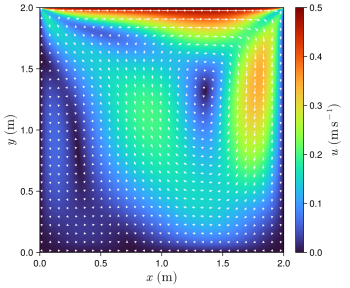
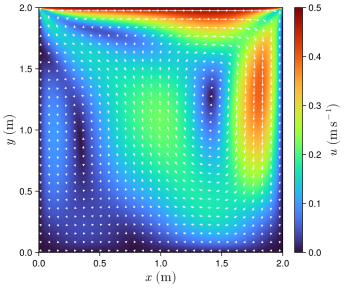
Table 5.6: Reactivity difference of step 1.4 of the Tiberga benchmark case compared to step 0.2. The benchmark results are shown alongside the LBM simulation results. The reactivity differences are listed for top lid velocities of 0.1, 0.3, and $0.5\ \mathrm{m\,s^{-1}}$ and for reactor powers of 0.2, 0.6, and $1.0\ \mathrm{GW}$.

Code	U_{lid} ($\mathrm{m\,s^{-1}}$)	$\rho_{1.4} - \rho_{0.2}$ (pcm)		
		$P = 0.2$ GW	$P = 0.6$ GW	$P = 1.0$ GW
LBM-Pijls		-268.5	-716.3	-1181.5

In table 5.6 it can be seen that the reactivity difference found in this research is slightly higher than five of the benchmark results (CNRS-SP_1, CNRS-SP_3, PoliMi, PSI, TUD-S_6) for all tested values of U_{lid} and P. The reactivity difference found for a reference power of $2\ \mathrm{GW}$ is in good agreement with the higher reactivity difference found in the TUD-S_6 code. However, for reference powers of 0.6 and $1.0\ \mathrm{GW}$, the reactivity difference is in between the five other benchmark codes and the TUD-S_6 results. Polderman and Entes only have results for a limited combination of top lid velocities and reference powers. Entes' reactivity differences are slightly higher than the TUD-S_6 results, which is unsurprising since both codes use the Phantom code with the same discretization for the neutronics. Polderman only reported results for a top lid velocity of $0.5\ \mathrm{m\,s^{-1}}$ and a reference power of $1.0\ \mathrm{GW}$. The reactivity difference is higher than the benchmark by about $120\ \mathrm{pcm}$, which can be explained by the different neutronics diffusion equation implemenation used. Overall, the reactivity differences in this research agree well with the benchmark and Entes' results, confirming that the multiphysics tool developed in this research can accurately model the steady-state multiphysics of a simplified molten salt fast reactor core.

To study the behaviour in the code in more detail, velocity heatmaps were created of each top lid velocity - reference power combination. These velocity heatmaps are shown in table 5.7, where arrows are added pointing in the direction of the flow and scaled by the velocity magnitude. Two competing fuel motions are expected: lid-driven cavity flow, induced by the moving top lid, on the one hand; and 2-cell buoyant flow, induced by the fission heat, on the other hand. In the case of a high lid velocity and low reference power (bottom-left cell in the table), it can be seen that the lid-driven cavity flow dominates. For the opposite case, of a low lid velocity and high reference power (top-right cell in the table), it can be seen that the 2-cell buoyant flow dominates, as expected. The other combinations show a mix of both effects, which is most pronounced in the high lid velocity, high reference power case (bottom-right cell in the table). Here, the combination of both effects creates three circulation cells: a strong buoyant cell on the right, a small buoyant cell on the bottom-left, and a lid-driven cavity cell on the top-left.

Table 5.7: Velocity heatmap with arrows for the nine simulated top lid velocity-reference power combinations of step 1.4 of the Tiberga benchmark case. The top row has a top lid velocity of $0.1\ \mathrm{m\,s^{-1}}$, the middle row $0.3\ \mathrm{m\,s^{-1}}$, and the bottom row $0.5\ \mathrm{m\,s^{-1}}$. Note that the colorbar range differs between figures.

$P = 0.2$ GW	$P = 0.6$ GW	$P = 1.0$ GW
		
		
		

6

Validation of Transient Multiphysics Coupled Models

In this final results chapter the results of the transient fully-coupled simulations will be discussed. The Tiberga benchmark included one transient case, where the power response of the model is tested for a sinusodial perturbation of the volumetric heat transfer coefficient for a range of frequencies. The specifics of this step are outlined in section 6.1 and the results in section 6.2. Additionally, to study the transient neutronics in the absence of flow and thermal gradients, a reactivity insertion was simulated for simplified geometries and compared to an analytical solution. This is discussed in section 6.3.

6.1. Description of phase 2.1 of the Tiberga benchmark

In the last step of the Tiberga benchmark, the transient behaviour of the multiphysics tool is studied. This step continues on the previous step, phase 1.4, considering the full coupled system with $U_{lid} = 0.5\,\mathrm{m\,s^{-1}}$ and $P_{ref} = 1\,\mathrm{GW}$. The transient response is studied in the most general manner by applying a perturbation in the frequency domain. Practically, this is done by perturbing the volumetric heat sink coefficient by a sine wave with an amplitude of 10% and a frequency f_{pert}. The new time-dependent volumetric heat sink coefficient is now given by

$$\gamma(t) = \gamma_0[1 + 0.1\sin(2\pi f_{pert} t)] \tag{6.1}$$

where γ_0 is the reference volumetric heat transfer coefficient used in the steady-state simulations. By varying the cooling of the salt, the reactor power will oscillate due to the negative salt expansion feedback coefficient. To study the transient response of the system, the phase shift and gain of the reactor power are computed for each frequency. The normalized gain is defined as

$$\text{Gain} = \frac{(P_{max} - P_{avg})/P_{avg}}{(\gamma_{max} - \gamma_{avg})/\gamma_{avg}} \tag{6.2}$$

Here, the subscripts max and avg denote maximum and average values, respectively. The gain is measured over the last period of the simulation, since at this point in time the perturbation has settled in a stable oscillation. Now we introduce the normalized heat transfer coefficient and power, shown in equations 6.3 and 6.4.

$$\tilde{\gamma} = \frac{\gamma(t) - \gamma_{avg}}{\gamma_{avg}} \tag{6.3}$$

$$\tilde{P} = \frac{P(t) - P_{avg}}{P_{avg}} \tag{6.4}$$

Here, γ_{avg} is equal to γ_0 and P_{avg} is equal to 1.0 GW. Since the heat transfer coefficient is perturbed by a sine with an amplitude of 10%, the denominator in equation 6.2, this equation can be simplified to:

$$\text{Gain} = 10\tilde{P}_{max} \tag{6.5}$$

The phase shift of the power is computed by comparing the time at which the maxima and minima occur between the heat transfer coefficient and the reactor power. This is expressed by

$$\text{Phase shift} = \pi \cdot f_{pert}\Big[(t_{\gamma_{max}} - t_{P_{max}}) + (t_{\gamma_{min}} - t_{P_{min}})\Big] \tag{6.6}$$

The phase-shift is determined over the last period of the simulation, since at this point the system has settled into a consistent oscillation.

6.2. Phase 2.1: Forced Convection Transient

The response of the system was simulated for frequencies of $[0.0125, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8]$ Hz. Snapshots of the power and heat transfer coefficient over the last two periods for each perturbation frequency can be found in appendix C. For each frequency, the gain and phase shift of the power with respect to the heat transfer coefficient was computed. Bode plots of the computed gain and phase shift are shown in figure 6.1, plotted with the results of the Tiberga benchmark and the results of Polderman's study. These results were generated using 10 medium timesteps per big timestep, and 100 small timesteps per medium timestep for the PCQSM algorithm.

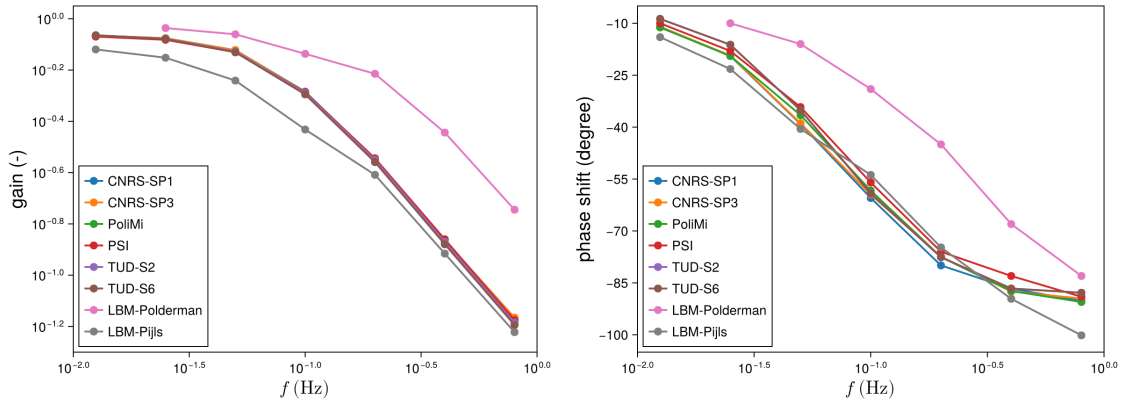


Figure 6.1: Response of the coupled multiphysics tool to a perturbation of the heat transfer coefficient for several frequencies compared to the Tiberga benchmark and Polderman's study. The power gain as defined in equation 6.2 is shown on the left, the phase-shift as defined in equation 6.6 is shown on the right.

In figure 6.1 it can be seen that the benchmark codes agree well with each other, especially for the gain plot where they completely overlap. The LBM algorithm developed by Polderman (LBM-Polderman) displays a delayed response. The gain and phase shift show the same trend as the benchmark codes, decreasing for increasing frequency; however, the response is shifted to the right in the frequency space. The LBM algorithm developed in this research (LBM-Pijls) agrees better with the benchmark codes, but still shows some discrepancies. To better understand these discrepancies, the difference and discrepancy of the gain and phase shift compared to the mean of the benchmark codes is plotted in figures 6.2 and 6.3. The discrepancy was computed using equation 4.2.

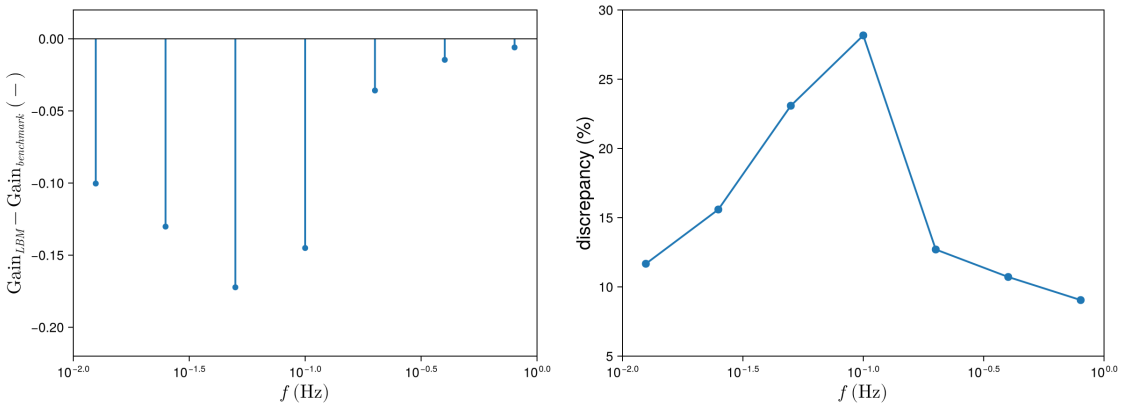


Figure 6.2: The difference (left figure) and discrepancy (right figure) of the gain between the LBM results and the mean of the benchmark codes plotted for each frequency.

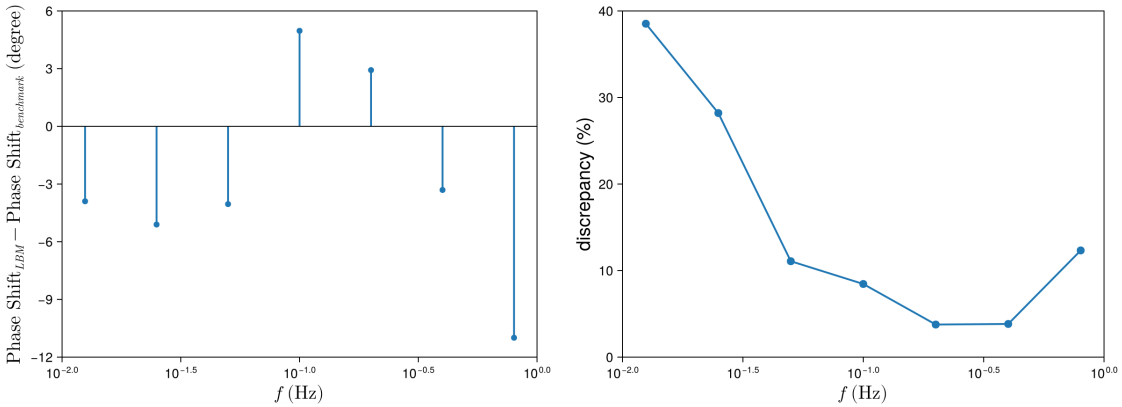


Figure 6.3: The difference (left figure) and discrepancy (right figure) of the phase shift between the LBM results and the mean of the benchmark codes plotted for each frequency.

In the above figures it can be seen that for the gain the largest differences are observed in the 0.0125 Hz to 0.1 Hz region, with a discrepancy peak of 28.2% at $f_{pert} = 0.1$ Hz. For frequencies of 0.2 Hz and higher, the difference and discrepancy are relatively constant, hovering around 0.02 and 10%, respectively. The phase shift shows more oscillatory behaviour, displaying a negative difference for the low frequencies (0.0125 Hz, 0.025 Hz, 0.05 Hz), a positive difference for the middle frequencies (0.1 Hz, 0.2 Hz), and again a negative difference for the high frequencies (0.4 Hz, 0.8 Hz). The discrepancy is the largest for the the extreme frequencies (0.125 Hz, 0.025 Hz, 0.8 Hz) and lowest for the middle-high frequencies (0.2 Hz, 0.4 Hz), with an average discrepancy of 3.8%.

Three potential sources of the discrepancy in the gain and phase shift can be identified: the thermal-hydraulics solver, the neutronics solver, or the coupling between them. To validate the transient neutronics solver, the response of a reactivity insertion in the absence of flow and thermal effects is studied in the next section.

6.3. Phase 2.2: Reactivity Insertion

To study the validity of the transient neutronics FM-LBM solver, a situation is considered where a reactivity insertion is made in a system with no flow and constant temperature T_0. For one neutron energy group and n precursor families, this system can be simulated using the following point-kinetics equations for neutron density $n(t)$ and precursor density C_d:

$$n(t+\Delta t) = \left(\frac{\rho - \beta_{tot}}{\Lambda} n(t) + \sum_d \lambda_d C_d \right) \Delta t + n(t) \tag{6.7}$$

$$C_d(t+\Delta t) = \left(\frac{\beta_d}{\Lambda} n(t) - \lambda_d C_d \right) \Delta t + C_d(t) \tag{6.8}$$

Here, Δt denotes the simulation timestep, and ρ, β_{tot}, β_d, and Λ are the point-kinetics parameters for the reactivity, total delayed neutron fraction, delayed neutron fraction, and neutron generation time, respectively. λ_d is the decay constant of precursor family d.

Two cases were simulated for the transient FM-LBM model, the first case concerned a 1-dimensional slab with boundary conditions on both sides, while the second case concerned a square domain periodic over all axes. Nuclear parameters such as decay constant, delayed neutron fraction, and all cross sections were equal among the three models. For both cases and the analytical point-kinetics model a reactivity of $0.1\beta_{tot}$ was inserted at $t = 0$. The time step was taken as 1.0×10^{-4} s for all models; shorter time steps did not show any difference. The spatial convergence for the 1-dimensional slab was studied and the results are shown in figure 6.4. The discrepancy was computed using equation 4.2 with the analytical model as reference. From the figure it can be concluded that at a grid size of $N = 200$ the FM-LBM algorithm has spatially converged, since the graph flatten with increasing grid size and the absolute discrepancy only changes little (63.99% for $N = 25$ compared to 63.82% for $N = 200$).

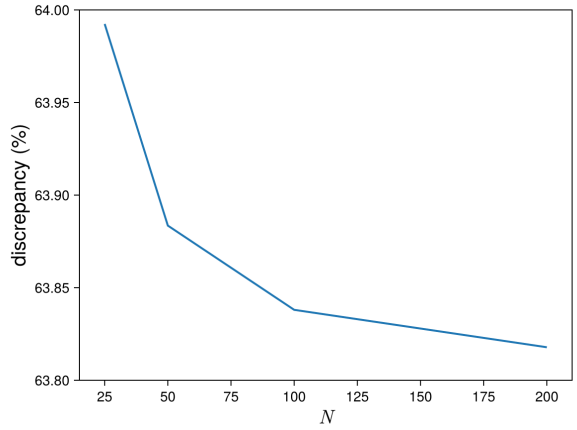


Figure 6.4: Discrepancy of the 1D-slab FM-LBM algorithm with the analytical model for different grid sizes N. The discrepancy was computed using equation 4.2.

The results of the simulations for the 1-dimensional slab and the periodic square are shown in figure 6.5. To iterate, only the neutrons and precursors are simulated in the FM-LBM model, flow and heat effects are removed by imposing a stagnant fuel and constant temperature.

As can be seen in figure 6.5, the 1-dimensional slab and the periodic square follow the same trend as the analytical model but over predict the neutron density. To better understand this over prediction, the ratio of the FM-LBM neutron density over the analytical neutron density is plotted in figure 6.6. In this figure it can be seen that with increasing time the over prediction increases for both the 1-dimensional slab and the periodic square. These results show a small over prediction of the prompt neutron density ($10^{-4} - 10^{-1}$ s region) and a large over prediction of the delayed neutron density ($10^{1} - 10^{2}$ s region).

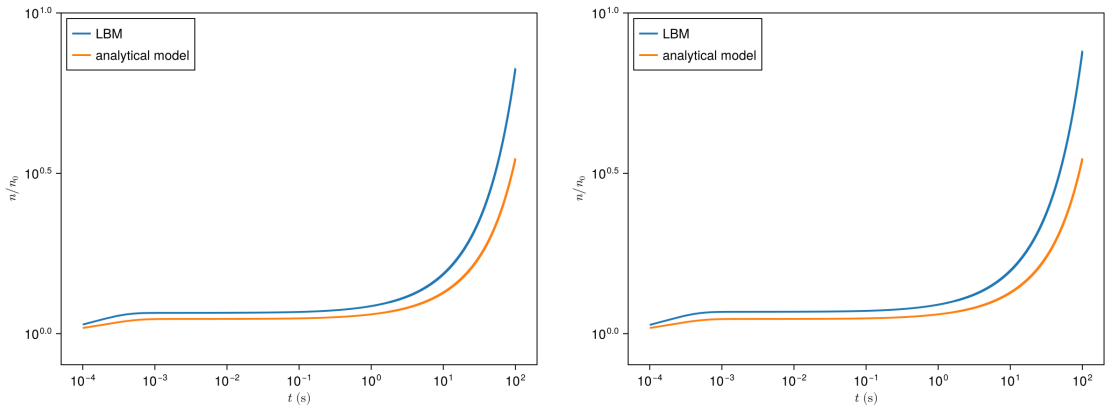


Figure 6.5: Neutron density response for a 1-dimensional slab (left figure) and a periodic square (right figure) in blue for a reactivity insertion of $\rho = 0.1\beta_{tot}$ at $t = 0$. The analytical model is shown in orange. The neutron density is normalized to the initial neutron density n_0 at $t = 0$.

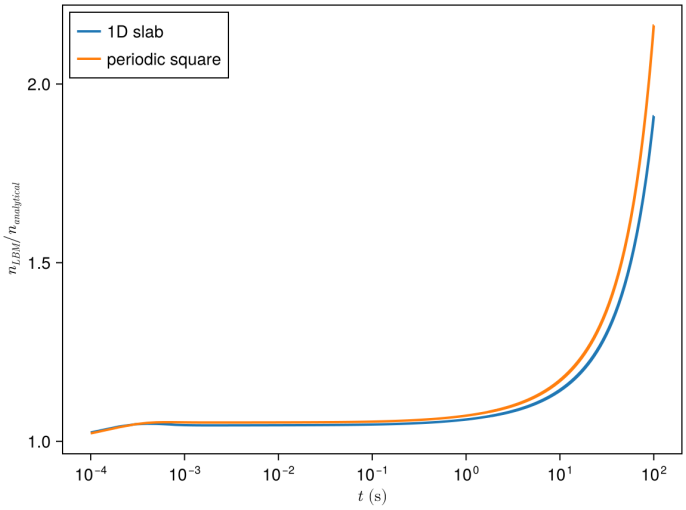


Figure 6.6: The relative neutron density for the 1D slab and the periodic square compared to the analytical neutron density.

Figure 6.6 also helps us explain the power gain of step 2.1 shown in figure 6.1. In figure 6.6 it can be seen that up to times of 5 s the over prediction of the neutron density is a factor of about 1.1, while above these times it increases to a factor of 2. This coincides with the gain discrepancy jump observed in figure 6.1, where a large jump is observed between the 0.1 Hz and 0.2 Hz frequencies, which have periods of 10 s and 5 s, which falls exactly in the transition regime of the relative neutron density.

7

Conclusion and Recommendations

In this research, a simulation tool was developed to model the coupled thermal-hydraulics, neutronics, and precursor transport in a MSFR reactor core. The simulation tool made use the GPU-accelerated filter-matrix-lattice Boltzmann method (FM-LBM) algorithm. The conclusions of this research will be discussed in this chapter, starting with the conclusions of the steady-state FM-LBM model presented in section 7.1. Section 7.2 discusses the results from the transient simulations. The computational performance will be discussed in section 7.3. Finally, recommendations for future research will be given in section 7.4.

7.1. Steady-state FM-LBM Model Development

In the first part of this research, a steady-state solver for the coupled thermal-hydraulics, precursor transport, and neutronics was developed for MSFR cores using a GPU-accelerated FM-LBM algorithm. This algorithm employed a double distribution function approach, where the 18 physical fields (momentum, enthalpy, 6 neutron groups, and 8 precursor families) each had their own distribution function. The LBM algorithm operates by alternating between propagation and collision of the distribution functions over a two-dimensional lattice grid. Interactions between the fields occurred in the collision step, through convective and source terms. The collision was performed using the FM-LBM algorithm, which uses a filter-matrix to transform the non-physical distribution function to a physical solution vector, which is modified and then transformed back to a post-collision distribution function which was propagated. The FM-LBM algorithm filters out non-physical terms, thereby increasing the stability of the simulation.

The timescales of the neutronics are some orders of magnitude smaller than those of the thermal-hydraulics and precursor transport; therefore, a two-domain approach was used where the neutronics were simulated in a seperate domain using a shorter timestep. Since the neutronics domain operates on a much shorter timescale, for each thermal-hydraulics timestep the neutronics were simulated to convergence. The power-method was used to iteratively solve the steady-state neutronics. A novel FM-LBM solution vector was derived for the neutron diffusion equation, which was analogous to the solution vector of the enthalpy and precursor transport. Additionally, a novel boundary condition treatment for the vacuum boundary condition of the neutron diffusion equation was developed, which interpolates the scalar neutron flux to the wall.

The steady-state FM-LBM model was validated using the steady-state single physics (Phase 0) and steady-state coupled physics (Phase 1) steps of the Tiberga benchmark, the results of which can be found in chapters 4 and 5. In the Tiberga benchmark, instead of using maximum values, observables measured along the horizontal and vertical centerlines and reactivity differences were used to compare results. The accuracy of the results was quantized by computing the discrepancy with the average of the benchmark codes. For all but one observable, the discrepancy of the FM-LBM model was below 1.0%. Only the fission rate density change of step 1.2 reported discrepancies greater than 1.0%. This can explained by the fact that the benchmark displayed a large discrepancy among the codes here.

Two significant findings were identified from the steady-state results of the FM-LBM model. Firstly,

the very high Prandtl and Schmidt numbers of the Tiberga benchmark, which effectively remove diffusion effects in transport, proved impossible to result in stable and computational efficient simulations. Therefore, the Prandtl and Schmidt numbers were significantly reduced to 1200, resulting in stable and computational efficient simulations that agreed well with the benchmark. The largest discrepancies with the benchmark were observed along the walls, which was unsurprising since the largest gradients, and thus diffusion effects, can be found there. Additionally, further increasing the Prandtl and Schmidt numbers does not result in a significant improvement of simulation results.

Secondly, the reactivity differences reported in each step were consistently slightly higher than the benchmark codes. The higher reactivity was already observed in the uncoupled neutronics step, suggesting that the higher reactivity is caused by the implementation of the neutronics and not due to the thermal-hydraulics or coupling method. A possible error source of the neutronics is in the source implementation of the filter-matrix: for both the neutronics and precursor source, the scalar flux and precursor concentration appear in the source term, while the source term also updates the scalar flux and precursor concentration. It is assumed that the time steps are chosen sufficiently small for these errors to be negligible, however it may be the case that at some times and locations in the domain these errors are significant. It may also be the case that the neutron diffusion equation is not sufficient to model the neutronics in LBM, reinforced by Polderman who also had difficulties with modeling the neutronics. Therefore, it can be explored how the neutron transport equation can be implemented in the FM-LBM framework.

7.2. Transient FM-LBM Model Development

To simulate the transient behaviour of a MSFR core, the steady-state FM-LBM algorithm was modified to capture the quick reaction times of the prompt neutrons. This was done using the predictor-corrector quasi-static method (PCQSM), where the neutron scalar flux is factorized in a slowly changing, spatially dependent shape function and a fast changing, non-spatially dependent amplitude function. The shape function is solved over big time steps using the steady-state LBM neutronics solver, while the amplitude function is solved over small time steps using the point-kinetics equations. The point-kinetics parameters are computed from the shape function, which is interpolated over medium time steps.

The Tiberga benchmark offers one transient case, in which the power response of the FM-LBM model was measured for a perturbation of the volumetric heat transfer coefficient. The gain and phase shift of the power response was computed with respect to the heat transfer coefficient for a range of frequencies and these were compared to the benchmark codes in a Bode plot. While the FM-LBM code showed the same trends for both gain and phase shift as the benchmark codes, the absolute values did not match. For low frequencies the FM-LBM model reported a significant lower gain and the phase shift was for some frequencies above the benchmark, while for other frequencies it was below the benchmark.

Additionally, a reactivity insertion case was simulated in the absence of flow and thermal feedback effects, to study how the FM-LBM model performed simulating only neutronics and precursors. This case showed that the transient FM-LBM model predicted the neutron density response for all geometries tested. The prompt neutron density was slightly over predicted, while the delayed neutron density increase was too early. This confirms that the cause of the discrepancy power response is due to the transient neutronics and precursors solver.

Two possible causes for the inaccuracy of the transient solver can be identified. Firstly, the predictor-corrector quasi-steady method may not be implemented correctly or be suited to this solver, however this is deemed unlikely. Secondly, it could be that the neutronics model used, either in the FM-LBM model or the use of the neutron diffusion equation - is not well suited to simulate the full neutronics of a MSFR core - something that was also observed in the steady-state cases.

7.3. Computational Performance

The FM-LBM algorithm was accelerated using the Julia-CUDA framework. In the acceleration, optimizations such as multi-dimensional array flattening and the use of shared and register memory were used. The GPU-acceleration was measured for the single physics benchmark cases, to study the speed up for each physical field. Step 0.3simulating the temperature reported the highest computational performance with a maximum of 112 million lattice updates per second (MLUPS), step 0.1 simulating the

lid-driven cavity flow the second highest with 32 MLUPS, and step 0.2 simulating the neutronics the lowest with 13 MLUPS. This was expected, since the temperature requires the least memory usage and is most simple to simulate, while the neutronics demand the most memory usage.

Compared to similar GPU-accelerated LBM algorithms, the GPU-accelerated FM-LBM algorithm underperformed by a factor of 10 to 100 times. Although the FM-LBM algorithm made use of multidimensional array flattening and the use of shared and register memory to speed up memory access, this could be optimized further. Furthermore the threads per block can be optimized, which was not done in this research.

7.4. Recommendations for Future Research

For future research, the following recommendations are suggested to improve the computational performance and accuracy of the model:

- The Tiberga benchmark does not consider all physical processes in a MSFR reactor core. Doppler shifts, which depend on the reactor temperature, and decay heat generated by precursors are omitted. Inclusion of these effects would increase the realism of the FM-LBM model.
- The molten salt has a high melting point, making the mixture prone to freezing in low temperature areas. This is especially a concern in the heat exchangers of a MSFR reactor core, where heat is actively extracted from the molten salt. To extend the FM-LBM model to full MSFR reactor core simulations, the freezing and melting of salt should be incorporated into the model.
- Turbulence was explicitly omitted in the Tiberga benchmark case due to its computational complexity. However, to simulate realistic MSFR reactor cores the modelling of turbulence is necessary. To reduce computational demands, local grid refinements can be used to achieve a greater resolution near the walls and other areas with large gradients.
- Realistic MSFR reactor core designs are 3-dimensional and contain complex shapes and boundaries. The HBB and ABB boundary techniques used are designed for Cartesian boundaries; to study complex, non-Cartesian boundaries new boundary techniques can be developed or tested to simulate complex geometries with the most accuracy and smallest computational burden. The possibility of the use of non-grid-aligned periodic boundaries is also interesting to research.
- In this research, the neutron diffusion equation was used and a FM-LBM algorithm was created. However, in the future it can be researched how the neutron transport equation can be implemented using the FM-LBM algorithm and if this leads to accuracy improvements. Wang et al. showed that the SP_3 discretization can be modeled in the LBM using two distribution functions for each neutron energy group [48]. Implementing a GPU-accelerated, FM-LBM version of this approach might improve the accuracy of the neutronics and allow for the correct simulation of transient problems.
- Lastly, while the GPU-acceleration of this FM-LBM algorithm improved the computational performance compared to the CPU FM-LBM algorithm, the computational performance is 10 to 100 times below that of similar GPU-accelerated LBM algorithms. In future research, more optimizations and benchmarking of the code can be performed to improve the computational performance of the algorithm, allowing for the simulation of more complex cases and quicker validation and development.

References

[1] Koning, A. J. et al. “The JEFF evaluated nuclear data project”. In: *International Conference on Nuclear Data for Science and Technology* (2007), pp. 721–726. DOI: 10.1051/ndata:07476. URL: https://doi.org/10.1051/ndata:07476.

[2] M. Allibert et al. “Molten salt fast reactors”. In: *Handbook of Generation IV Nuclear Reactors*. Elsevier, 2016, pp. 157–188. ISBN: 978-0-08-100149-3. DOI: 10.1016/B978-0-08-100149-3.00007-0. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780081001493000070 (visited on 05/13/2025).

[3] World Nuclear Assocation. “Carbon Dioxide Emissions From Electricity”. In: (2024). Accessed on 2025-04-13. URL: https://world-nuclear.org/information-library/energy-and-the-environment/carbon-dioxide-emissions-from-electricity.

[4] Antonio Cammi et al. “A multi-physics modelling approach to the dynamics of Molten Salt Reactors”. In: *Annals of Nuclear Energy* 38.6 (2011), pp. 1356–1372.

[5] Zhenhua Chai and Baochang Shi. “Multiple-relaxation-time lattice Boltzmann method for the Navier-Stokes and nonlinear convection-diffusion equations: Modeling, analysis, and elements”. In: *Physical Review E* 102.2 (Aug. 17, 2020), p. 023306. ISSN: 2470-0045, 2470-0053. DOI: 10.1103/PhysRevE.102.023306. URL: https://link.aps.org/doi/10.1103/PhysRevE.102.023306 (visited on 03/03/2025).

[6] Dipankar Chatterjee. “An enthalpy-based thermal lattice Boltzmann model for non-isothermal systems”. In: *Europhysics Letters* 86.1 (2009), p. 14004.

[7] 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.

[8] Thomas Dietz, Rachael L Shwom, and Cameron T Whitley. “Climate change and society”. In: *Annual Review of Sociology* 46.1 (2020), pp. 135–158.

[9] Thomas James Dolan. *Molten salt reactors and thorium energy*. Woodhead Publishing, 2017.

[10] James J. Duderstadt and Louis J. Hamilton. *Nuclear reactor analysis*. Wiley, 1976. URL: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/89079/1976_Nuclear_Reactor_Analysis.pdf (visited on 07/05/2024).

[11] Sandra Dulla, Ernest H. Mund, and Piero Ravetto. “The quasi-static method revisited”. In: *Progress in Nuclear Energy* 50.8 (Nov. 2008), pp. 908–920. ISSN: 01491970. DOI: 10.1016/j.pnucene.2008.04.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S014919700800084X (visited on 07/16/2024).

[12] Tom Entes. “Development and Integration of a GPU-Accelerated Lattice Boltzmann Simulation Tool for Thermal Hydraulics with Neutronics for Multiphysics Simulations in Molten Salt Fast Reactor Cores”. PhD thesis.

[13] Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational Methods for Fluid Dynamics*. Google-Books-ID: i9CpDwAAQBAJ. Springer, Aug. 16, 2019. 606 pp. ISBN: 978-3-319-99693-6.

[14] Carlo Fiorina. “GeN-Foam: a novel OpenFOAM® based multi-physics solver for 2D/3D transient analysis of nuclear reactors”. In: *Nuclear Engineering and Design* (2015).

[15] Generation IV Internation Forum. “Generation IV Goals, Technologies and GIF R&D Roadmap”. In: (2001). Accessed on 2025-04-13. URL: https://www.gen-4.org/generation-iv-criteria-and-technologies.

[16] Generation IV International Forum. “Molten Salt Reactors (MSR)”. In: (2001). Accessed on 2025-05-13. URL: https://www.gen-4.org/generation-iv-criteria-and-technologies/molten-salt-reactors-msr.

[17] D Gérardin et al. “Design Evolutions of the Molten Salt Fast Reactor”. In: ().

[18] Irina Ginzburg and Frederik Verhaeghe. “Two-Relaxation-Time Lattice Boltzmann Scheme: About Parametrization, Velocity, Pressure and Mixed Boundary Conditions”. In: *Commun. Comput. Phys.* (2008).

[19] Johannes Habich et al. “Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on nVIDIA GPUs using CUDA”. In: *Advances in Engineering Software* 42.5 (2011), pp. 266–272.

[20] Robert Hargraves and Ralph Moir. “Liquid Fluoride Thorium Reactors: An old idea in nuclear power gets reexamined”. In: *American Scientist* 98.4 (2010), pp. 304–313. ISSN: 00030996. URL: http://www.jstor.org/stable/27859537 (visited on 05/15/2025).

[21] Xiaoyi He, Shiyi Chen, and Gary D Doolen. “A novel thermal model for the lattice Boltzmann method in incompressible limit”. In: *Journal of computational physics* 146.1 (1998), pp. 282–300.

[22] Tianliang Hu et al. “Finite volume method based neutronics solvers for steady and transient-state analysis of nuclear reactors”. In: *Energy Procedia* 127 (2017), pp. 275–283.

[23] Timm Krüger et al. *The Lattice Boltzmann Method: Principles and Practice*. Graduate Texts in Physics. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-44647-9 978-3-319-44649-3. DOI: 10.1007/978-3-319-44649-3. URL: http://link.springer.com/10.1007/978-3-319-44649-3 (visited on 06/28/2024).

[24] Julia Language. “Julia Language Documentation”. In: (2025). Accessed on 2025-05-19. URL: https://docs.julialang.org/en/v1/.

[25] David LeBlanc. “Molten salt reactors: A new beginning for an old idea”. en. In: *Nuclear Engineering and Design* 240.6 (June 2010), pp. 1644–1656. ISSN: 00295493. DOI: 10.1016/j.nucengdes.2009.12.033. URL: https://linkinghub.elsevier.com/retrieve/pii/S0029549310000191 (visited on 05/13/2025).

[26] Jaakko Leppänen et al. “The Serpent Monte Carlo code: Status, development and applications in 2013”. In: *Annals of Nuclear Energy* 82 (Aug. 2015), pp. 142–150. ISSN: 03064549. DOI: 10.1016/j.anucene.2014.08.024. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454914004095 (visited on 04/08/2025).

[27] NVIDIA. “CUDA Toolkit Documentation”. In: (2025). Accessed on 2025-05-16. URL: https://docs.nvidia.com/cuda/.

[28] Coco Polderman. “Developing a multi-physics tool for the neutronics-thermal-hydraulics in a molten salt reactor using the lattice Boltzmann method”. PhD thesis. Delft University of Technology, 2024. 101 pp.

[29] J. N. Reddy. *An Introduction to Continuum Mechanics*. 2nd ed. Cambridge University Press, 2013.

[30] H. Rouch et al. “Preliminary thermal–hydraulic core design of the Molten Salt Fast Reactor (MSFR)”. In: *Annals of Nuclear Energy* 64 (Feb. 2014), pp. 449–456. ISSN: 03064549. DOI: 10.1016/j.anucene.2013.09.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454913004829 (visited on 05/13/2025).

[31] Jérôme Serp et al. “The molten salt reactor (MSR) in generation IV: Overview and perspectives”. In: *Progress in Nuclear Energy* 77 (Nov. 2014), pp. 308–319. ISSN: 01491970. DOI: 10.1016/j.pnucene.2014.02.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S0149197014000456 (visited on 05/13/2025).

[32] J A Somers. “Direct simulation of fluid flow with cellular automata and the lattice-Boltzmann equation”. In: ().

[33] Edward A Spiegel and G Veronis. “On the Boussinesq approximation for a compressible fluid.” In: *Astrophysical Journal, vol. 131, p. 442* 131 (1960), p. 442.

[34] GISTEMP Team. “GISS Surface Temperature Analysis (GISTEMP), version 4”. In: *J. Geophys. Res. Atmos.* 124.12 (2024). Dataset accessed 2025-04-13, pp. 6307–6326. DOI: 10.1029/2018JD. URL: data.giss.nasa.gov/gistemp/.

[35] M. Tiberga. “Developemnt of a high-fidelity multi-physics simulation tool for liquid-fuel fast nuclear reactors”. PhD thesis. Delft University of Technology, 2020.

[36] Marco Tiberga et al. “Preliminary investigation on the melting behavior of a freeze-valve for the Molten Salt Fast Reactor”. In: *Annals of Nuclear Energy* 132 (Oct. 2019), pp. 544–554. ISSN: 03064549. DOI: 10.1016/j.anucene.2019.06.039. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454919303573 (visited on 05/13/2025).

[37] Marco Tiberga et al. “Results from a multi-physics numerical benchmark for codes dedicated to molten salt fast reactors”. In: *Annals of Nuclear Energy* 142 (July 1, 2020), p. 107428. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2020.107428. URL: https://www.sciencedirect.com/science/article/pii/S0306454920301262 (visited on 06/28/2024).

[38] 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.

[39] Nhat-Phuong Tran, Myungho Lee, and Dong Hoon Choi. “Memory-efficient parallelization of 3D lattice Boltzmann flow solver on a GPU”. In: *2015 IEEE 22nd International Conference on High Performance Computing (HiPC)*. IEEE. 2015, pp. 315–324.

[40] Nhat-Phuong Tran, Myungho Lee, and Sugwon Hong. “Performance optimization of 3D lattice Boltzmann flow solver on a GPU”. In: *Scientific Programming* 2017.1 (2017), p. 1205892.

[41] Shisheng Wang et al. “A passive decay heat removal system for emergency draining tanks of molten salt reactors”. In: *Nuclear Engineering and Design* 341 (Jan. 2019), pp. 423–431. ISSN: 00295493. DOI: 10.1016/j.nucengdes.2018.11.021. URL: https://linkinghub.elsevier.com/retrieve/pii/S0029549318309567 (visited on 05/13/2025).

[42] Yahui Wang and Yu Ma. “lbmNTH: A unified lattice Boltzmann framework for coupled neutronics-thermal-hydraulics analysis”. In: *Annals of Nuclear Energy* 166 (Feb. 1, 2022), p. 108750. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2021.108750. URL: https://www.sciencedirect.com/science/article/pii/S0306454921006265 (visited on 06/28/2024).

[43] Yahui Wang and Yu Ma. “Unstructured finite-volume lattice Boltzmann method for the multi-group SP3 simulation”. In: *Annals of Nuclear Energy* 170 (June 2022), p. 109012. ISSN: 03064549. DOI: 10.1016/j.anucene.2022.109012. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454922000470 (visited on 07/08/2024).

[44] Yahui Wang, Yu Ma, and Ming Xie. “GPU accelerated lattice Boltzmann method in neutron kinetics problems”. In: *Annals of Nuclear Energy* 129 (July 1, 2019), pp. 350–365. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2019.02.009. URL: https://www.sciencedirect.com/science/article/pii/S0306454919300763 (visited on 06/28/2024).

[45] Yahui Wang, Yu Ma, and Ming Xie. “High-order lattice Boltzmann method for multi-group neutron diffusion solution”. In: *Progress in Nuclear Energy* 110 (Jan. 2019), pp. 341–353. ISSN: 01491970. DOI: 10.1016/j.pnucene.2018.10.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S014919701830266X (visited on 07/05/2024).

[46] Yahui Wang, Ming Xie, and Yu Ma. “Analysis of the multi-physics approach using the unified lattice Boltzmann framework”. In: *Annals of Nuclear Energy* 143 (Aug. 2020), p. 107500. ISSN: 03064549. DOI: 10.1016/j.anucene.2020.107500. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454920301985 (visited on 07/05/2024).

[47] Yahui Wang, Liming Yan, and Yu Ma. “Lattice Boltzmann solution of the transient Boltzmann transport equation in radiative and neutron transport”. In: *Physical Review E* 95.6 (June 23, 2017), p. 063313. ISSN: 2470-0045, 2470-0053. DOI: 10.1103/PhysRevE.95.063313. URL: http://link.aps.org/doi/10.1103/PhysRevE.95.063313 (visited on 07/05/2024).

[48] Yahui Wang et al. “High-order lattice Boltzmann framework and its adaptive mesh refinement in the neutron transport SP3 solutions”. In: *Progress in Nuclear Energy* 128 (Oct. 2020), p. 103449. ISSN: 01491970. DOI: 10.1016/j.pnucene.2020.103449. URL: https://linkinghub.elsevier.com/retrieve/pii/S0149197020302018 (visited on 07/08/2024).

[49] Mees Wortelboer. “Investigating GPU-accelerated Double Distribution Function Lattice Boltzmann Schemes for Heat Transfer and Phase Change in Turbulent Flows”. PhD thesis. Delft: Delft University of Technology, July 20, 2023.

[50] Yuankai Yang and Moran Wang. “Pore-scale modeling of chloride ion diffusion in cement microstructures”. In: *Cement and Concrete Composites* 85 (Jan. 2018), pp. 92–104. ISSN: 09589465. DOI: 10.1016/j.cemconcomp.2017.09.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S0958946516303729 (visited on 03/19/2025).

[51] Ting Zhang et al. “General bounce-back scheme for concentration boundary condition in the lattice-Boltzmann method”. In: *Physical Review E* 85.1 (Jan. 3, 2012), p. 016701. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.85.016701. URL: https://link.aps.org/doi/10.1103/PhysRevE.85.016701 (visited on 12/06/2024).

[52] Congshan Zhuo and Chengwen Zhong. “Filter-matrix lattice Boltzmann model for microchannel gas flows”. In: *Physical Review E* 88.5 (Nov. 25, 2013), p. 053311. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.88.053311. URL: https://link.aps.org/doi/10.1103/PhysRevE.88.053311 (visited on 09/09/2024).

[53] 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 (Nov. 25, 2016), pp. 543–553. ISSN: 1061-8562, 1029-0257. DOI: 10.1080/10618562.2016.1254777. URL: https://www.tandfonline.com/doi/full/10.1080/10618562.2016.1254777 (visited on 09/07/2024).

[54] Congshan Zhuo and Chengwen Zhong. “LES-based filter-matrix lattice Boltzmann model for simulating turbulent natural convection in a square cavity”. In: *International Journal of Heat and Fluid Flow* 42 (Aug. 2013), pp. 10–22. ISSN: 0142727X. DOI: 10.1016/j.ijheatfluidflow.2013.03.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0142727X13000702 (visited on 09/09/2024).

[55] Congshan Zhuo, Chengwen Zhong, and Jun Cao. “Filter-matrix lattice Boltzmann model for incompressible thermal flows”. In: *Physical Review E* 85.4 (Apr. 11, 2012), p. 046703. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.85.046703. URL: https://link.aps.org/doi/10.1103/PhysRevE.85.046703 (visited on 09/09/2024).

A

Neutronics and Precursor Data from Tiberga Benchmark Case

This appendix contains the data used in the simulations for the six neutron energy groups and the eight precursor families. The data was retrieved from the benchmark by Tiberga et al. [[37]]

A.1. Neutronics data

Table A.1: Total (removal) cross sections, fission cross sections, diffusion constants, and velocities for the six neutron energy groups. Retrieved from Tiberga et al. [37]

Group, g	$\Sigma_{t,g}\ [\mathrm{m}^{-1}]$	$\Sigma_{f,g}\ [\mathrm{m}^{-1}]$	$D_g\ [\mathrm{m}]$	$v_g\ [\mathrm{m\,s}^{-1}]$
1	1.65512	0.0111309	0.0280064	
2	2.17253	0.0108682	0.0184021	
3	3.18009	0.0152219	0.0113110	
4	2.42093	0.0258190	0.0144786	
5	2.50351	0.0536326	0.0139750	
6	2.72159	0.0144917	0.0128252	

Table A.3: P_0 scattering cross sections for the six neutron energy groups. Retrieved from Tiberga et al. [37]

	$\Sigma_{s,0,g'\to g}\ [\mathrm{m}^{-1}]$					
$g'\backslash g$	1	2	3	4	5	6
1	1.08476E+1	5.23316E+0	4.01805E−1	1.09869E−2	2.53290E−3	3.78334E−4
2	0	1.83666E+1	3.19138E+0	2.34218E−3	2.25259E−4	2.00405E−5
3	0	0	2.98293E+1	1.63470E+0	1.70575E−3	1.24625E−4
4	0	0	0	2.17472E+1	1.90243E+0	1.36858E−6
5	0	0	0	0	2.27173E+1	1.05885E+0
6	0	0	0	0	0	2.37826E+1

Table A.5: Average number of neutrons emitted per fission event, prompt neutron spectrum, delayed neutron spectrum, and average energy emitted per fission event for the six neutron groups. Retrieved from Tiberga et al. [37]

Group, g	$\nu_{tot,g}$ $[-]$	$\chi_{p,g}$ $[-]$	$\chi_{d,g}$ $[-]$	E_{fiss} [J]
1	2.85517	3.53812×10^{-1}	4.30325×10^{-3}	3.240722×10^{-11}
2	2.54532	5.23642×10^{-1}	3.87734×10^{-1}	3.240722×10^{-11}
3	2.43328	1.21033×10^{-1}	5.81848×10^{-1}	3.240722×10^{-11}
4	2.43127	1.35457×10^{-3}	2.27947×10^{-2}	3.240722×10^{-11}
5	2.43330	1.51226×10^{-4}	2.89130×10^{-3}	3.240722×10^{-11}
6	2.43330	7.37236×10^{-6}	4.28935×10^{-4}	3.240722×10^{-11}

A.2. Precursor data

Table A.7: Decay constants and delayed neutron fraction for the eight families of delayed neutron precursors. Retrieved from Tiberga et al. [37]

Family, d	λ_d $[\mathrm{s}^{-1}]$	β_d $[-]$
1	1.24667×10^{-2}	2.33102×10^{-4}
2	2.82917×10^{-2}	1.03262×10^{-3}
3	4.25244×10^{-2}	6.81878×10^{-4}
4	1.33042×10^{-1}	$1.37726\ e{-3}$
5	2.92467×10^{-1}	2.14493×10^{-3}
6	6.66488×10^{-1}	6.40917×10^{-4}
7	$1.63478 \times 10^{+0}$	6.05805×10^{-4}
8	$3.55460 \times 10^{+0}$	1.66016×10^{-4}

B

Heatmaps of Simulation Results from Tiberga Benchmark Case

B.1. Phase 0: Single-Physics Simulations

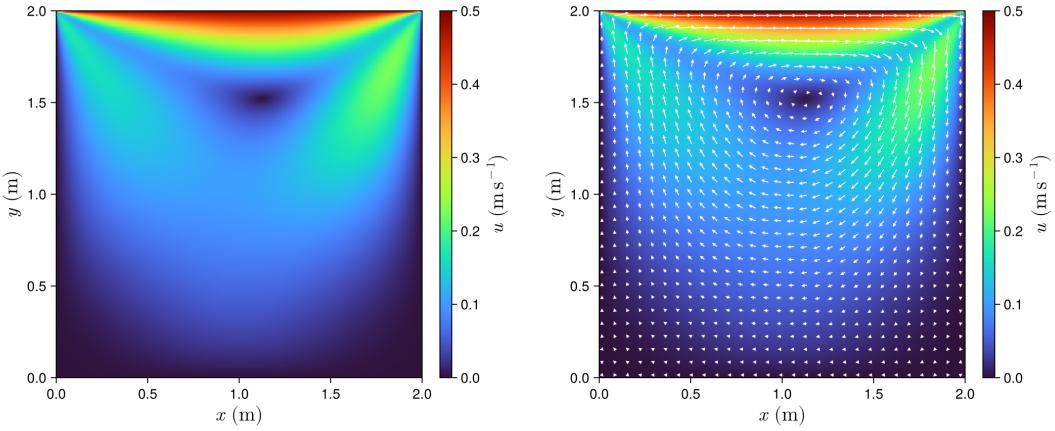


Figure B.1: Heatmap of the simulated velocity field from step 0.1 of the Tiberga benchmark case. In the right plot, arrows denote the flow direction of the fluid. The arrows are scaled to the local velocity magnitude. The FMLBM algorithm was used on a 200×200 grid.

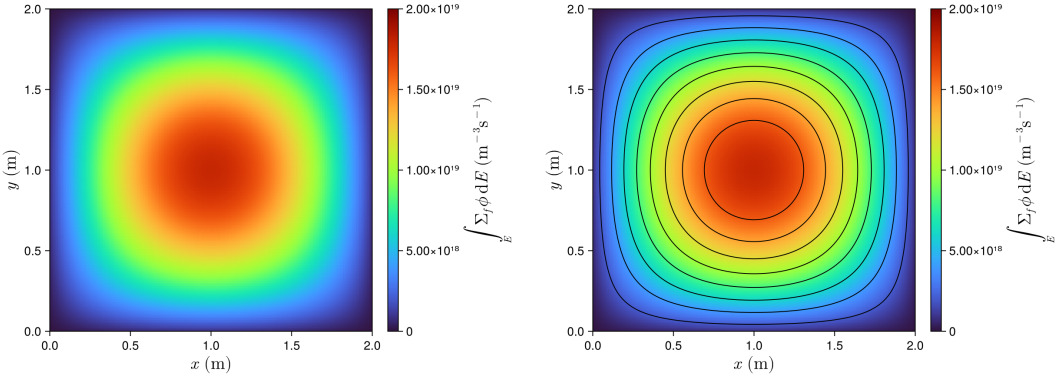


Figure B.2: Heatmap of the simulated fission density from step 0.2 of the Tiberga benchmark case. In the right plot, isoline intervals of 2.0×10^{18} m^{-3} s^{-1} are drawn. The FMLBM algorithm was used on a 200×200 grid.

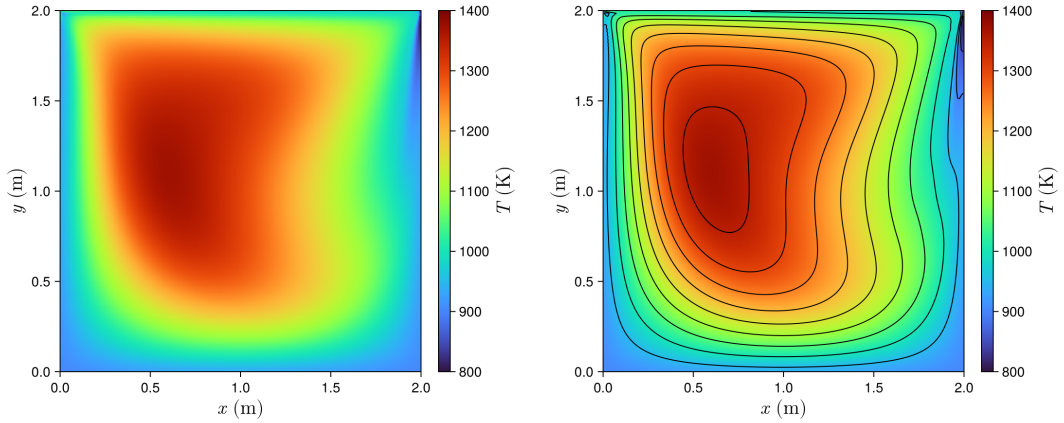


Figure B.3: Heatmap of the simulated temperature field from step 0.3 of the Tiberga benchmark case. In the right plot, isoline intervals of $50\ \mathrm{K}$ are drawn. The FMLBM algorithm was used on a 200×200 grid with $\mathrm{Pr} = 1200$.

B.2. Phase 1: Coupled Steady-State Simulations

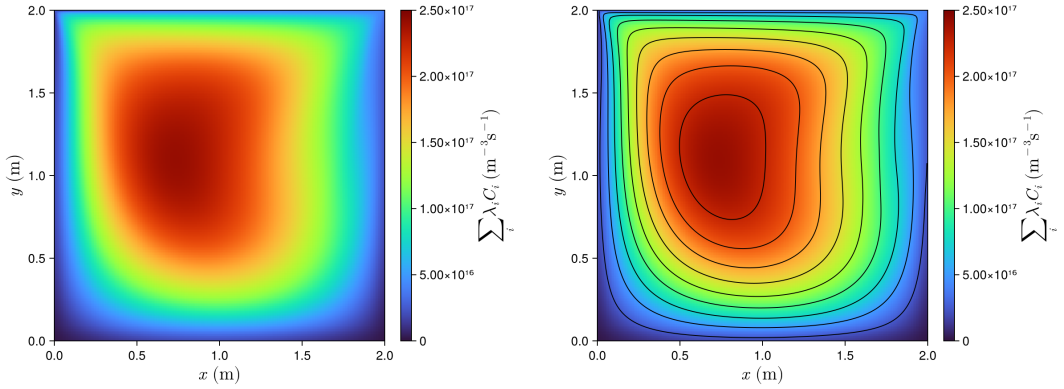


Figure B.4: Heatmap of the simulated delayed neutron source from step 1.1 of the Tiberga benchmark case. In the right plot, isoline intervals of $2.5 \times 10^{16}\ \mathrm{m}^{-3}\,\mathrm{s}^{-1}$ are drawn. The FMLBM alogrithm was used on a 200×200 grid with $\mathrm{Sc} = 1200$.

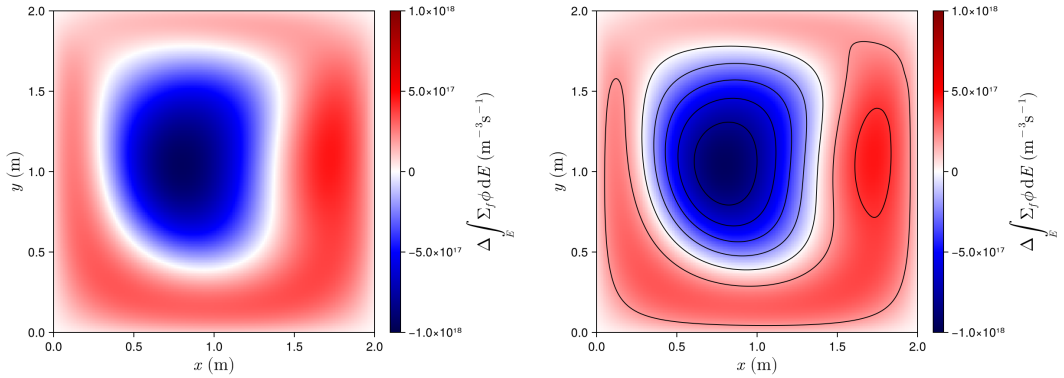


Figure B.5: Heatmap of the simulated delta fission density from step 1.2 of the Tiberga benchmark case. In the right plot, isoline intervals of $2.0 \times 10^{17}\ \mathrm{m}^{-3}\,\mathrm{s}^{-1}$ are drawn. The FMLBM algorithm was used on a 200×200 grid with $\mathrm{Pr} = \mathrm{Sc} = 1200$.

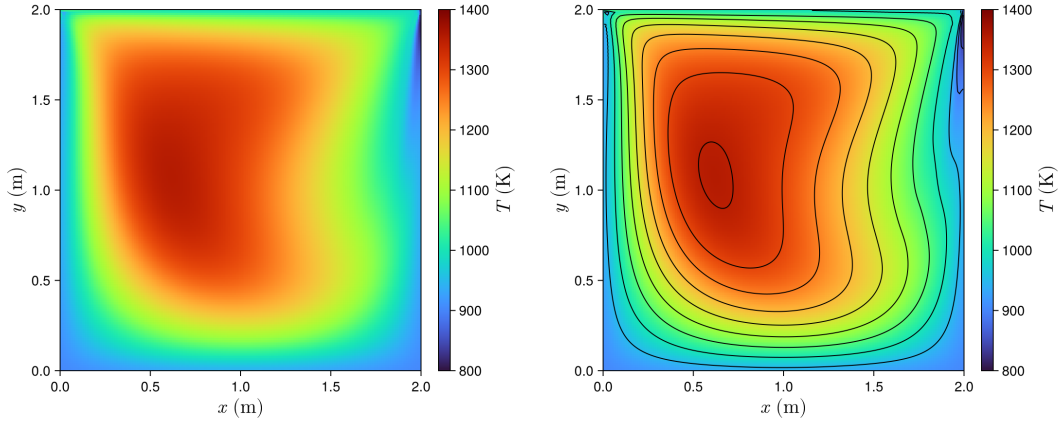


Figure B.6: Heatmap of the simulated temperature field from step 1.2 of the Tiberga benchmark case. In the right plot, isoline intervals of $50\ \mathrm{K}$ are drawn. The FMLBM algorithm was used on a 200×200 grid with $\mathrm{Pr} = \mathrm{Sc} = 1200$.

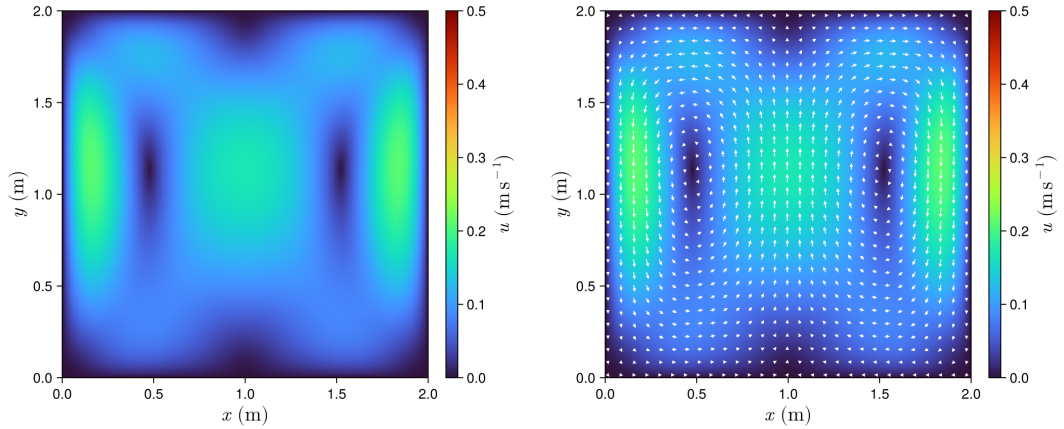


Figure B.7: Heatmap of the simulated velocity field from step 1.3 of the Tiberga benchmark case. In the right plot, arrows denote the flow direction of the fluid. The arrows are scaled to the local velocity magnitude. The FMLBM algorithm was used on a 200×200 grid with $\mathrm{Pr} = \mathrm{Sc} = 1200$.

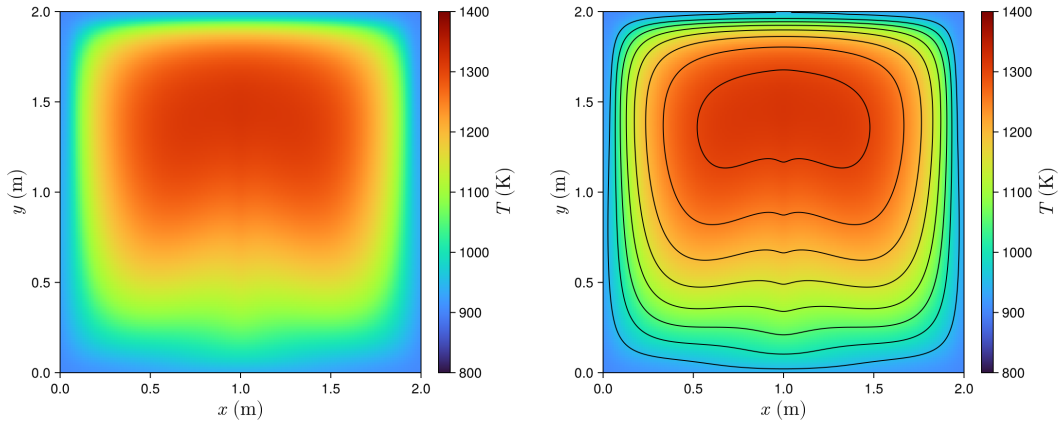


Figure B.8: Heatmap of the simulated temperature field from step 1.3 of the Tiberga benchmark case. In the right plot, isoline intervals of $50\ \mathrm{K}$ are drawn. The FMLBM algorithm was used on a 200×200 grid with $\mathrm{Pr} = \mathrm{Sc} = 1200$.

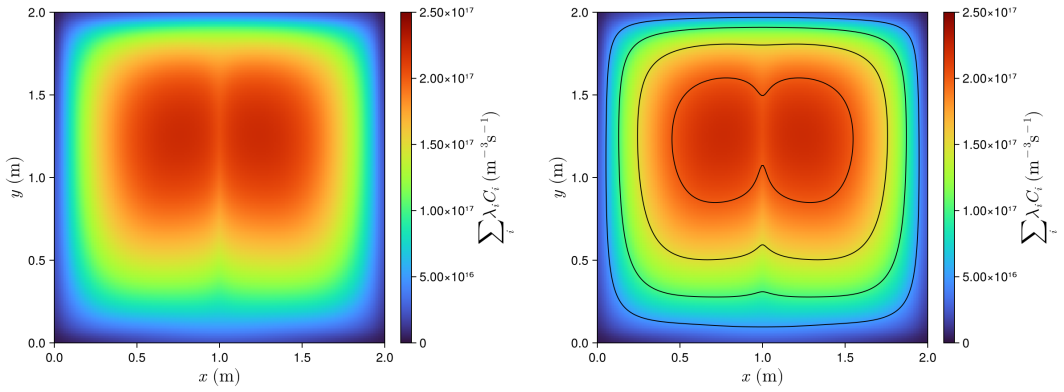


Figure B.9: Heatmap of the delayed precursor source from step 1.3 of the Tiberga benchmark case. In the right plot, isoline intervals of $5 \times 10^{16}\ \mathrm{m}^{-3}\,\mathrm{s}^{-1}$ are drawn. The FMLBM algorithm was used on a 200×200 grid with $\mathrm{Pr} = \mathrm{Sc} = 1200$.

Table B.1: Heatmap of the simulated temperature field from step 1.4 of the Tiberga benchmark case. The top row shows simulations with $U_{lid} = 0.1\ \mathrm{m\,s^{-1}}$, the middle row with $U_{lid} = 0.3\ \mathrm{m\,s^{-1}}$, and the bottom row with $U_{lid} = 0.5\ \mathrm{m\,s^{-1}}$. Isolines are drawn for intervals of $25\ \mathrm{K}$ for the left column, $33\ \mathrm{K}$ for the middle column, and $50\ \mathrm{K}$ for the right column. The results were generated using the FMLBM algorithm on a 200×200 grid with $\mathrm{Pr} = \mathrm{Sc} = 1000$. Note that the colobar range differs between the plots.

$P = 0.2$ GW	$P = 0.6$ GW	$P = 1.0$ GW

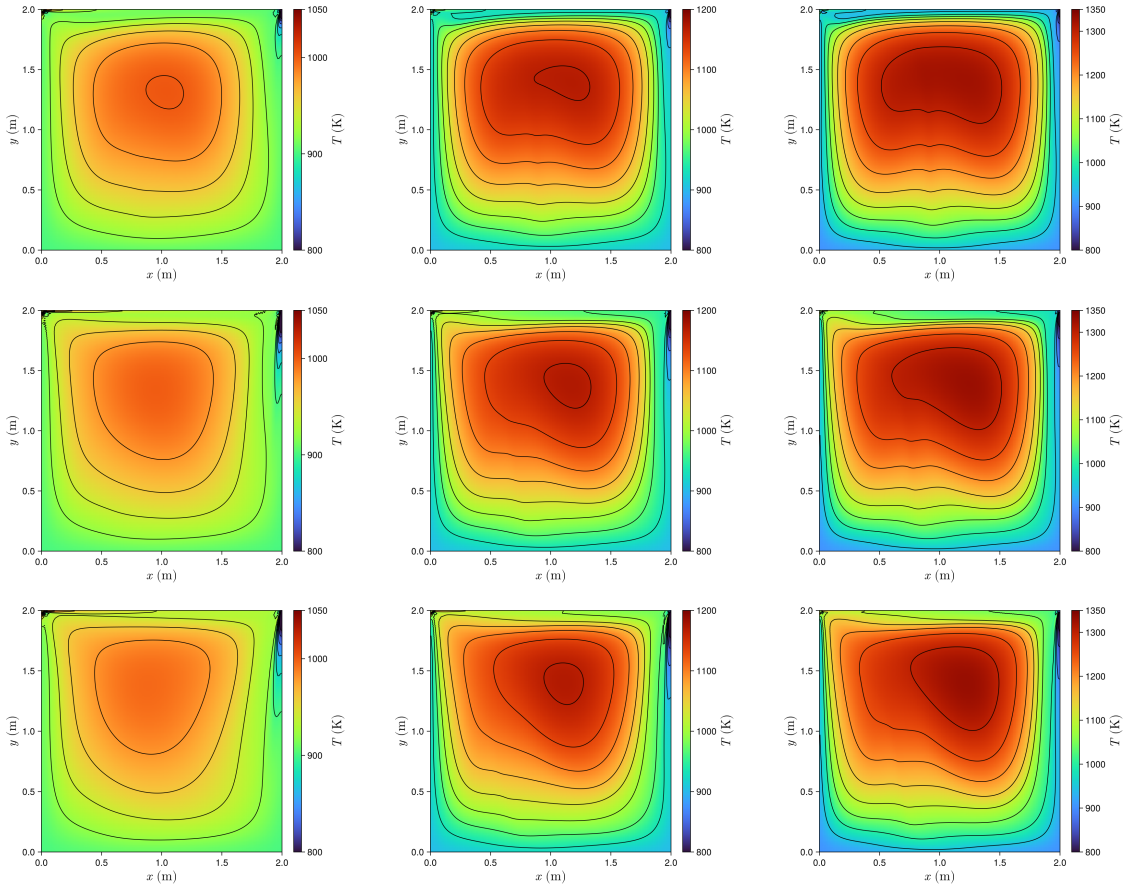
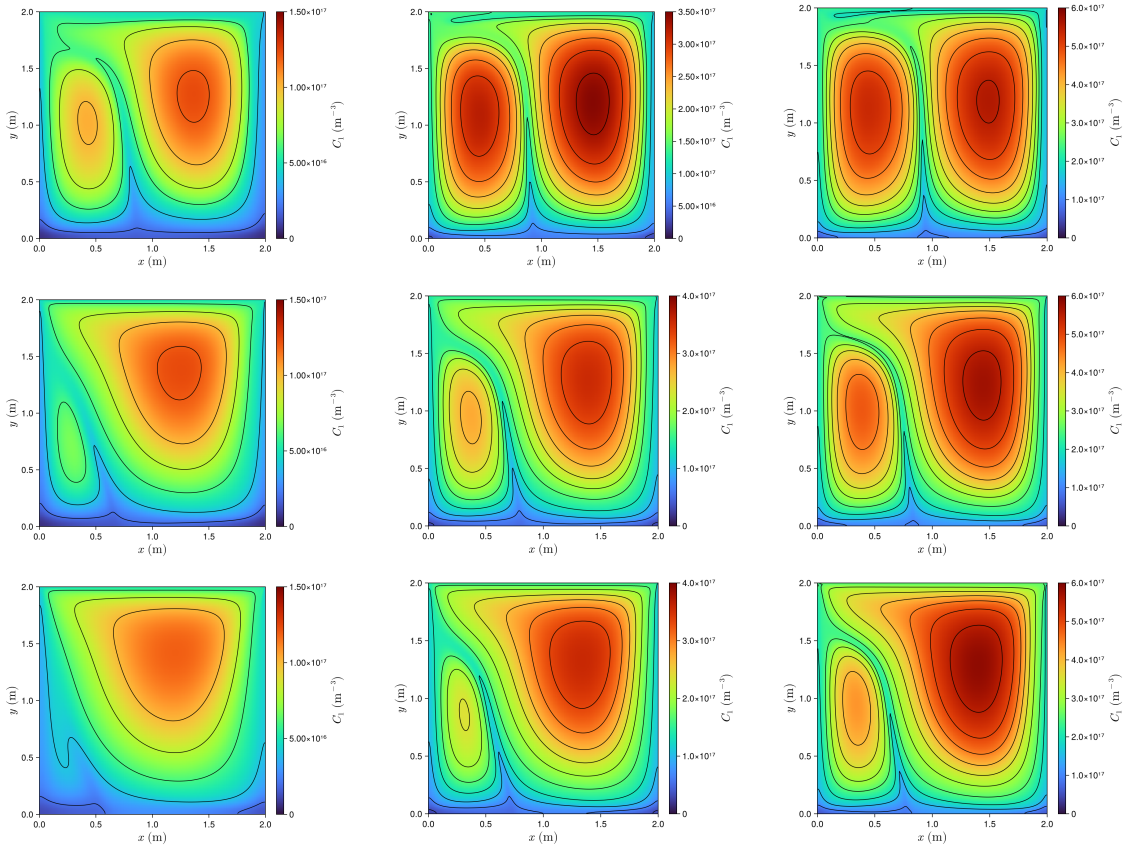


Table B.2: Heatmap of the simulated concentration of the first delayed precursor family from step 1.4 of the Tiberga benchmark case. The top row shows simulations with $U_{lid} = 0.1\ \mathrm{m\,s^{-1}}$, the middle row with $U_{lid} = 0.3\ \mathrm{m\,s^{-1}}$, and the bottom row with $U_{lid} = 0.5\ \mathrm{m\,s^{-1}}$. Isolines are drawn for intervals of $2.0 \times 10^{16}\ \mathrm{m^{-3}}$ for the left column, $3.3 \times 10^{16}\ \mathrm{m^{-3}}$ for the middle column, and $5.0\ \mathrm{m^{-3}}$ for the right column. The results were generated using the FMLBM algorithm on a 200×200 grid with $\mathrm{Pr} = \mathrm{Sc} = 1000$. Note that the colorbar range differs between the plots.

$P = 0.2$ GW	$P = 0.6$ GW	$P = 1.0$ GW
		

C

Snapshots of Transient Simulation Results

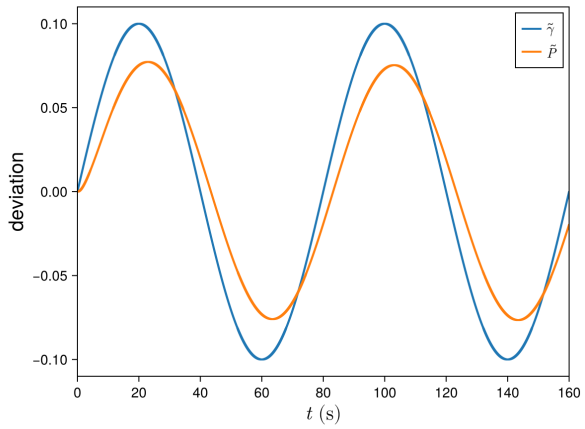


Figure C.1: Power response to the perturbation of the heat transfer coefficient with $f_{pert} = 0.0125$ Hz.

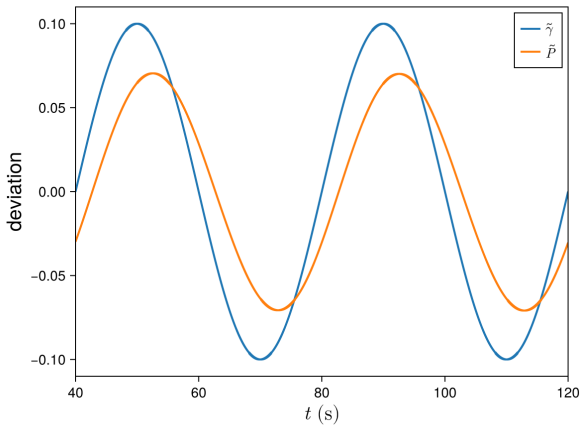


Figure C.2: Power response to the perturbation of the heat transfer coefficient with $f_{pert} = 0.025$ Hz.

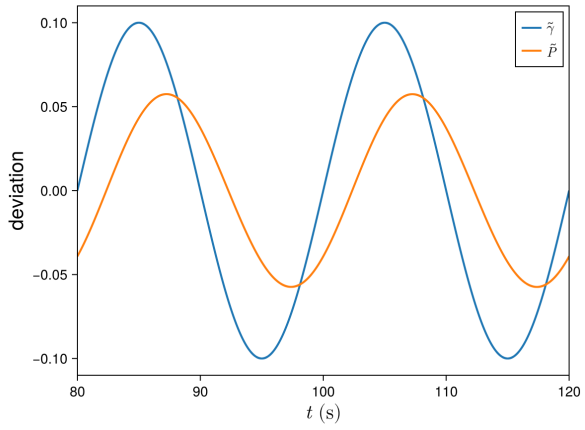


Figure C.3: Power response to the perturbation of the heat transfer coefficient with $f_{pert} = 0.05$ Hz.

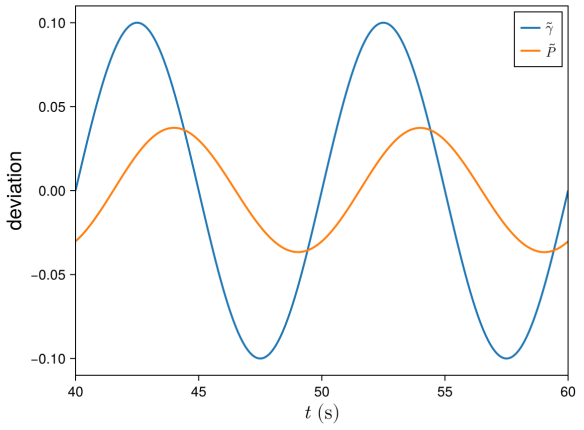


Figure C.4: Power response to the perturbation of the heat transfer coefficient with $f_{pert} = 0.1$ Hz.

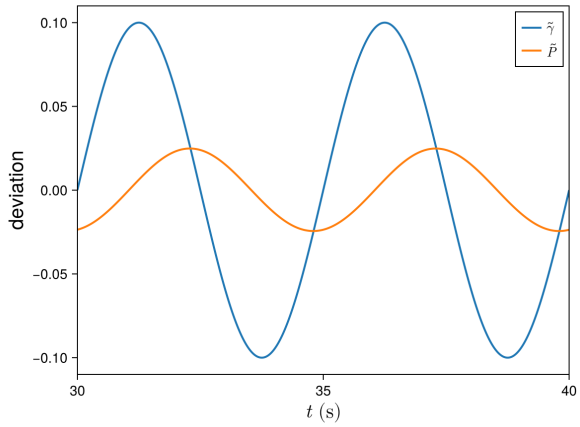


Figure C.5: Power response to the perturbation of the heat transfer coefficient with $f_{pert} = 0.2$ Hz.

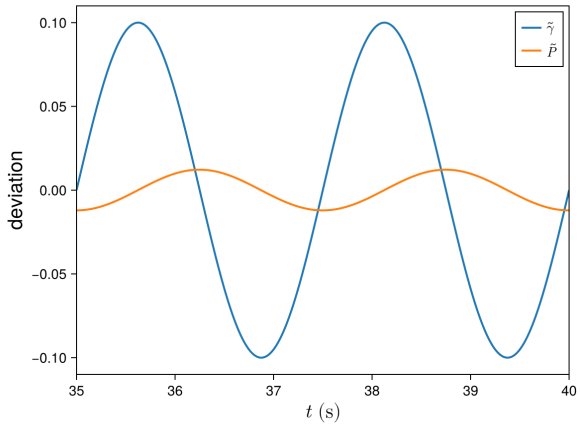


Figure C.6: Power response to the perturbation of the heat transfer coefficient with $f_{pert} = 0.4$ Hz.

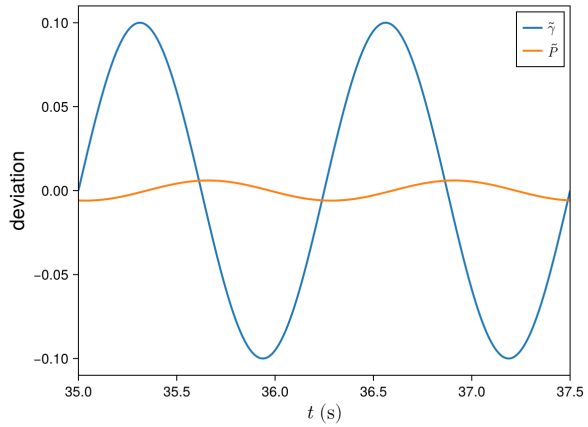


Figure C.7: Power response to the perturbation of the heat transfer coefficient with $f_{pert} = 0.8$ Hz.