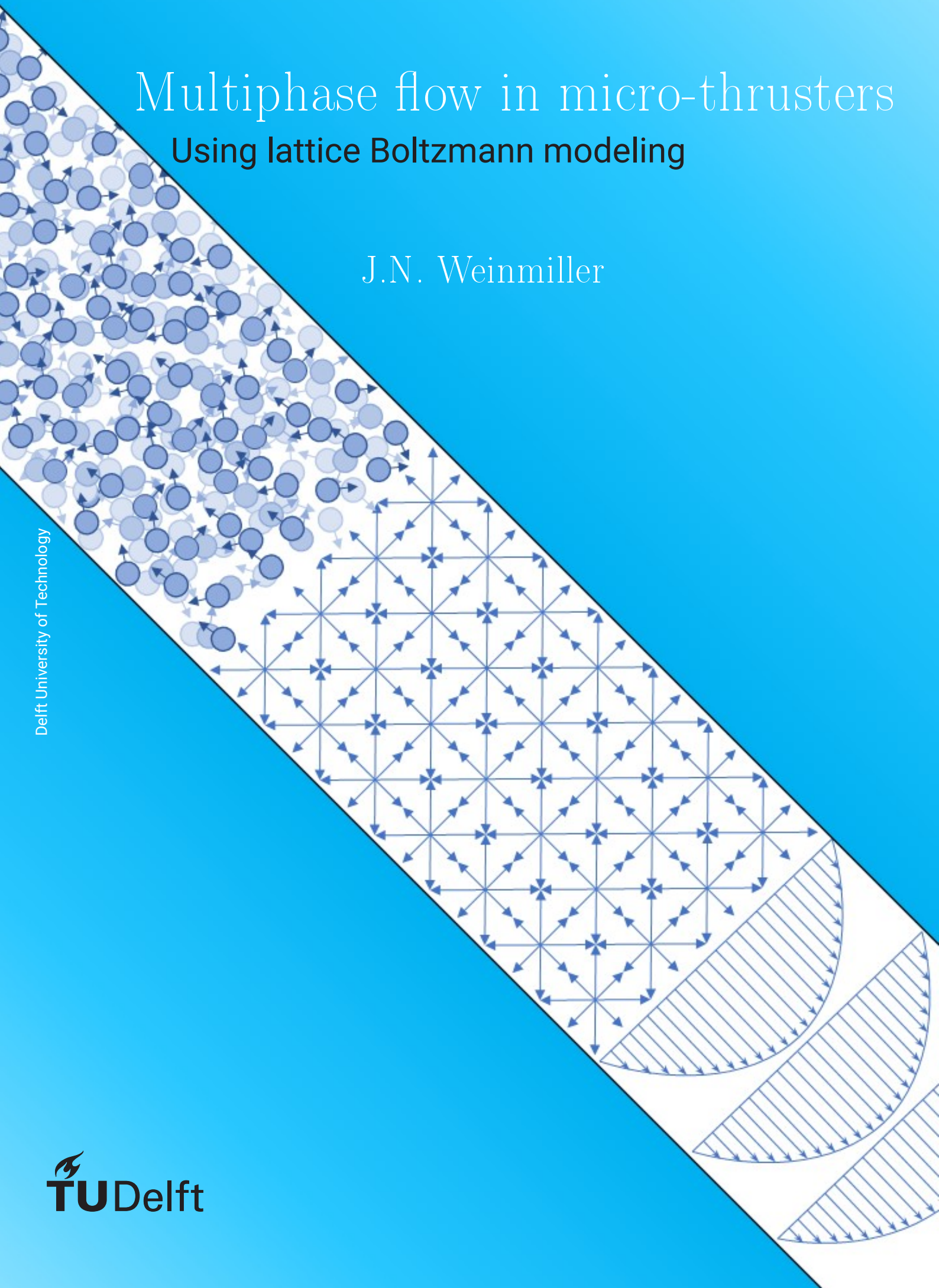
# Multiphase flow in micro-thrusters

## Using lattice Boltzmann modeling

J.N. Weinmiller



Delft University of Technology

TU Delft

# Multiphase flow in micro-thrusters

## Using lattice Boltzmann modeling

by

Julius Nikolaus Weinmiller

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday January 21, 2021 at 09:00 AM.


| | | |
|---|---|---|
| Student number: | 4277597 | |
| Thesis committee: | Asst. Prof. Dr. A. Cervone , | TU Delft, Committee chair |
| | Ir. B.T.C. Zandbergen, | TU Delft, Daily supervisor |
| | Asst. Prof. Dr. L. Portela , | TU Delft, External examiner |




TU Delft

# Acknowledgements

It has almost been a year since I handed in my thesis proposal in the beginning of December 2019. In some parts it feels like yesterday, in others a lifetime ago, which I attribute to the seemingly time warping pandemic that occurred in the meantime. While this resulted in academic discussions becoming more sparse, each one carried more importance, making each more memorable.

As such I want to express my gratitude to my supervisor who has made the most memorable impact. When I approached B.T.C. Zandbergen regarding my master thesis, we found a topic that is both of interest to him and played to my strengths. Over the course of my thesis he was always available to talk to and made sure that I stayed on track.

Special thanks go to Dr. S. Mukherjee and Dr. L. Portela for their help in answering my questions regarding multiphase flow and lattice Boltzmann method. The OpenLB community was helpful in getting a few specific code problems sorted.

To all the students in the master room who sat with me pre-pandemic; the think-tank sessions are missed. Finally, I want to thank my friends and family who patiently listened to me talking when I needed someone to listen to me going through my problems.

*J. Weinmiller*
*The Hague, November 2020*

# Abstract

Even though hundreds of CubeSats have been launched, few have launched with a micropropulsion module on board. Propulsion would allow for an extended lifetime and better mission capabilities, thus greatly increasing the attractiveness of CubeSats. TU Delft is working on a type of thruster suitable for CubeSats, called the Vaporizing Liquid Microthruster (VLM). For micro-propulsion it is viable to generate sufficient thrust by using an external heat source to vaporize a liquid and expanding the vapour via a nozzle. The usage of such micropropulsion modules are still inhibited by issues in the nozzle and heating chamber. One issue is the multiphase flow occurring inside the micronozzle, which lowers the thrust and specific impulse performance of the thruster as well as the stability due to the explosive boiling phenomena.

Previous research indicates that the multiphase flow in the nozzle most likely occurs inside the heating chamber and flows into the nozzle. Little is known on how exactly this multiphase flow looks like, ranging from annular flow to dispersed droplets, nor on how it is generated within the chamber. It is indicated that the boiling may lead to large pressure oscillations inside the chamber, which contributes to the instability of the thruster. These pressure oscillations occur due to multichannel interactions from nucleate boiling in microchannels, as well as explosive boiling.

In this thesis, the lattice Boltzmann method (LBM) is used to simulate the complex phase change multiphase flow occurring inside the microchannels of the vaporizing liquid micro-thruster. This method has recently gained a lot of attraction in simulating microscale fluid problems. Research shows that LBM applied to multiphase flows can be an order of magnitude faster than conversional Navier-Stokes solvers.

This thesis utilizes the Bhatnagar–Gross–Krook (BGK) collision operator in combination with the pseudopotential method to simulate the multiphase flow. A modified Guo forcing scheme is used to ensure thermodynamic consistency. The water is modelled using the non-ideal Peng-Robinson equation of state. The thermal problem is solved using the double distribution function method, in combination with a semi-hybrid source term solved via a multiple relaxation time (MRT) collision operator. The phase change term is derived from the local balance of entropy, using the equation of state to calculate the latent heat.

The open-source lattice Boltzmann method solver OpenLB is extended to be capable of simulating phase change flow. Verification tests are performed to show that the code extension is correctly implemented. The code was released to the OpenLB, making it the first open-source lattice Boltzmann method framework capable of simulation phase change flow.

In the analysis of the tool developed, the pool boiling simulations showed how nucleation, bubble formation, and departure occurs. Correct behaviour are shown in the thermodynamic consistency, Laplace pressure and wall wettability. The effect of the spurious currents on the thermal solution are analysed in more depth than found in literature.

The procedure for the multichannel simulation verification and validation is given, including a novel analysis method. The novel analysis method can estimate the multiphase flow type occurring inside real thrusters without any additional equipment required in a test setup. The required experiment data are the specific impulse values at various massflow rates, and the massflow rate at which multiphase flow starts to occur.

The simulations are strongly limited by the BGK collision operator, which is apparent by the large spurious currents at high density ratio multiphase flow. The spurious currents can cause simulation instabilities, but are mitigated by using a thicker diffuse interface. This mitigation results in a larger metastable phase change region, which inhibits the nucleation process. Thus, nucleation inside the microchannels did not occur. However, using a MRT collision operator should reduce the spurious currents without needing to increase the diffuse interface thickness, allowing for nucleation to take place in the microchannels.

# Contents

# List of Figures

# List of Tables

# Listings

# Nomenclature

| | |
|---|---|
| AD | Advection Diffusion |
| BB | Bounce Back |
| BE | Boltzmann equation |
| BGK | Bhatnagar Gross and Krook |
| DDF | Double distribution function |
| EDM | Exact Difference Method |
| EoS | Equation of State |
| FD | Finite difference |
| GNU | GNU is not Unix |
| GSL | GNU Scientific Library |
| LBE | Lattice Boltzmann equation |
| LBGK | Lattice Bhatnagar Gross and Krook |
| LBM | Lattice Boltzmann method |
| MRT | Multiple Relaxation Time |
| MPI | Message Passing Interface |
| NEBB | Non Equilibrium Bounce Back |
| NIST | National Institute of Standards and Technology |
| NS | Navier-Stokes |
| NSE | Navier Stokes Equations |
| OpenLB | Open source Lattice Boltzmann |
| OpenFOAM | Open source Field Operation And Manipulation |
| Palabos | Parallel Lattice Boltzmann Solver |
| PR | Peng-Robinson |
| PP | Pseudopotential |
| SC | Shan-Chen |
| SC | Single Component |
| SciPy | Scientific Python |
| SRT | Single Relaxation Time |
| TRT | Two Relaxation Time |
| TU Delft | Delft University of Technology |
| RAM | Random Access Memory |
| RQ | Research question |
| VLM | Vaporizing Liquid Micro-thruster |
| VoF | Volume of Fluid |
| ZOT | Zero-One-Three |

# List of Symbols

| | |
|---|---|
| $\Delta t$ | Lattice timestep |
| $\dot{m}_g$ | Vapourized massflow |
| $\dot{m}_l$ | Unvaporized massflow |
| $\dot{m}$ | Massflow |
| $\gamma$ | Surface tension |
| $\Lambda$ | Relaxation time diagonal matrix |
| $\Lambda_{TRT}$ | Magic Parameter |
| $\mathbb{R}$ | Universal gas constant |
| $\mathbf{c}$ | Lattice velocity set |
| $\mathbf{c}_i$ | Individual lattice velocity vector of a lattice velocity set |
| $\mathbf{F}$ | Force vector |
| $\mathbf{M}$ | Matrix to convert from velocity to moment space |
| $\mathbf{m}$ | Particle distribution function in moment space |
| $\mathbf{u}$ | Velocity vector |
| $\mathbf{v}'$ | Modified force shifted lattice velocity vector |
| $\mathbf{v}$ | Force shifted lattice velocity vector |
| $\mathbf{w}$ | Lattice weight set |
| $\mathbf{x}$ | Space vector |
| $\nu$ | Viscocity |
| $\Omega$ | Collision operator |
| $\omega_{EoS}$ | Equation of State Acentric Factor |
| $\partial$ | Partial derivative operator |
| $\phi$ | Unvaporized massflow fraction |
| $\psi$ | Pseudopotential |
| $\psi_0$ | Pseudopotential reference potential |
| $\rho$ | Macroscopic density |
| $\rho_0$ | Pseudopotential reference density |
| $\sigma$ | Force shifted velocity vector correction parameter |
| $\tau$ | Relaxation time |
| $\tau_g$ | Secondary population relaxation time |
| $\xi$ | Microscopic particle velocity |
| $\beta$ | Dimensional index |
| $a$ | Van der Waals EoS parameter: measure of the average interparticle attraction |
| $b$ | Van der Waals EoS parameter: measure of particle size |
| $C$ | Secondary lattice tracked concentration |
| $c$ | Equation of State parameter |
| $c_g$ | Specific heat capacity of gas phase |
| $c_l$ | Specific heat capacity of liquid phase |
| $c_p$ | Specific heat at constant pressure |
| $c_s$ | Lattice speed of sound |
| $c_v$ | Specific heat at constant volume |
| $D$ | Thermal diffusion |
| $D_b$ | Bubble departure diameter |
| $E$ | Total energy |
| $e$ | Internal energy |
| $f$ | Particle distribution function |
| $f^{eq}$ | Equilibrium particle distribution function |
| $f_i$ | Individual particle distribution function scalar of a particle distribution function set |

| | |
|---|---|
| $f_i^{eq}$ | Individual equilibrium particle distribution function scalar |
| $G$ | Pseudopotential interaction strength |
| $g$ | Secondary particle distribution function |
| $g_i$ | Secondary population along the $i^th$ velocity |
| $h_{fg}$ | Latent heat |
| $I_{sp}$ | Specific impulse |
| $M$ | Molar mass |
| $p$ | Pressure |
| $P_{EoS}$ | Pressure of an equation of state |
| $P_s$ | Saturation pressure of Maxwell construction |
| $Q_i$ | Secondary population source term along the $i^th$ velocity |
| $R$ | Gas constant |
| $T$ | Temperature |
| $t$ | Time |
| $T_c$ | Critical temperature |
| $Tr$ | Temperature Ratio |
| $v_{fg}$ | Difference in specific volume between state $f$ and $g$ |
| $v_{sp,l}$ | Specific volume for liquid phase |
| $v_{sp,v}$ | Specific volume for vapour phase |
| $v_{sp}$ | Specific volume |
| $w_i$ | Individual lattice weight scalar of a lattice weight set |

# 1 Introduction

The current space industry trend focuses on smaller satellites that retain the capabilities of their larger counterpart. The CubeSat standard enabled easier integration between spacecraft and launcher, creating the possibility to piggyback ride on launches for larger spacecraft. This resulted in large numbers of CubeSats and NanoSats launched, see fig. 1.1. Nanosats.eu state that in their database, "'nanosatellite' covers all CubeSats, PocketQubes, TubeSats, SunCubes, ThinSats and non-standard picosatellites" and include spacecraft from 100g to 10kg in addition to lighter SunCubes [66, 67].

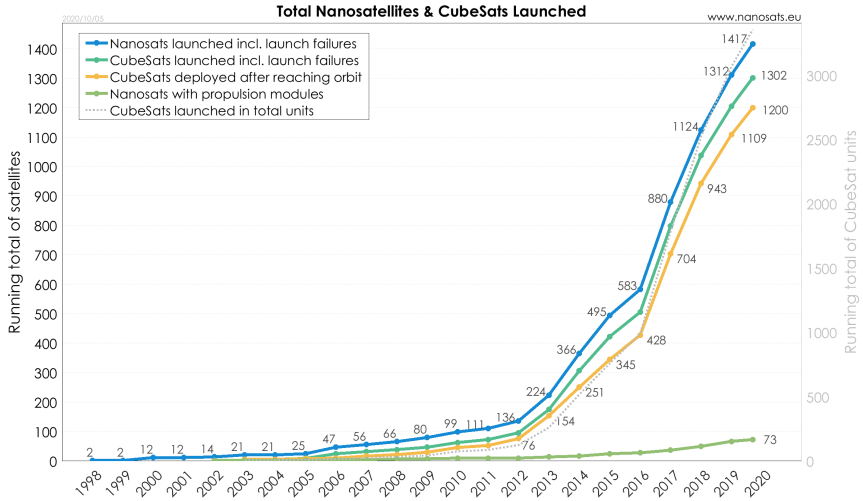



Figure 1.1: Total launched nanosats [66]

To retain all capabilities, small satellites also need to be able to perform orbital manoeuvre, which requires a propulsion module. This module can also extend the spacecrafts lifetime and allow for a controlled end of life. However, from fig. 1.1, only 73 out of the 1417 Nanosats total launched up to year 2020 included a propulsion module, which is barely more than 5%.

The problem encountered is that phenomena which are small for macro propulsion, and produce small losses, are large when miniaturized, and thus produce large losses [147]. An example are in micro nozzles the large viscous losses due to the much lower Reynolds number, the large surface-to-volume ratio, and relative impact of surface roughness [140].

The small spacecrafts either launch with many others small spacecraft or piggyback on larger spacecraft. Thus it is vital that they do not interfere nor pose a large potential risk. Explosive propellants, which is the propellant of choice for most macro propulsion systems, are a risk and thus other safer propellants are needed [142].

Delft University of Technology (TU Delft) are developing with their Delfi Program [28] small spacecrafts. Part of the TU Delft's Aerospace Engineering faculty is the Space Engineering department, which has as one of their research topics the miniaturization of propulsion [27]. Current research focuses on the Vaporizing Liquid Micro-thruster (VLM) and Low Pressure Micro-resistojet. An example schematic of a VLM is shown in fig. 1.2.

The VLM, as the name implies, vaporizes a liquid in its chamber and uses the superheated vapour to produce thrust by expanding it via a nozzle. Any liquid can be used, but water is great for testing purposes since it

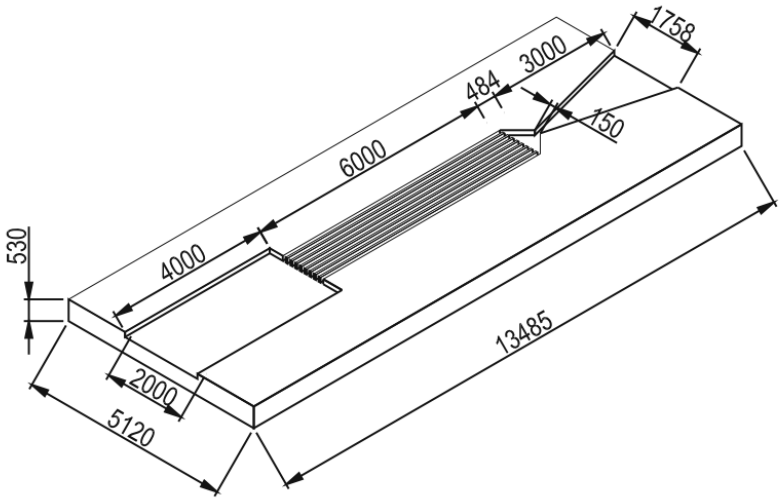



Figure 1.2: VLM using straight channels with dimensions (unit: $\mu$m) by Cen and Xu [14]

is safe and well studied. The vaporization process causes a energetic and complex interaction between the vapour and liquid. This may lead to undesirable outcomes, such as unvaporized flow exiting the chamber, which can negatively impact the thruster performance [14, 17, 48, 114]. This is a phenomena not unique to VLM, and similar situations for heated microchannels are detailed in literature, [3, 24, 50, 58, 60, 85, 92, 105, 131, 149, 156], but the focus is on heat transfer and not full evaporation. A detailed overview of multiphase flow in microchannels was performed in the literature study preceding this thesis.

The layout of the heating chamber and the thruster operational point define how the vaporization process progresses. Various layouts of the chamber have been tested at TU Delft, from multichannel [15, 69, 94, 128], to stacked pillar [128, 145], immersed fins [15, 69] and porous media [144].

To avoid multiphase flow in the nozzle to interfere with the test results "the power is manually controlled to allow full vaporization (visually) of the propellant" [128]. Spacecrafts have a limited power budget and thus a low power usage thruster is preferred. As such there is a yet unfulfilled need to design a layout which minimizes the power usage while fully vaporizing the liquid water.

While there are a few numerical studies on the design of the heating chamber [129], few simulate the phase change inside. Previous attempts using phase-field in COMSOL have had some success, but said to be unsuitable due to large computational cost, and a Navier-Stokes based approach has been met with instabilities [52]. As such the flow inside the heating chamber is still a big open question in the design of the VLM. From this need, and the literature study performed, the following research objective is derived which is:

> *to provide quantitative data regarding the unvaporized flow characteristics exiting the heating chamber of the VLM and investigate how design parameters influence the unvaporized flow aiding further research into performance impact assessment and development mitigation methods.*

## Research questions

With the research goal set, the following research questions were drawn up:

RQ-1 What is the influence of design parameters on the unvaporized flow characteristics and flow instabilities for straight channel heating chambers?

RQ-2 What are the unvaporized flow characteristics exiting the heating chamber resulting from the numerical simulation?

RQ-3 How agreeable are the conclusions drawn from mathematical analysis to the ones drawn from the numerical simulations, with regard to the unvaporized flow characteristics, and influence of the design parameters?

In the above, the term design parameters refer to the operational point, heating chamber parameters, and fluid properties. The goal of the thesis to get a better understanding of the fluid mechanics inside the heating chamber, and not perfecting the heating chamber design, and as such the focus will be related to the fluid dynamics. The relevant parameters considered in this thesis are presented below:

- The operational point refers to the substrate temperature, mass flow rate, and whether gravity is taken into account or not.
- The heating chamber parameters can encompasses a lot. To name a few: channel length, size, shape, cross-sectional geometry and amount of channels, inlet/outlet plenum shape and size, ect. For this thesis, most chamber parameters are fixed. A constant height of $100\mu m$, cross sectional geometry is rectangular, the channel shape is straight and inlet/outlet plenum shapes are trapezoidal. The amount of channels are varied to see its impact on cross channel instabilities. If there is time, the size and length effects are investigated, as well as a chamber made of porous medium.
- The fluid properties that are of interest is: surface tension, wettability, viscosity, specific heat capacity. Not mentioned parameters are not varied, most notably: density, molar mass.

A successful thesis will be able to answer the above research questions. However, it is rather difficult to assess whether or not a question has been sufficiently answered. As such, the following requirements can be set up for each of the research questions. This allows answering how well each research question has been answered.

To answer RQ-1, the design parameters need to be varied. Thus the following requirements are given:

1. The design parameters shall be varied individually by ±10% and ±20%. This should give an indication on the effect of the design parameters on the unvaporized flow.
2. The design parameters shall be varied in pairs by ±10%. This should give some insight how two different parameters act interact with each other. Only the ±10% is chosen since it overlaps with the previous simulations, but limits the total amount of simulations needing to be run.

To answer RQ-2, a proper simulation environment needs to be set up. The following requirements are related to this simulation:

1. The simulations shall be applicable to the VLM. This means that the simulation needs to encompass thermal, phase change and multiphase fluid flow models. Since the VLM is very small in size, it borders on the edge of the macroscopic simulations models.
2. The simulation shall be three dimensional. 2D simulations impact the size and distributions of the droplets since three dimensions allow for two small droplets to vertically overlap, which would otherwise coalesce into a single big droplet.
3. The simulations shall encompass compressible fluid flow. For bubble expansion and explosive boiling to propagate effectively via density waves, the simulations should be at least weakly compressible.
4. The simulation's spatial resolution shall allow for accurate simulation of the expected minimum relevant droplet size. The expected minimum relevant droplet size can be calculated via a conservative evaporative estimate, such that a droplet is fully evaporated from the exit of the heating chamber to the nozzle throat. The simulations should allow for proper resolution of the interface, which depending on the method used, requires an order of magnitude smaller than the minimum droplet size, i.e. 10 units for a minimum drop of 1 unit.
5. The simulation's temporal accuracy shall allow for proper simulation at the selected spatial resolution.

6. The simulation duration shall be enough for a quasi-steady flow to emerge. An appropriate duration could be at minimum 4 times the time it takes for a fully gaseous flow to flow from the inlet boundary to the outlet boundary; the first period is to establish initial flow, the second to smooth out setup errors, the 3rd and 4th to evaluate the quasi-steady period if it exists. Double check whether this simulation time is within the quasi steady period prediction.

7. The simulation shall calculate any pressure oscillation period and magnitude with at least 1.5 standard deviation. This is a secondary time constraint which forces any prediction of the oscillation period to be within 86% of the value. While this does not guarantee accuracy, it does give some precision.

8. The simulation physical boundaries, i.e. the heating chamber walls, have a fixed wall temperature and a no slip condition. Since inside the heating chamber, the continuum assumptions is still valid, a no-slip boundary condition is appropriate. The fixed temperature condition is less appropriate, but greatly simplifies the simulation complexity.

9. The simulation save intervals shall allow identification of flow patterns in the quasi steady state solution, as well as for identification of the unvaporized liquid.

10. The simulation shall predict the correct flow patterns 95% of the time. The prediction correct flow patterns are the foundation of multiphase flow, and if the simulation cannot predict those, any results will be very questionable.

11. The simulation shall be able to predict flow transition location within 15% accuracy. This will be used for validation, and 15% is said to be neither too ambitions nor too lax for comparisons.

To answer RQ-3, the simulations are compared to the analysis. This question is there to assess the accuracy of the simulation and does not pose any new requirement on the discrepancy. A few requirements on what is compared and how these comparisons are presented do exist.

1. The analysis shall compare the flow patterns predicted by flow pattern with the simulations.

2. The analysis shall compare the droplet dynamics using the Basset–Boussinesq–Oseen equations and to the path predicted by the simulations.

3. The analysis shall compare a simple droplet evaporation model with evaporative droplet simulations.

4. The analysis shall compare the film breakup with resulting droplet size estimations to the droplets simulated to occur from film breakage.

5. The comparisons shall express the accuracy in absolute and percentage terms. If possible, a confidence interval for the simulation is included, thus also reporting how likely the simulation is accurate.

6. The analysis shall include a sensitivity analysis of the design parameters. This is done by taking the derivative of the design parameter with respect to the relevant analysis. The result will be used to analyse the results of RQ-1.

## Thesis approach

A trade-off was performed in the literature study, shown in appendix A, to determine which numerical simulation is well suited to achieve the above requirements. This resulted in the selection of the relatively novel lattice Boltzmann method (LBM). A more detailed overview of LBM is given in chapter 2.

LBM has a few advantages over conventional Navier-Stokes based solvers for multiphase flow, which lead to this method being chosen. In LBM the multiphase flow has a diffuse interface, and no grid refinements are needed to track the phase interface [64]. In addition, LBM is reported to be an order of magnitude faster for simulating multiphase flow, compared to conventional solvers [103]. This results in stable, and quick simulations which were two of the main issues reported previously [52].

LBM can capture the relevant physics, rarefaction and gas compressibility, which occur in microchannels [73, 88, 117]. Furthermore, droplet collision has been accurately simulated [91]. Specific studies in flow boiling in microchannels have been performed as well [44].

In addition, physics in relevant scales such as particle flow [141], and complex flow in porous media such as in fuel cells [104] have been demonstrated. Recently, LBM has been used for supersonic flow [29, 34–36, 110, 116, 123] which should allow simulations in the nozzle beyond the continuum assumptions.

Since the LBM method has not been taught in the aerospace master course, the viability of this project was discussed with Dr. Portela, who teaches multiphase flow in applied physics, and the then PhD candidate Siddhartha Mukherjee, who was simulating turbulent emulsion using LBM. They said that my problem of simulating boiling multiphase flow in microchannels should be possible using LBM.

For the thesis, the first part is to learn how to simulate channel flows using LBM. The LBM simulation framework OpenLB will be used to start with single phase isothermal single channel flow. More physics is added once the previous step is completed. This results in a steady build up of knowledge where any bugs can be more easily isolated, identified and removed.

## Thesis report structuring

This report continues with an overview of the simulation tool used. This overview is split into two chapters to avoid confusion; theory and implementation. In chapter 2, an overview of the lattice Boltzmann method theory essentials is given. In chapter 3, the simulation framework and extensions to the framework is detailed. That has the advantage that readers who are not interested in the code itself can pass over chapter 3 without loosing too much of the background. Whereas readers who want to use the code know exactly where to look.

With the simulation tool explained, it is then verified. The verification is performed in two steps: isothermal, chapter 4, and thermal, chapter 5. The isothermal verification ensures that the multiphase behaviour is as excepted. Then the thermal verification shows not only that the thermal behaviour is as expected, but also how including the thermal problem has additional effects on the multiphase behaviour.

The simulation tool is then applied to additional problems with an in depth discussion of recurring problems, given in chapter 6. The boiling behaviour of the tool is shown. The multichannel geometry generator and a plan for the validation of the simulation tool is presented.

The report is ended with a conclusion and recommendation in chapter 7. The conclusion reflects on what research question requirements were achieved. The recommendation summarizes how unsolved issues encountered can be approached.

# 2 Lattice Boltzmann Method - LBM

This chapter describes the modelling method used in the simulations. This starts of with an overview of the essentials used in relation to this thesis. The chapter is not a comprehensive overview of the lattice Boltzmann method and many details are missing. Rather, the aim is that people unfamiliar with the lattice Boltzmann method, but with a background knowledge of traditional fluid dynamics can follow along the remaining aspects of the thesis.

Since the lattice Boltzmann method (LBM) is not standard teaching material at the TUDelft aerospace master, this chapter starts of with a series of sections detailing the essentials. section 2.1 starts with a very short introduction in how LBM fits into the overall scheme of computational fluid solvers. This is followed by section 2.2 which details the physics and implementation of the basic LBM, which is isothermal, incompressible single phase flow. section 2.3 builds up from the basic LBM by including multiphase flow, and section 2.4 adds the thermal aspect.

What is not covered in this chapter is how all the theory is implemented into the simulation framework, which will be discussed in chapter 3. While the majority of the theory is present, such as the collision operators, forcing methods, the basic multiphase methods and a thermal method, not always are the provided implementations sufficient.

## 2.1. LBM introduction

The lattice Boltzmann method (LBM) is a numerical discretization of the Boltzmann equation used to simulate flow problems. LBM distinguishes itself from the Navier-Stokes equations due to its origin in kinetic theory. Historically, the Chapman-Enskog analysis is used to show that LBM recovers the Navier-Stokes equations [64, 74].

How LBM fits compares to the other simulation is shown well in fig. 2.1, where Euler refer to the Euler equations, RANS for Reynolds averaged Navier Stokes, LES for large eddy simulations, DNS for direct numerical simulation, DPD for dissipative particle dynamics, DSMC for direct simulation Monte Carlo, and MD for molecular dynamics. DSMC has been previously used at the TUDelft aerospace faculty for simulating rarefied flow in microchannels [139].

LBM was developed from the molecular dynamics solver lattice-gas automata around 1990 [6]. It gained traction due to the simplicity in implementing the method into code and computational efficiency.

This bottom up approach allows capturing of some more nuanced physical phenomena, which would require additional terms, or numerical exploits to simulate. It is especially useful in simulating complex geometry, such as porous media. Due to its simplicity for numerical implementation, LBM scales very well with the number of available precessing cores. LBM is now used in several industries which deal with microfluidics or macroscale porous problems; such as pharmaceutical (Lab-on-a-chip), biomedical research (clot forming & aneurysm), surface reactions, for example in fuel cells and battery research, and geoengineering (carbon dioxide sequestration in depleted gas reservoirs), and flow cooling research [8, 19, 20, 59, 91, 97, 138].

## 2.2. LBM fundamentals

This section will elaborate on the lattice Boltzmann method, giving the derivations and mathematical notations required to understand it. The LBM is a specific discretization of the Boltzmann equation. Thus first the BE will be introduced, and how it can recover fluid flow. Then the discretization process is covered. Finally,

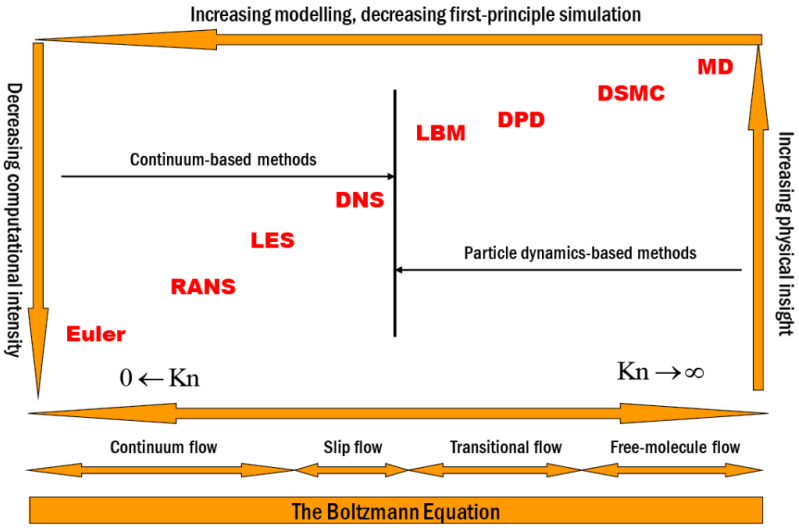



Figure 2.1: Modeling and simulation methods for multiscale phenomena [80, 91]

the section is concluded with the concepts needed to create a simple LBM simulation.

### 2.2.1. Boltzmann Equation

A small introduction to the Boltzmann equation (BE) was given in section 2.1, and is further explored here. The Boltzmann equation with forces, **F**, is shown in eq. (2.1).

$$\frac{\partial f}{\partial t} + \xi_\beta \frac{\partial f}{\partial x_\beta} + \frac{F_\beta}{\rho} \frac{\partial f}{\xi_\beta} = \Omega(f) \tag{2.1}$$

Where $t$ denotes time, $\mathbf{x}$ is the position, $\xi$ is the microscopic particle velocity, $\mathbf{F}$ external body forces, $\beta$ the index for dimension, $\rho$ the macroscopic density, $\Omega$ the collision operator, and $f$ is the particle distribution function.

In the BE, it is not the microscopic momentums that are tracked, but the particle distribution function $f$, i.e. given a point in time, $t$ and space, $\mathbf{x}$, how are the particles distributed along all the possible velocities, $\xi$. This particle distribution function advects itself, i.e. flows along its macroscopic velocity, which represents the second term on the right hand side of the Boltzmann equation.

For fluid dynamics, the biggest difference between the NSE and the BE is how the macroscopic momentum is represented [4]. In the NSE at a given location there is the macroscopic density $\rho$ and local velocity $\mathbf{u}$, and macroscopic momentum is the product of those two. In the BE at a given location, there is a distribution of density $f$ along all velocities $\xi$, i.e. the momentum is scattered. While this paragraph highlights the BE and NSE difference using momentum, it extends to both density and energy as well.

The macroscopic density $\rho$ is the sum of all scattered microscopic densities, i.e. the sum of all particles. This is performed by integrating over the microscopic velocity domain, i.e. $\int \mathrm{d}^3\xi$. For density only the sum of all particles is needed, thus the integral is performed over the particle distribution function $f$. This results in the equation shown in eq. (2.2). To recover the local macroscopic momentum, the sum of all local microscopic momentums is needed, shown in eq. (2.3). The microscopic momentum is the product between the distribution density $f$ and its microscopic velocity $\xi$. The macroscopic kinetic energy density can be found in the same manner eq. (2.4). In addition, the internal energy density can be calculated from the deviation of the mean velocity $\mathbf{u}$, see eq. (2.5). The internal energy due to internal degrees of freedom can be added [107, 126].

$$\rho(\mathbf{x},t) = \int f(\mathbf{x},\xi,t)\mathrm{d}^3\xi \tag{2.2}$$

$$\rho(\mathbf{x},t)\mathbf{u}(\mathbf{x},t) = \int \xi f(\mathbf{x},\xi,t)\mathrm{d}^3\xi \tag{2.3}$$

$$\rho(\mathbf{x},t)E(\mathbf{x},t) = \frac{1}{2}\int |\xi|^2 f(\mathbf{x},\xi,t)\mathrm{d}^3\xi \tag{2.4}$$

$$\rho(\mathbf{x},t)e(\mathbf{x},t) = \frac{1}{2}\int |\xi-\mathbf{u}|^2 f(\mathbf{x},\xi,t)\mathrm{d}^3\xi \tag{2.5}$$

The last key part of the BE is handling the molecular collision. It is possible to explicitly express the collision term using the molecular chaos assumption. This results in a sextuple integral over the twice the velocity domain and considers all possible two particle collisions. This approach is complicated and prohibitively computationally expensive to calculate.

The more computationally friendly route is to assume that in case of many collisions and over a long period of time, the local particle distribution approaches an equilibrium distribution, eq. (2.6), with the mean density and momentum equal to the macroscopic density and momentum, eqs. (2.7) to (2.8). This distribution is called the equilibrium distribution, designated by $f^{eq}$, and the origin of the statistical Maxwell-Boltzmann distribution.

$$f^{eq} = \rho\left(\frac{1}{2\pi RT}\right)^{3/2} \mathrm{e}^{-|\mathbf{u}|^2/(2RT)} \tag{2.6}$$

$$\rho(\mathbf{x},t) = \int f^{eq}(\mathbf{x},\xi,t)\mathrm{d}^3\xi \tag{2.7}$$

$$\rho(\mathbf{x},t)\mathbf{u}(\mathbf{x},t) = \int \xi f^{eq}(\mathbf{x},\xi,t)\mathrm{d}^3\xi \tag{2.8}$$

This tendency to relax to the local equilibrium distribution allows the complex collision integral to be expressed as a simplified collision operator, $\Omega(f)$. The simplest of collision operators is the Bhatnagar, Gross and Krook (BGK) collision operator, shown in eq. (2.9), where $\tau$ is called the relaxation time. From the microscopic point of view, this relaxation time represents the collision frequency. For the macroscopic world it characterizes the fluids transport property. For fluid flow that is momentum transport, i.e. viscosity; $\nu = \mathrm{f}(\tau)$, which is explained in section 2.2.5.

$$\Omega(f) = -\frac{1}{\tau}(f - f^{eq}) \tag{2.9}$$

Classically, the Chapman-Enskog analysis is used to show that the BE recovers the NSE [16, 74]. Where recovering means that the BE approximates but is not equal to the NSE. The Chapman-Enskog analysis is a special case of multi-scale analysis. It is based on the representation of the particle distribution function as the sum of the equilibrium particle distribution and a power expansion of the non-equilibrium particle distribution, in the order of the Knudsen number. The non-equilibrium particle distribution is the difference between the particle distribution and the equilibrium particle distribution.

The first two terms of the power expansion are needed to recover the NSE, and using only the first term recovers the Euler equations [64]. Using the third and fourth terms give the Burnett and super Burnett equations [109], which while theoretically applicable to rarefied flows, are unstable [132]. This throws some doubt on the validity of the Chapman-Enskog ansatz [74]. As such, other methods to recover the NSE from the BE have been performed, such as the Hermite expansion series by Grad [64, 126], explicit distribution perturbation, asymptotic expansion by Sone [64] to name a few, but the Chapman-Enskog seems to still be the most popular.

### 2.2.2. Discretization

The BE is continuous over the space, time and velocity domain. For the simulation, both the space, time and velocity domains need to be discretized. The theoretical framework with a good explanation is given by Shan et al. [126] and in the book *The Lattice Boltzmann Method* by Kruger et al. [64] . A regular grid is deployed to discretice space, a lattice for the velocity domain, and constant time steps for the time domain. The use of an irregular grid is possible, but they require additional interpolation, thus introducing a new error source and computational cost [21], and as such are not commonly used.

The discretization of the velocity domain utilizes the fact that to recover the macroscopic transport phenomena, not all microscopic events needs to be simulated. In fact, to recover the macroscopic properties, only the corresponding equilibrium distribution is required, eqs. (2.7) to (2.8)[64, 126].

The equilibrium distribution function $f^{eq}$ is a known exponential function. Such a function can be expressed using a series of physicists' Hermite polynomials, which in turn can be integrated exactly via the Gauss-Hermite quadrature [126]. The process is then applied to the particle distribution function. This process produces a set of velocity directions $\mathbf{c}$ and weights $\mathbf{w}$, with which macroscopic properties can be calculated from either distribution function.

The point where the Hermite series is truncated determines the degree of accuracy. To conserve mass, moment and energy, only the first three Hermite series terms are required[126]. Higher orders are reported to help accuracy and stability of the simulation, however they do also require a larger set of velocities.

The set of velocities must fulfil the conditions that it is rotationally isotropic and conserves mass, momentum and energy. It is however efficient to use a velocity set that points to other grid nodes to avoid interpolation [86], which introduces several errors beyond numerical interpolation errors [70]. For full computational efficiency, these nodes should be the nearest neighbour. However, the resulting nearest neighbour velocity set is not suitable to include thermal transport. Thus it is common to assume isothermal flow when discretizing the BE. This however is problematic, since for phase change, the isothermal assumption is not valid. In section 2.4, it is explained how the thermal problem is incorporated back into the LBM model.

An important parameter that is dependent on the lattice is the speed of sound, $c_s$. This parameter is introduced since for many lattices the velocity set contains the factor $1/\sqrt{3}$. As such, it is simpler and more efficient to present and calculate with a velocity set given by $\mathbf{c} = \xi / c_s$. When using LBM to propagate sound waves, the ideal wave equation recovered is $\nabla^2 p = \frac{1}{c_s^2}\partial_t^2 p$. Thus, the waves travel at the speed of $c_s$, i.e. the speed of sound.

Velocity sets are named in a DnQm fashion, where n indicates the dimension, and m the number of velocities. Common sets of velocities are: D1Q3, D2Q9, D3Q27, the first two shown in fig. 2.2. In some applications, the D3Q27 lattice can be pruned to D3Q15 and D3Q19, all three shown in fig. 2.3, trading off some accuracy / stability for computational efficiency. Individual velocity directions are given by the vector $\mathbf{c}_i$, where $i$ is the $i$th velocity direction in the set. As such $f_i$ is the particle distribution function in the $i$th velocity direction. $f_i$ is often shortened to population, i.e. the particle population. There is no standard method to define which velocity has what number, and as such care should be taken when using papers, although 0 is commonly taken as the rest position.

Lattices with higher order truncations are possible to recover the full thermal NSE. These lattices that extend beyond the nearest neighbour are called multispeed or higher-order lattices. An important lattice base is D1Q5-ZOT (zero, one, three) lattice and its higher dimensional counterparts of D2Q25 and D3Q125 [23, 35]. This highlights that for a full recovering of the NSE in 3D requires a tremendous amount of memory and processing. However with smart usages of various techniques, the computational cost can be decreased greatly. It is possible to prune the lattice to a D3Q41 lattice, while still recovering the NSE to an acceptable level. It is shown that with a pruned multispeed D2Q16 lattice, it is possible to simulate supersonic flow [29].

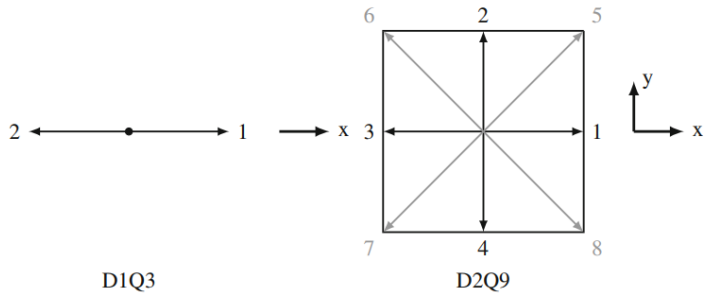



Figure 2.2: D1Q3 and D2Q9 velocity sets taken from [64]

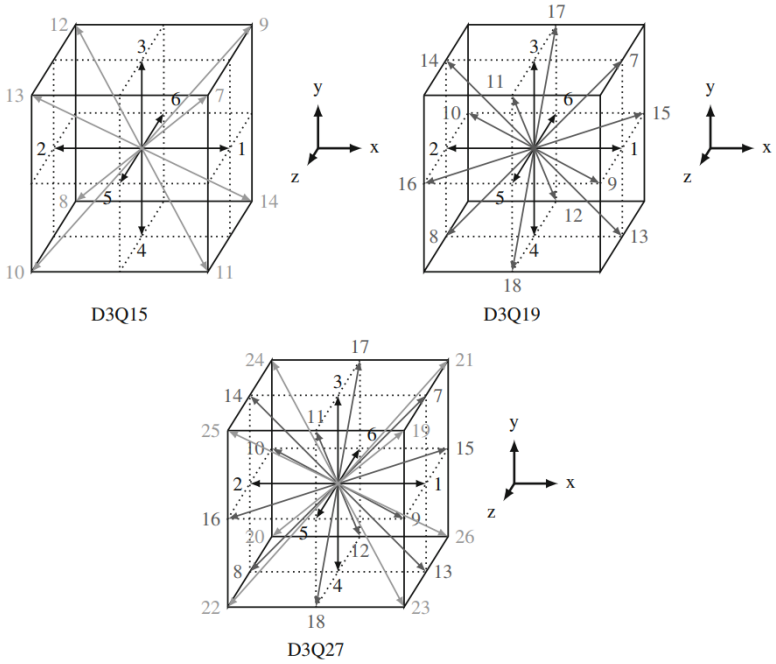



Figure 2.3: D3Q15, D3Q19 and D3Q27 velocity sets taken from [64]

### 2.2.3. Lattice Boltzmann equation

After discretization the Boltzmann equation, one can arrive at the lattice Boltzmann equations (LBE), eq. (2.10), where $\Delta t$ is the timestep, which is for the nondimentionalized simulations simply 1, and often dropped from the equations for clarity.

$$f_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) \tag{2.10}$$

The equilibrium particle distribution function $f_i^{eq}$ for the isothermal simplification is given in eq. (2.11), where $w_i$ is the weight corresponding to the lattice used and current velocity direction $\mathbf{c}_i$ . The macroscopic density $\rho$ and velocity $\mathbf{u}$, can be calculated from the local node via eqs. (2.12) to (2.13)

$$f_i^{eq} = w_i\rho\left(1+\frac{\mathbf{u}\cdot\mathbf{c}_i}{c_s^2}+\frac{(\mathbf{u}\cdot\mathbf{c}_i)^2}{2c_s^4}-\frac{\mathbf{u}\cdot\mathbf{u}}{2c_s^2}\right) \tag{2.11}$$

$$\rho(\mathbf{x}, t) = \sum_i f_i(\mathbf{x}, t) \tag{2.12}$$

$$\rho\mathbf{u}(\mathbf{x}, t) = \sum_i \mathbf{c}_i f_i(\mathbf{x}, t) \tag{2.13}$$

LBM simulations split the LBE into two parts, collision and streaming, [4], visualized in fig. 2.4. In the collision step, the right hand side of the LBE is calculated and stored. The calculated population in the collision step is usually noted via a superscript star. In the streaming step, the particle population at each node are updated from the stored post collision populations. Demonstrating those two steps with the BGK collision operator (resulting equation is often called lattice BGK or LBGK equation):

$$f_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) = f_i(\mathbf{x},t) - \frac{\Delta t}{\tau}\left(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)\right) \tag{2.14}$$

Collision step:

$$f_i^*(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{\Delta t}{\tau}\left(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)\right) \tag{2.15}$$

$$= f_i(\mathbf{x},t)\left(1-\frac{\Delta t}{\tau}\right) + f_i^{eq}(\mathbf{x},t)\frac{\Delta t}{\tau} \tag{2.16}$$

Streaming step:

$$f_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) = f_i^*(\mathbf{x},t) \tag{2.17}$$

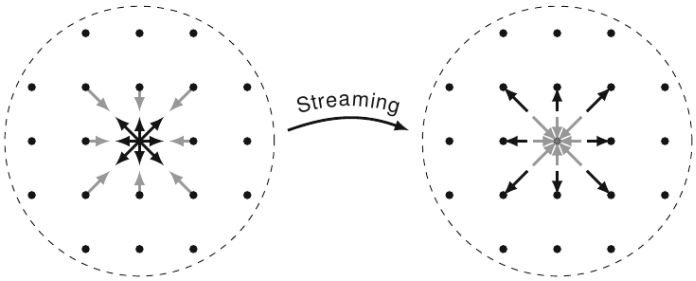



Figure 2.4: "Particles (black) streaming from the central node to its neighbours, from which particles (grey) are streamed back. To the left we see post-collision distributions $f_i^*$ *before* streaming, and to the right we see pre-collision distributions $f_i$ *after* streaming" [64]

The above two collision and streaming steps are repeated till the end of the simulations. The only interruptions are saving the current simulation state. As can be seen, the LBM simulation calculation steps are extremely straight forward. While the calculations are numerous, with up to 27 velocity directions in three dimensions for isothermal flow, they are very simple.

All the calculations are fully local, requiring only information available at the local node and not neighbouring nodes. This shows the strongest advantage of LBM, the scalability. LBM does not suffer greatly from splitting the simulation domain into many smaller domains, and rejoining the boundaries are greatly simplified. The simple computations and ease of scalability results that LBM can have a numerical edge over conventional NS solvers. However, LBM requires a much larger memory capacity, and is often limited not by processing speed, but by memory access speed. [8, 12, 97, 133]

### 2.2.4. Collision operators

The collision operator describes the tendency of the fluid to relax to its local equilibrium. As was mentioned in section 2.2.1, the simplest collision operator is the BGK operator, repeated below for completion. Since there is only one relaxation time, $\tau$, the BGK is also called the single relaxation time (SRT) operator.

$$\Omega_{BGK} = -\frac{1}{\tau}\left(f - f^{eq}\right)$$

While the BGK is very popular due to its simplicity, it carries many problems because of it. All fluid properties are linked to this one relaxation time, such as bulk and shear viscosity. And the relaxation time is interlinked to the grid refinement. Changing the grid spacing, or time step, changes the relaxation time and thus viscosity. This means that achieving the wanted viscosity may lead to very fine grid spacing and time steps, thus resulting in long simulations.

The BGK collision operator is not specific to the LBM, and does not require the BE to be discretised. However, any collision operator mentioned from here on, is specific to the LBM and requires discretization.

Opposed to the BGK / SRT collision operator is the multiple relaxation time (MRT) collision operator. The MRT collision operator relaxes each population $f_i$ with a different relaxation time, stored in a relaxation matrix $\Lambda$. The BGK model is a special case of the MRT where all the relaxation times are equal. The relaxation is most efficiently performed in the momentum space, thus an additional transformation matrix $\mathbf{M}$ is needed. The matrix multiplication of $\mathbf{M}\cdot\mathbf{f}$ results in a momentum vector containing the velocity moments, $\mathbf{m}$, up to the same order as the truncation order used in the discretization. This includes density (zeroth order), momentum (first order), momentum flux i.e. pressure (second order) and higher order momenta. This results in the MRT collision operator, eq. (2.18).

$$\Omega_{MRT} = -\mathbf{M^{-1}}\Lambda\mathbf{M}\left(\mathbf{f}-\mathbf{f}^{eq}\right) = -\mathbf{M^{-1}}\Lambda\left(\mathbf{m}-\mathbf{m}^{eq}\right) \tag{2.18}$$

The advantage is that the MRT operator tends to be more stable, and fluid properties can be independently adjusted for. Surprisingly, the MRT operator is only 15-20% more computationally expensive compared to the BGK operator [64]. The main drawback is the implementation complexity, since all the matrices are QxQ large. For a D3Q19 lattice, that results in two unique 19x19 matrices. In addition, not all values in these matrices have a macroscopic definition, yet still must be defined. These are the higher order momenta recovered from the BE which are not there in the NSE. While they do not impact the recovered NSE equation, they do influence the simulations, and optimal choices can only be found via numerical studies.

A simpler approach compared to the MRT, while allowing for some flexibility in the viscosity choice, is the two relaxation time (TRT) collision operator. The TRT is a special case of the MRT where one relaxation time $\tau^+$ relaxes even order momenta, and the second $\tau^-$ relaxes the odd order momenta [39, 40]. A numerical trick, using the symmetry of the lattice, is used to define symmetric $f^+$ and anti-symmetric $f^-$ populations and relaxing those, where the $f_{\bar{i}}$ is the population in the opposite velocity direction. The viscosity is purely a function of $\tau^+$, and $\tau^-$ can be gotten via the "magic" parameter $\Lambda_{TRT}$ [30].

$$\Omega_{TRT} = -\frac{1}{\tau^+}\left(f^+ - f^{eq+}\right) - \frac{1}{\tau^-}\left(f^- - f^{eq-}\right) \tag{2.19}$$

$$f_i = f_i^+ + f_i^- \qquad\qquad f_i^\pm = \frac{f_i \pm f_{\bar{i}}}{2} \tag{2.20}$$

$$\Lambda_{TRT} = \left(\tau^+ - \frac{1}{2}\right)\left(\tau^- - \frac{1}{2}\right) \tag{2.21}$$

With TRT, the viscosity can be independently chosen, aiding in the stability and accuracy of the simulation. Specific values of $\Lambda_{TRT}$ cancel out specific order of errors, thus allowing a simpler way to further achieve better simulation accuracy and stability.

The BGK and MRT collision operators are used for incorporating the thermal problem back into the standard LBM. Both the TRT and MRT were attempted to be used in combination with the multiphase LBM, but was unsuccessfully implemented. However, the main recommendation provided of the thesis is to implement the MRT collision operator for multiphase LBM, which is why it is provided in more detail.

In addition to the straight forward collision operators, there are others that encompass other theorems. A noteworthy extension is the entropic collision operator, which is based on the fact that entropy must always increase. Thus the collision operator is extended to include an additional term, based on the Boltzmann $\mathscr{H}$-Theorem, to ensure that entropy increases [10, 11, 155]. The resultant simulations tend to be more stable and entropic LBM is promising for high Mach number simulations.

### 2.2.5. Recovering macroscopic properties

In section 2.2.3, it was briefly explained how the macroscopic density $\rho$ and velocity $\mathbf{u}$ is recovered from LBM. This section continues from that and explains how other physical properties are recovered from the LBM.

The density and velocity are the more straightforward parameters that can be recovered and for convenience are reiterated here:

$$\rho(\mathbf{x},t) = \sum_i f_i(\mathbf{x},t) = \sum_i f_i^{eq}(\mathbf{x},t)$$
$$\rho\mathbf{u}(\mathbf{x},t) = \sum_i \mathbf{c}_i f_i(\mathbf{x},t) = \sum_i \mathbf{c}_i f_i^{eq}(\mathbf{x},t)$$

It can be shown, using the Chapman-Enskog expansion, that the LBM BGK simulation recovers the following equations eq. (2.22) [64], neglecting $O(\mathbf{u}^3)$ terms, using the Einstein's summations convention, where $\partial_t = \frac{\partial}{\partial t}$ and $\partial_\alpha \lambda(\mathbf{x}) = \frac{\partial \lambda(\mathbf{x})}{\partial x_\alpha} = \nabla\lambda(\mathbf{x})$

$$\begin{aligned} \partial_t \rho + \partial_\gamma(\rho u_\gamma) &= 0 \\ \partial_t(\rho u) + \partial_\beta(\rho u_\alpha u_\beta) &= -\nabla(\rho c_s^2) + \partial_\beta\left[\rho c_s^2\left(\tau - \frac{\Delta t}{2}\right)(\partial_\beta u + \partial_\alpha u_\beta)\right] \end{aligned} \tag{2.22}$$

From eq. (2.22), the definition of pressure can be seen, shown for clarity in eq. (2.23). Pressure is purely a function of density. An overly simplified explanation is that the equation of state is a function of density and temperature $p(\rho, T)$. Since the isothermal assumption is used, pressure becomes a function purely of density.

$$p = \rho c_s^2 \tag{2.23}$$

Another parallel that is possible to draw is that for an ideal gas, the following equation of state holds: $p_{ideal} = \rho R_{sp} T$. The speed of sound in a gas is defined as eq. (2.24), where the subscript $_s$ refers to a constant entropy, i.e. constant temperature. Then substituting the ideal gas law into eq. (2.24) results in eq. (2.25), which shows that $c^2_{ideal} = R_{sp}T$. That can be substituted back into the ideal gas law showing that $p = \rho c^2_{ideal}$.

$$c^2 = \left(\frac{\partial p}{\partial \rho}\right)_s \tag{2.24}$$
$$= \left(\frac{\partial p}{\partial \rho}\right)_s = \left(\frac{\partial(\rho R_{sp}T)}{\partial \rho}\right)_T = R_{sp}T \tag{2.25}$$

Using the same method as in pressure, the macroscopic viscosity is recovered, shown in eq. (2.26). For the BGK nearest neighbour LBM, viscosity is purely a function of the relaxation time and timestep. For non-dimensional simulations, the $\Delta t = 1$, thus making viscosity further limited. This may look bit counter intuitive, and thus a small discussion is given.

$$\begin{aligned} \nu_{shear} &= c_s^2\left(\tau - \frac{\Delta t}{2}\right) \\ \nu_{bulk} &= \frac{2}{3}\nu_{shear} \end{aligned} \tag{2.26}$$

The BGK LBM has a unique error behaviour which has no counterpart in conventional simulation methods. The relaxation time $\tau$ controls both the discretization error $Er \approx (\tau/\delta t - 1/2)^2$ and viscosity [64]. Thus simulations are very dependent on this parameter and can lead to non-unique solutions to the same flow problem. I.e., two simulations of the same flow problem but with different grid refinement will have two different $\tau$ but the same macroscopic viscosity. Since the $\tau$ is different, the simulations results will be slightly different, even though the same problem is simulated. Thus TRT and MRT collision operators, which can set viscosity independently of $\tau$ are recommended to be used.

In case of using the TRT collision operator, the parameters related to relaxation time, i.e. viscosity, are changed.

$$\nu = c_s^2 \left( \tau^+ - \frac{\Delta t}{2} \right) \tag{2.27}$$

The MRT collision operator allows even more flexibility, allowing setting different viscosities for shear and bulk.

$$\nu_{shear} = c_s^2 \left( \tau_\nu - \frac{1}{2} \right) \qquad \nu_{bulk} = c_s^2 \left( \tau_\zeta - \frac{1}{2} \right) - \frac{\nu_{shear}}{3} \tag{2.28}$$

This study uses the BGK collision operator due to its simplicity, and attempts were made to implement the MRT collision operator for multiphase flow without success. For the thermal problem, using the MRT vs the BGK did not provide additional accuracy, but helped in the stability of the simulation. One of the main recommendations is to implement the MRT collision operator for the multiphase LBM.

### 2.2.6. Implementation of forces

In this section, the method to implement forces into the LBM will be discussed. The inclusion of forces is not a simple, but a well researched topic. Forces can be included into the LBE via several different ways, all of which recover the NSE correctly, but deviate in higher order terms [64]. As such it usually matters little which forcing scheme is used, unless large forces or high velocities are involved.

First, the generic LBGK with forces will be given. Then the Guo, Shan-Chen and Kupershtokh forcing schemes that are relevant to this thesis are compared, highlighting how they are differing. Further discussion in how these forcing schemes handle multiphase flow is illustrated in more detail in section 2.3.

The generic LBM, second-order accurate in space and time, with the BGK collision operator, see section 2.2.4, is given in eq. (2.29) [51]. Adding the force also changes the actual fluid velocity, which is denoted as $\nu$. The term $F_i$ is the forcing term is defined in eq. (2.30). Guo et al. [51] showed that any forcing scheme can be characterised in such manner, even if originally it may look quite different. The parameters $A_0$, $\mathbf{B}$ and $\mathbf{C}$ are specific to the forcing scheme used. They must fulfil the condition of hydrodynamic consistency.

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) + F_i(\mathbf{x}, t) \tag{2.29}$$

$$F_i = w_i \left[ A_0 - \frac{\mathbf{B} \cdot \mathbf{c}_i}{c_s^2} + \frac{\mathbf{C} : (\mathbf{c}_i \mathbf{c}_i - c_s^2 \mathbf{I})}{2 c_s^4} \right] \tag{2.30}$$

To conserve mass, $A_0 = 0$, which is the case in all three of the schemes to be discussed. To recover the NSE, $\mathbf{B} = B_e \mathbf{F}$ and $\mathbf{C} = C_e(\mathbf{vF} + \mathbf{Fv})$[51]. However, different approaches to $B_e$, $C_e$, and $\mathbf{v}$ are used. The different values for those are shown in table 2.1.

Table 2.1: Comparison of forcing schemes in Guo generic formulation [77]

| Scheme | Velocity in $f_i^{eq}$ | Velocity in $F_i$ | Actual fluid velocity | $B_e, C_e$ |
|---|---|---|---|---|
| Shan-Chen | $\mathbf{u}$ | $\mathbf{u} + \frac{\tau \mathbf{F}}{2\rho}$ | $\mathbf{u} + \frac{\mathbf{F}}{2\rho}$ | 1 |
| Kupershtokh | $\mathbf{u}$ | $\mathbf{u} + \frac{\mathbf{F}}{2\rho}$ | $\mathbf{u} + \frac{\mathbf{F}}{2\rho}$ | 1 |
| Guo | $\mathbf{u} + \frac{\mathbf{F}}{2\rho}$ | $\mathbf{u} + \frac{\mathbf{F}}{2\rho}$ | $\mathbf{u} + \frac{\mathbf{F}}{2\rho}$ | $1 - \frac{1}{2\tau}$ |

Shan and Chen [125] have proposed in 1992 a very successful multiphase flow extension for the LBM, see section 2.3.1. However, the forcing schemes back then resulted in instabilities. Thus Shan and Chen proposed in the same paper a new forcing method, which is also applicable in standard LBM. In the original form, the forcing method was only a shift in the equilibrium velocity, $\mathbf{v} = \mathbf{u} + \tau \mathbf{F}/\rho$ , with no additional forcing term $F_i = 0$.

Kupershtokh's [68] method originated in kinetic theory, wanting to express the forcing term as a shift in the equilibrium populations. As such $F_i = f_i^{eq}(\rho, \mathbf{v}) - f_i^{eq}(\rho, \mathbf{u})$, where $\mathbf{v} = \mathbf{v} + \mathbf{F}/\rho$. This ensures that the forcing method behaves independent of $\tau$ used.

Guo et al.'s [51] approach was very methodical and is based on extensive analysis of the forcing schemes. Their scheme results in no unwanted forcing artefacts in the momenta due to the space and time discretization [64]. As such, their scheme has become quite popular.

### 2.2.7. Boundary handling

The boundary handling of numerical simulations is an important part to ensure that the physical problem domain is accurately represented. There are two types of boundaries: solid and open. The solid boundaries are walls, where no fluid is passing through. Most problems are not enclosed by walls and as such open boundaries, representing inlets and outlets, ensure the numerical domain closure.

Boundaries in LBM are not as straight forward as for conventional simulations. This is because the boundaries are described in the macroscopic viewpoint, such as no-slip at the walls, and then implemented in the mesoscopic method. This allows flexibility in how it is implemented, resulting in several methods acting differently in the mesoscopic model, but recovering the same macroscopic behaviour.

The book "The Lattice Boltzmann Method: Principles and Practice" by Krüger et al. [64] provides the most complete overview of the boundary method with good explanations and comparisons. Other books, like [98], skip over the boundaries or only lightly touch upon it. Original boundary papers assume a deep understanding of LBM and only provide selective comparisons.

#### Solid boundaries

The solid boundary behaviour greatly depends if the continuum assumption is used. If used, one can assume no slip flow, meaning that the tangential velocity to the wall is zero. If not, then a so called slip flow may be present, with a possible non zero slip flow.

Since the lattices used in the fluid simulations commonly have more than just the orthogonal velocity directions, they are well suited for handling complex geometry. Implementation and handling of edges and corners are especially tricky in 3D.

For the LBM, the simplest solid boundary implementation is the bounce back (BB) scheme, illustrated in fig. 2.5. For no slip flow, the incoming population is bounced back in the opposite direction. For slip flow, the incoming population is reflected similar to light being reflected onto a mirror.

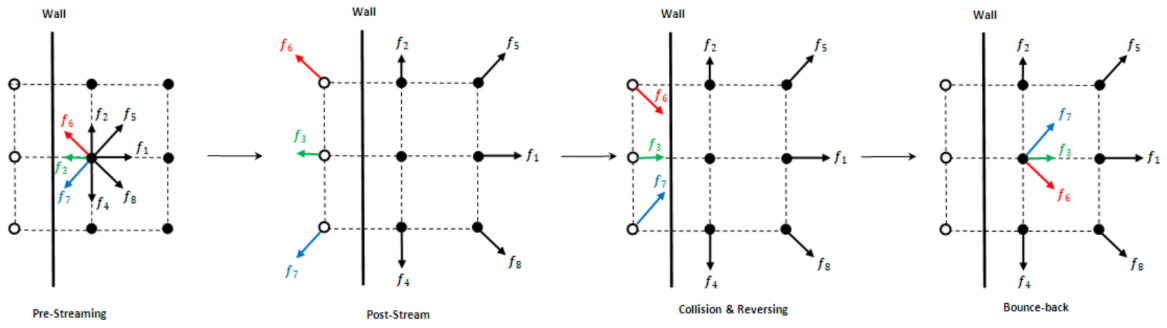



Figure 2.5: Illustration of mid-grid bounce-back taken from [4]

There are two numerical implementations for the BB scheme: fullway and halfway BB. Fullway BB means that the population is streamed to the wall node and the reflection of the population happens in the next time step, a total of two time steps. Halfway BB means that the population is reflected in the same time step. The fullway is simpler to implement while the halfway is better for transient simulations.

Both fullway and halfway BB assume that the wall is located halfway the node, and the node inside the wall is called a ghost node. If placed exactly halfway, the BB scheme is second order accurate. The location however is dependent on the relaxation time if the BGK collision operator is used. The TRT can be used to tune the wall location and eliminate certain errors while the MRT is the most accurate. The error of BGK BB is of the spatial second order and is visible as numerical slip.

To reduce the error, a new approach was taken resulting in wet-node methods. These methods assume that the wall lies very close to the node itself, resulting in a model shown in fig. 2.6. The simplest of wet-node methods is to assign an equilibrium population, based on the density, at the wall, resulting in the equilibrium scheme. This scheme has excellent stability but is only valid for trivial simulations. The simple scheme was extended to include a non-equilibrium part to allow non-trivial simulations. This part can be calculated via several different methods, but most commonly is derived by extrapolating from non-local nodes.

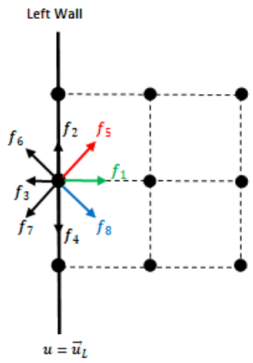



Figure 2.6: Illustration of on grid boundary conditions [4]

Zou and He introduced the non-equilibrium BB (NEBB) scheme (also called Zou-He scheme) [160], which incorporates principles from both the non-equilibrium extrapolation scheme and the BB scheme. The resultant scheme is third order accurate in space but is only viable to use in 2D simulations and may be unstable. Despite that, it is a popular choice for solid boundaries. Other third order accurate schemes include the Inamuro scheme and the Noble scheme, but these are not covered in depth here. The Bouzidi scheme is noteworthy for curved boundary handling.

When picking a boundary type, there is usually a trade-off between accuracy, stability, speed and complexity. The trade-off differs for 2D and 3D simulations, and some boundary conditions, while very accurate in momentum, do not conserve mass. Since the ideal boundary condition vary depending on the simulation, it is usually recommended to start with the BB and look into others if it is not sufficient [64].

### Open boundaries

The open boundary behaviour covers both inlets and outlets to the numerical simulation. Common open boundary conditions define a velocity profile or a constant pressure. Modifications to ensure proper handling of pressure waves, or convection boundaries which mimic the Neumann boundary exist as well but are not local.

The velocity boundary condition defines the velocity, while the density is variable. This is implemented using the BB approach as well, however with a non-zero velocity (for both tangential and normal). This is similar to a Dirichlet boundary condition for conventional simulations.

The pressure boundary condition defines the pressure, and for LBM in extension the density, while allowing the velocity to vary. The implementation is using a so called anti bounce back approach.

Specialized open boundary conditions, such as the convective boundary condition for advection diffusion problems based on finite differences can be used in combination with the above methods.

#### Periodic boundary

A special case is the periodic boundary condition which loops the simulation domain. This boundary is simple to implement and can be used to analyse cyclic phenomena. They also naturally conserve mass and momentum, and thus care should be taken when implementing forces to not add net momentum to the system. It is also possible to add a slight pressure increase to simulate a periodic pressure driven flow.

## 2.3. Multiphase LBM

This section introduces some methods that are available to introduce multiphase flow into the LBM. The multiphase flow method used in the simulations will be looked into in more detail subsequently in section 2.3.1. A good summary regarding multiphase flow and phase change flow was performed by Q.Li et al. [80].

Multiphase flows can be grouped by how many different substances are interacting with each other. For example, oil and water in their liquid state is an example of a multicomponent multiphase flow. Water evaporating into water vapour is a single component multiphase flow. In multicomponent flow, the transition between two components is usually unwanted, the contrary being true for single component problems.

Different methods have been developed to tackle these different problems, such as the free energy, the colour gradient, and the pseudopotential method. The free-energy method imposes macroscopic conditions onto the simulation [135, 136], while the colour gradient method introduces surface tension via an additional collision step [49, 118, 122]. However, both are unsuitable for high density ratio single component simulations (phase change problem), and are used more often for multicomponent with similar densities (water-oil mixtures) [2, 64, 125].

The pseudopotential method, often named the Shan-Chen (SC) method, was introduced in 1993 by X.Shan and H.Chen [125], and a 2014 review by L.Chen et al.[20] summarizes some advancements. The concept of this method is to include non-local interaction to incorporate non-ideal equation of states. The method is implemented by adding a local attraction force, which is dependent on the surrounding non-local densities. The different phases emerge due to this attraction by forcing surrounding densities together. The forcing term does not conserve local momentum, but does for the total momentum in the system. This method allows to incorporate any non-ideal equation of state, such as the van der Waal equation of state. The original method is often criticized to not be thermodynamically consistent, and only stable at high temperature ratios, and thus low density ratios. But these issues have been alleviated recently, making it a popular multiphase flow method. This method is applicable for both single and multicomponent fluid flows.

In a comparative analysis [103], it was found that the LBM can simulate multiphase flow an order of magnitude faster than a comparative NS volume of fluid solver.

The aforementioned entropic lattice Boltzmann method, section 2.2.4, can be incorporated into the free energy multiphase flow, greatly aiding the stability and thus extending the simulation range for multiphase problems [95, 96].

### 2.3.1. Pseudopotential method

In the previous section, section 2.3, the core concept of the pseudopotential method is introduced. This section discusses the method in more detail, presenting the forcing term and the extensions to enhance the original method to be thermodynamically consistent and viable for larger temperature ratios.

For a better understanding, a visual process of the phase separation is given in figs. 2.7 to 2.12. A water liquid/vapour mixture with average density of 1.3 with random fluctuations is initiated in a cube with periodic boundaries, initial state is shown in fig. 2.7. The non-dimensionalized densities are 8.2 for liquid and 0.04 for vapour. The phases separate, via spinodal decomposition [53], and water droplets emerge figs. 2.8 to 2.9. Then the droplets coalesce figs. 2.10 to 2.11 to form the lowest energy state of the system , which is either a single spherical droplet fig. 2.12, or a sheet of water depending on the average density available [53].

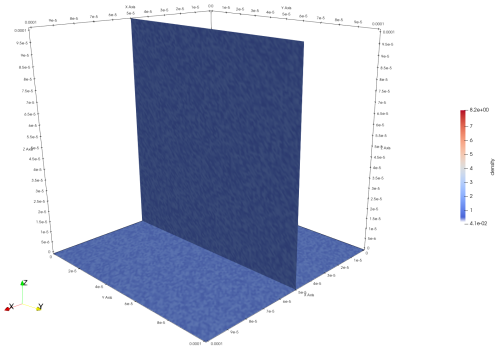


Figure 2.7: Pseudopotential phase separation process. Showcasing the initial conditions with random density fluctuations. $\Delta t = 0$

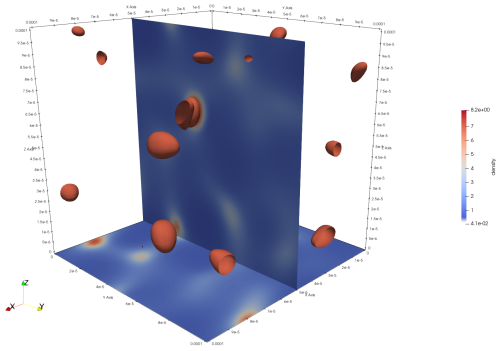


Figure 2.8: Pseudopotential phase separation process. Droplets start to form by attracting density from the local surroundings. $\Delta t = 600$

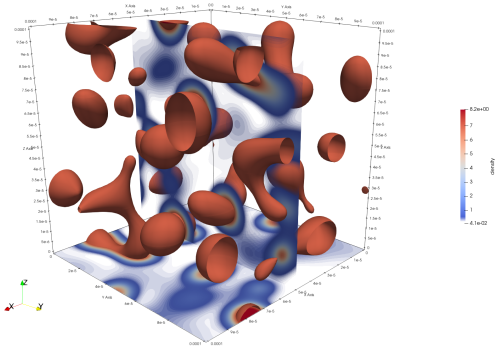


Figure 2.9: Pseudopotential phase separation process. Formed droplets continue to grow till surrounding density is at vapour density shown as transparent. $\Delta t = 800$

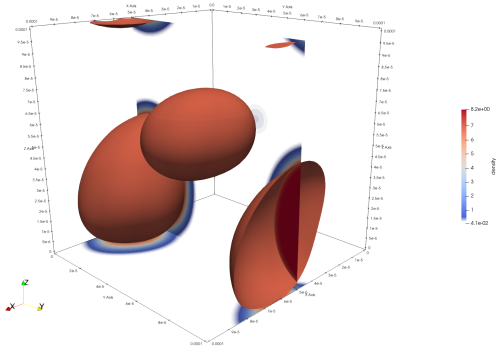


Figure 2.10: Pseudopotential phase separation process. Droplets coalesce into bigger droplets, here only two remain. $\Delta t = 3800$

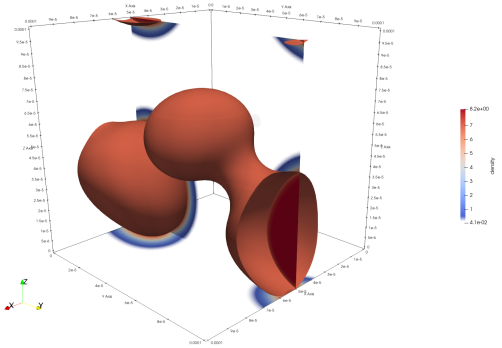


Figure 2.11: Pseudopotential phase separation process. Droplets coalesce into bigger droplets, an example coalescence is shown. $\Delta t = 4000$

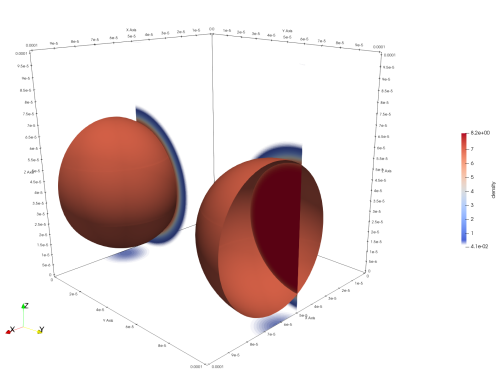


Figure 2.12: Pseudopotential phase separation process. The final state with one spherical droplet remaining. $\Delta t = 40000$

The popular implementation of the pseudopotential is its single component nearest neighbour form. The non dimensionalized pseudopotential forcing term at location $\mathbf{x}$ is expressed as shown in eq. (2.31).

$$\mathbf{F}(\mathbf{x}) = -\psi(\mathbf{x}) G \sum_i w_i \psi(\mathbf{x} + \mathbf{c}_i)\mathbf{c}_i \tag{2.31}$$

Where $\psi$ is the pseudopotential function, $G$ the interaction strength, $w_i$ the weight of the velocity direction and $\mathbf{c}_i$ the velocity direction vector.

The pseudopotential function describes the effective density used in the force calculation. It is purely based on the local density. Possible functions include eqs. (2.32) to (2.34), where the first one only works for multi-

component pseudopotential which was not introduced. The other two are both bounded such that the force lessens the closer it is to the reference density / pseudopotential.

$$\psi(\rho) = \rho \tag{2.32}$$

$$\psi(\rho) = \sqrt{\rho_0} \cdot \left(1 - e^{-\frac{\rho}{\rho_0}}\right) \tag{2.33}$$

$$\psi(\rho) = \psi_0 \cdot e^{-\frac{\psi_0}{\rho}} \tag{2.34}$$

Recovering the impact of the pseudopotential forcing onto the macroscopic continuum flow shows two impacts; emergence of surface tension and non-ideal equation of state. [124]

The pseudopotential force also introduces macroscopic surface tension, shown in eq. (2.35), which compared to the diffuse interface surface tension force $\mathbf{F} = \kappa\rho\nabla\delta\rho$ shows several parallels. Instead of density, the pseudopotential defines the surface tension and the parameter $G$ determines the strength.

$$\mathbf{F} = -\frac{c_s^4 G}{2}\psi\nabla\delta\psi \tag{2.35}$$

Including this forcing term into the simulation also changes the recovered macroscopic equation of state. Instead of $p = \rho c_s^2$, as is shown in section 2.2.5, the recovered macroscopic equation of state is shown in eq. (2.36). With the additional term the equation of state is no longer ideal, allowing for two phases to coexist.

$$p = \rho c_s^2 + \frac{c_s^2 G}{2}\psi^2(\rho) \tag{2.36}$$

The pseudopotential method has been used to simulate droplet merger , droplet impact on straight [115] and curved surfaces [26], droplet splashing [78]. Due to its simplicity and mesoscopic origin, it has become quite popular.

Higher order lattices for the pseudopotential multiphase flow exist, which allows incorporation of more accurate forcing representations, and lower spurious currents (velocity flows that occur due to simulating curved surfaces on a regular grid, see section 4.2.2 for more detail) [37, 99, 124]. The pseudopotential method can also be used for multicomponent, and allows for simulating complex multicomponent flow such as emulsion [102].

The pseudopotential does have a few flaws. In its original form, the resultant phases have a low density ratio. If any other than a specific equation of state is used, the simulation is also thermodynamically inconsistent. Many of those can be solved by using a realistic equation of state section 2.3.2 in combination with a modified forcing scheme section 2.3.4.

A flaw which is not addressed is that the pseudopotential for nearest neighbour is purely attractive, and as such all liquid will eventually coalesce into one blob. Including mid-range interactions can be used to include a repulsive term and allow for droplets coalescence avoidance [22, 33, 112, 121].

### 2.3.2. Equation of state

As mentioned in section 2.3.1, the pseudopotential method requires an equation of state to work. For the incompressible isothermal LBM, there exists only a single equation of state (EoS) that is thermodynamically consistent, eq. (2.34) with $\psi_0 = 1$ [77]. However, the resultant densities and simulations are not practical.

It is possible to rearrange the non-ideal equation of state to get a definition for $\psi$ as a function of pressure, eq. (2.37). The pressure can then be defined by an arbitrary equation of state, such as the van der Waals equation of state. However, the resultant simulations are not always thermodynamically consistent.

$$\psi(\rho) = \sqrt{\frac{2(P_{EoS}(\rho) - \rho c_s^2)}{c_s^2 G}} \tag{2.37}$$

Yuan and Schaefer [150] incorporated several equations of states and compared them. The EoS investigated by them are the van der Waals, Peng-Robinson, Redlich-Kwong, Redlich-Kwong-Soave and Carnahan-Starling. For validation, they compared their simulations to saturated water properties, and concluded that water is best represented by the Peng-Robinson (PR) equation of state, shown in eq. (2.38)[42, 150]. The PR equation of state is well known for its applicability for dipole molecule due to their inclusion of the acentric factor [89, 159].

$$p = \frac{\rho RT}{1 - b\rho} - \frac{a\rho^2 \alpha(\omega_{EoS})}{1 + 2b\rho - b^2\rho^2} \tag{2.38}$$

$$\alpha(\omega_{EoS}) = \left[1 + \left(0.37464 + 1.54226\omega_{EoS} - 0.26992\omega_{EoS}^2\right)\left(1 - \sqrt{\frac{T}{T_c}}\right)\right]^2 \tag{2.39}$$

Yuan and Schaefer [150] used values of $a = \frac{2}{49}$, b=$\frac{2}{21}$, and $R = 1$ for their comparisons of the cubic EoS. Since the majority of literature is based on their paper, these values have become quite prevalent. A note is that the parameter $a$ is adjusted to change the interface width, but is usually a multiple of $\frac{1}{49}$. The acentric factor that they used is $\omega_{EoS} = 0.344$

For water the critical point is $p_c = 220.64$ bar, $T_c = 647. \pm 2$ K and $\rho_c = 322$ kg/m$^3$, where when needed the acentricity $\omega_{EoS} = 0.3443$ is used [106].

### 2.3.3. Maxwell construction

The Maxwell construction can be used to calculate the liquid and vapour densities predicted by an equation of state. These predicted densities are used to verify that the simulation is correctly modelling multiphase flow. This method is also called the Maxwell equal area rule, or just equal area method, but in this thesis it is referred to as Maxwell construction.

The non-ideal equations of state in section 2.3.2 all have a physical deficiency. In a thermal equilibrium, pressure should not increase as volume increases. Any isothermal curve plotted for pressure - volume, below the critical temperature, has a segment where the curve's pressure increases as the volume increases, as can be seen in fig. 2.13. This is of course an aphysical phenomena, going against the statement of a fluid in thermal equilibrium. Theoretically, this segment describes a fluid whose liquid and gas phases are inseparable and transition smoothly, i.e. a supercritical fluid below the critical point.

To remedy this physical deficiency in the equation of state, a horizontal line is introduced into the plot, as shown in fig. 2.14. This line represents real fluid behaviour to separate into liquid and vapour at a specific pressure. This line intersects the isotherm curve three times, creating two areas. The left most intersection gives the liquid density, the rightmost the vapour density. The Maxwell construction postulates that the two areas have to be equal. In mathematical terms, this is expressed as the integral from $v_{sp,l}$ to $v_{sp,v}$ of the equation of state [5, 20, 64].

$$\int_{v_{sp,v}}^{v_{sp,l}} P_{EoS} \quad \mathrm{d}v_{sp} = P_s(v_{sp,v} - v_{sp,l}) \tag{2.40}$$

$$\int_{v_{sp,v}}^{v_{sp,l}} P_{EoS} - P_s \quad \mathrm{d}v_{sp} = 0 \tag{2.41}$$

The fact that this line exists and that the two areas are equal can also be explained via the concept of Gibbs potential: "The abrupt transition from vapor to liquid during a process of isothermal compression can there-

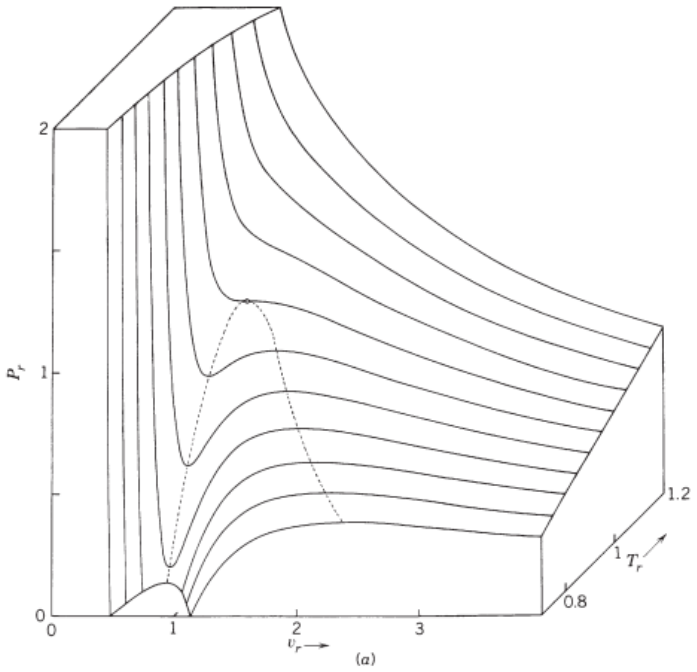



Figure 2.13: Three-dimensional surface corresponding to the van der Waals equation of state taken from [5]

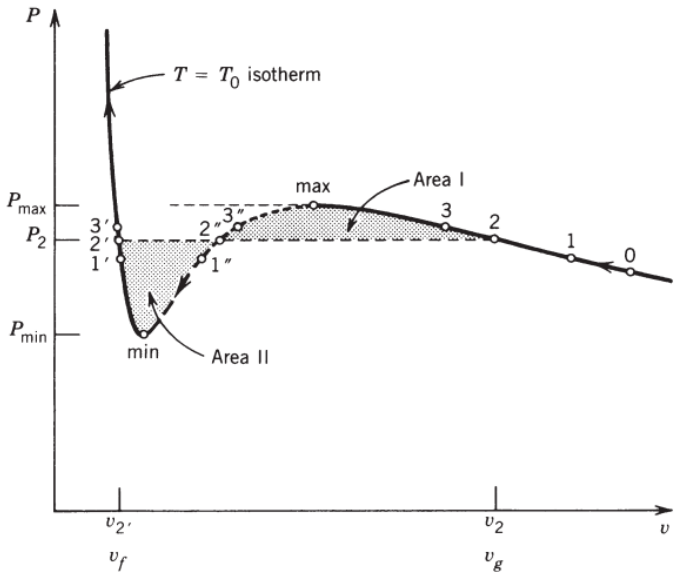



Figure 2.14: Maxwell construction for determining the pressure of the gas–liquid transition taken from [5]

fore be rationalized by invoking the principle of minimum Gibbs free energy at constant temperature and pressure." [5]

### 2.3.4. Pseudopotential forcing scheme

In section 2.3.1, the pseudopotential multiphase flow method is presented. In the conclusion it was stated that in its original form, there are a few issues remaining. The inclusion of a realistic equation of state is given

in detail in section 2.3.2, however, the thermodynamic inconsistency issue remains unanswered. The issue is presented here in more detail and two possible solutions are presented.

The thermodynamic inconsistency problem of the pseudopotential is a major point to criticize the method. How inconsistent the simulation is, is also dependent on how the force is implemented and what the relaxation time is. Where different combinations of relaxation time and force schemes resulted in better or worse outcomes [134]. It is noteworthy that the theoretically most accurate forcing scheme, the Guo forcing scheme, performs worst [113].

There are a few methods to achieve the thermodynamic consistency with a realistic EoS, however two are mentioned here: the beta scheme and Li's modified Guo velocity scheme. They are chosen due to the popularity of the methods.

The beta scheme [1, 42], introduced in 2011 by Gong and Cheng, modifies the original forcing term of the pseudopotential to eq. (2.42). Choosing an appropriate $\beta$ allows to approximate thermodynamic consistency and reduce spurious currents. When $\beta = 1$ the scheme reduces to the original pseudopotential forcing scheme.

$$\mathbf{F} = -\beta \left[ \psi(\mathbf{x}) G \sum_i \psi(\mathbf{x}+\mathbf{c}_i)\mathbf{c}_i \right] + \frac{1-\beta}{2} \left[ G \sum_i \psi(\mathbf{x}+\mathbf{c}_i)^2 \mathbf{c}_i \right] \tag{2.42}$$

In the next year, Li et al. [77] performed a higher order forcing scheme analysis and noted that the SC and EDM forcing schemes outperforms the Guo forcing scheme, since the introduced higher order errors cancel out the higher order errors of the pseudopotential force. With that knowledge they proposed a modified Guo forcing scheme, adding a correction term to cancel out the pseudopotential error term. The original pseudopotential force is kept the same, unlike in the beta scheme. Li and Lou later expanded their scheme for usage with MRT for D2Q9 [78], for D3Q15 by Xu et al. [148] and for D3Q19 by Zhang et al. [153].

The proposed modification to the Guo forcing scheme, detailed in section 2.2.6, is to add a velocity shift dependent on the local force and pseudopotential. The BGK version is shown in eq. (2.43), where the modified velocity term is given by eq. (2.44). The term $\nu$ is the kinematic viscosity, which they define as eq. (2.45). Compared to the general BGK viscosity (eq. (2.26): $\nu = c_s^2(\tau - 0.5)$ ) , the $c_s^2$ term is missing. That may however be a printing error, since reproduction of their results resulted in a different value used, which is detailed in section 3.3.

$$F_i = w_i \left(1 - \frac{1}{2\tau}\right) \left[ \frac{\mathbf{c}_i - \mathbf{v}'}{c_s^2} + \frac{\mathbf{c}_i \cdot \mathbf{v}'}{c_s^4} \mathbf{c}_i \right] \mathbf{F} \tag{2.43}$$

$$\mathbf{v}' = \mathbf{v} + \frac{\sigma \mathbf{F}}{\nu \psi^2} \tag{2.44}$$

$$\nu = \tau - 0.5 \tag{2.45}$$

Comparisons of Li et al's method with the three base methods and with a modified Kupershtokh method can be found in a paper by A.Hu, L.Li and R.Uddin [56]. They concluded that Li et al's method is an improvement, slightly better than the modified Kupershtokh and much better than the base three when comparing thermodynamic consistency. They also mentioned that "the numerical errors and effect of $\tau$ are still noticeable when the temperatures are relatively low, especially when interfaces are curved" [56], where the lowest temperature ratio tested was 0.6.

### 2.3.5. Surface wettability

The surface wettability is usually characterized by the contact angle a droplet makes on the surface. To model this interaction, the fluid next to the wall experiences an additional force $F_s$. Various approaches have been taken to modelling this force. They are all similar to the definition of the pseudopotential method, and make use of a switching function $s(\mathbf{x})$. This switching function is 1 for walls, and 0 otherwise. A good summary of

the various methods are given by Li et al. [80], and are paraphrased here. The contact angle can be calculated from the droplet height and the wetting length as is explained in [31].

The first method gives the fluid a force dependent on the parameter $G_w$ and local density $\rho$ as shown in eq. (2.46) [96]. The parameter $G_w$ can be used to adjust the contact angle. This method is used by Wi et al. [146] to simulate the microchannel nucleation using surface patterns with different wettability.

$$\mathbf{F_s}(\mathbf{x}) = G_w \rho(\mathbf{x}) \sum_i w_i s(\mathbf{x}+\mathbf{c}_i)\mathbf{c}_i \tag{2.46}$$

It is also possible to give the wall a fictitious density, and model the wall as if it were a fluid [7, 130]. The resultant equation, shown in eq. (2.47), where $G$ is the same as the pseudopotential. Since this method is easy to implement, it is also widely used [20, 26]. A snapshot of a droplet for various different wall densities are given in fig. 2.15, see section 2.3.5.

$$\mathbf{F_s}(\mathbf{x}) = G\psi(\mathbf{x}) \sum_i w_i \psi(\mathbf{x}+\mathbf{c}_i)\mathbf{c}_i \tag{2.47}$$

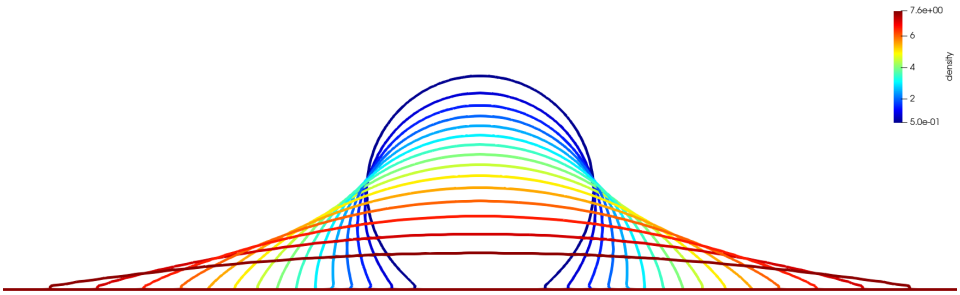



Figure 2.15: Showcasing the droplet contours at various surface wettability levels.

Li et al. [79, 80] performed an analysis comparing the previous two implementations and found that a combination of the two performs best, where they swap between the two depending on whether the surface is more hydrophobic or more hydrophilic. Extensions for the methods exist for correct modelling of a moving droplet tail, and multirange approaches [115].

### 2.3.6. Gravity

There are three main ways to implement gravity, shown in eqs. (2.48) to (2.50), where $\mathbf{g}$ is the gravitational vector [152]. The method described in eq. (2.48) directly implements gravity affecting every single fluid node proportional to the local density.

$$\mathbf{F}_g(\mathbf{x}) = \rho(\mathbf{x})\mathbf{g} \tag{2.48}$$

The magnitude of the force can be reduced by only modelling the buoyancy eq. (2.49). This can be interpreted that only fluid nodes lighter than the liquid experiences a force. A complementarity implementation is using vapour density $\rho_v$, thus applying force to liquid nodes only.

$$\mathbf{F}_g(\mathbf{x}) = \big(\rho(\mathbf{x}) - \rho_l\big)\mathbf{g} \tag{2.49}$$

The last method eq. (2.50) describes the gravitational force without adding a net momentum to the simulated system. This is useful for simulations with periodic boundaries, such as for infinitely falling droplets, since no net energy is introduced into the system through gravity.

$$\mathbf{F}_g(\mathbf{x}) = \big(\rho(\mathbf{x}) - \rho_{avg}\big)\mathbf{g} \tag{2.50}$$

## 2.4. Thermal LBM

Up to this point, the LBM methods discussed use the isothermal assumption. This section starts with giving an overview of the different methods for implementing temperature into the LBM: multispeed, double distribution and hybrid. It continues with discussing the used method in more detail, section 2.4.1, including how the boundary conditions differ from the conventional LBM. The section concludes with the topic of multiphase thermal simulations including phase change.

#### Multispeed lattice method

In section 2.2.2, it is mentioned that the nearest neighbour velocity set is not suitable to include thermal transport. This implies correctly that a higher order velocity sets can be used to simulate thermal transport [107, 111]. However, since higher order velocity sets are more complex to implement efficiently into simulation frameworks, it is not commonly used [93].

#### Double distribution function method

From a transport phenomena point of view, the LBM effectively solves the momentum advection diffusion problem of a fluid flow, where advection-diffusion is shown in fig. 2.16. Replacing momentum with temperature and LBM can be used to solve thermal advection-diffusion problems. This results in two separate problems, each solved on its own lattice. The advection of the thermal problem is coupled with the velocity of the fluid. This method is called the double distribution function (DDF) [64, 127, 152].

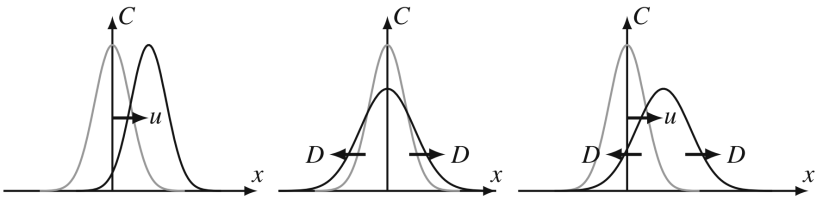



Figure 2.16: "Illustration of pure advection (*left*), pure diffusion (*middle*) and advection-diffusion (*right*). $u$ and $D$ are the advection velocity and diffusion coefficients, respectively. The *grey curve* is and initial concentration distribution C(x,t=0), the *black curve* shows the concentration at a later time" [64]

#### Hybrid method

It is possible to utilize the regular grid of the LBM to simulate the temperature advection diffusion problem by using a classical finite difference (FD) or volume of fluid (VoF) method. This results in a hybrid thermal LBM [47, 79, 81]. The advantage is that the underlying physics are well understood, relying on the long history these methods in numerical simulations. The disadvantage is that the temperature is simulated using macroscopic methods, and thus loose the computational advantages of the LBM solver (computational speed and ease for parallelizing).

### 2.4.1. Double distribution function

The double distribution function is solving the advection diffusion (AD) problem of a concentration. For thermal AD, the concentration $C$ can be temperature, internal or total energy. Viscous heating and compression work is usually neglected or added as a source term to the dynamics [64]. The resultant equation is shown in eq. (2.51), where $g_i$ is the particle distribution function for the secondary lattice and $Q$ representing the source term eq. (2.53), where $q$ is the macroscopic source. The recovered macroscopic thermal equation is given eq. (2.55) [64].

The thermal diffusion, $D$, is dependent on the relaxation time used eq. (2.54). An example for a thermal LBM with different thermal diffusivity is presented by [143]. However, due to both thermal and fluid lattice sharing the same spatial and time steps, the diffusion between the fluid and thermal can only deviate by so much.

To enable simulations that work for both diffusivities, the lowest diffusivity is used with the lowest possible relaxation time $\tau$. Then the other relaxation time is calculated with the set grid refinement. An extreme case example may be, that for the thermal problem $\tau \approx 0.6$ is used, which results for the fluid $\tau \approx 10$. This would work, but the computational time would be very much excessive. TRT and MRT collision operators alleviate this problem somewhat.

A workaround is to use the source term eq. (2.53) to remove the LBM AD and add a calculated source term based on finite difference, resulting in a semi-hybrid thermal method [57, 82]. This however is computationally inefficient and thus it is recommended that a different method is used.

$$g_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = g_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) + Q_i \tag{2.51}$$

$$\sum_i g_i = \sum_i g_i^{eq} = C \tag{2.52}$$

$$Q_i = \left(1 - \frac{1}{2\tau_g}\right) w_i q \tag{2.53}$$

$$D = c_s^2 \left(\tau_g - \frac{1}{2}\right) \tag{2.54}$$

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\mathbf{u}) = \nabla \cdot (D \nabla C) + q \tag{2.55}$$

When using internal or total energy as the simulated concentration, an additional error arise from the force affecting the local density which is present in both the fluid and thermal lattice [81]. This forcing error can be neglected for small forces, but at multiphase interfaces, this force becomes large and as such the error term needs to be corrected. This term is only needed when internal or total energy is used, not for temperature, as was done by [152].

Since the temperature AD problem only has one conserved quantity (temperature) and not two as in the fluid (density, velocity), it is possible to prune the used lattice further to a D2Q5 and D3Q7 [64]. Using MRT dynamics with a pruned lattice results in an accuracy comparable to a D2Q9 or D3Q19 with BGK dynamics, while being computationally less expensive [63, 75, 87]. While for fluid flow, the second order truncation of the equilibrium distribution function is used, it is sometimes beneficial to use the first order truncation, shown eq. (2.56) where $C$ is defined by eq. (2.52) and $\mathbf{u}$ is defined by the fluid lattice, to eliminate certain numerical errors for the AD lattice [64, 82].

$$g_i^{eq} = w_i C \left(1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2}\right) \tag{2.56}$$

Since the DDF is used for the thermal flow, the boundary conditions for the fluid flows section 2.2.7 can be applied to the thermal flows. An adiabatic wall can be represented by the bounce back method. A constant temperature wall is implemented using a constant pressure wall. A heatflux can be modelled by adding a local source term to a pressure wall, or using a Inamuro boundary condition. [64]

### 2.4.2. Phase change LBM

For phase change LBM using the DDF approach, there are two parts that need to be included. The phase change in the fluid lattice and the latent heat in the thermal lattice.

First is the phase change in the fluid lattice, which for the potential method is quite straight forward. During the computation of the pseudopotential, one of the input to the EoS is the temperature. A higher temperature lowers the potential, and by extension the force, in the liquid and increases the potential for the vapour. This results in any liquid near the interface to evaporate and increase the vapour density, resulting in evaporation. To ensure that phase change does not happen, a constant temperature for the pseudopotential calculation must be kept [57].

When not using the pseudopotential method, another possible method is defining the evaporative massflow rate. This rate specifies at interfaces how much liquid is evaporating to vapour. The massflow rate can be determined from Cahn-Hilliard continuity equation [32, 119, 120].

Secondly is the incorporation of the latent heat during phase change, where the macroscopic phase change temperature equation is given in eq. (2.57), where $q$ represents the energy change due to phase change. The DDF energy equation contains the source term $Q_i$, which has to ensure the local energy balance is kept. At minimum, this term contains the latent heat, but can be extended to include other corrections, such as the impact of a changing thermal conductivity in the phase interface. Two approaches are presented below.

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla C = \frac{\nabla \cdot (\lambda \nabla C)}{\rho c_v} + q \tag{2.57}$$

The first approach starts from the macroscopic energy balance equation for a general multicomponent multiphase flow and incorporates the latent heat directly, shown in eq. (2.58) where $h_{lv}$ is the latent heat, $c_p$ the specific heat at constant pressure and $\phi$ a marker function which is zero for vapour and one for liquid [152]. This is performed by Kamali et al. [57] and continued by Zarghami et al. [152].

$$q = -\frac{h_{lv}}{c_p} \frac{d\phi(\rho)}{d\rho} \frac{d\rho}{dt} \tag{2.58}$$

$$\phi(\rho) = (\rho - \rho_{vap})/(\rho_{liq} - \rho_{vap}) \tag{2.59}$$

The approach by Zarghami was used originally in this thesis, but resultant simulations showed aphysical behaviour. Using the model as presented resulted in heat always flowing from the vapour phase into the liquid phase. A more detailed analysis of their model used showed that Zarghami et al. included the force correction term of Li et al. [81] for energy based DDF into their temperature based DDF. This is however not needed as is shown in section 2.4.3.

Since the method given by Zarghami et al. is faulty, a different method is used in the thesis. This second approach derives a thermal source term from the thermodynamic local entropy balance equation, neglecting viscous heat dissipation, using the equation of state from the pseudopotential. This method was first given by Házi and Márkus [53], which they rewrote for computational efficiency in [93], shown in eq. (2.60).

$$q = -\frac{T}{\rho c_v} \left( \frac{\partial P_{EoS}}{\partial T} \right)_\rho \nabla \cdot \mathbf{u} \tag{2.60}$$

To correct for LBM solving $\nabla \cdot (T\mathbf{u})$ instead of $\mathbf{u} \cdot \nabla T$, the source term often includes the term $T \nabla \cdot \mathbf{u}$ and are combined as shown in eq. (2.61).

$$q = T \left[ 1 - \frac{T}{\rho c_v} \left( \frac{\partial P_{EoS}}{\partial T} \right)_\rho \right] \nabla \cdot \mathbf{u} \tag{2.61}$$

Gong and Cheng [41–43] used this thermodynamic entropy based source term and studied nucleate boiling using the DDF thermal method. This method is also used to simulate microchannel nucleation by Wi et al.[146]. In 2015, Q.Li et al. [79] showed that it is possible to simulate the whole spectrum of the boiling curve with LBM using the hybrid thermal approach. In 2017, Q.Li et al. [82], improved Gong and Cheng's thermal LBM to include variable thermal conductivity via the semi-hybrid approach and eliminate certain errors. In 2018, W.Gong et al. [45] showed an alternative heat equation to be solved using a hybrid LBM simulation. In 2020, Mukherjee at al. [101] used this method to simulate bubble departure in microchannels.

W.Gong et al. [45] said that "the term $\frac{T}{\rho c_v} \left( \frac{\partial P_{EOS}}{\partial T} \right)_\rho \nabla \cdot \mathbf{v}$ existing in the recovered macroscopic energy equation in most of the normally used pseudopotential thermal LB models for liquid-vapour phase change, is paradoxical because the equations of state for both ideal gas and non-ideal gas are adopted simultaneously in the derivation process." This statement was refuted by Q.Li et al. [83] in a technical note stating: "In this technical

note we clarify that the modified phase change pseudopotential LB model proposed by Gong et al. was based on a misunderstanding of the derivations in the previous studies"

The exact mechanism behind the flow boiling using the pseudopotential are slowly being revealed, where Q.Li et al. [84] published in 2020 a paper detailing their qualitative observations and presenting correlations between pseudopotential LBM to thermodynamic theory.

### 2.4.3. Chapman-Enskog analysis

This section details the Chapman-Enskog analysis performed by Li et al. [81] on their DDF force correction, except that instead of internal energy, temperature is used for the secondary lattice. The analysis shows that the correction factor presented in their paper is only needed when using internal energy.

The starting point for the analysis is eq. (2.51), which rewritten and expanding the collision operator gives eq. (2.62). The tracked concentration is the internal energy, as defined in eq. (2.63).

$$g_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) - g_i(\mathbf{x}, t) = \frac{1}{\tau_g}\left(g_i - g_i^{eq}\right) \tag{2.62}$$

$$g_i = c_v T f_i^{eq} \tag{2.63}$$

First the eq. (2.62) is expanded using the Taylor series.

$$\delta t(\partial_t + \mathbf{c}_i\cdot\nabla)g_i + \frac{\delta_t^2}{2}(\partial_t + \mathbf{c}_i\cdot\nabla)^2 g_i + \ldots = -\frac{1}{\tau_g}\left(g_i - g_i^{eq}\right) \tag{2.64}$$

Then the following multiscale expansion is used, eq. (2.65), to rewrite eq. (2.64) into their scale orders eq. (2.66) and eq. (2.67).

$$\begin{aligned} \partial_t &= \partial_{t0} + \delta t\partial_{t1} \\ g_i &= g_i^{eq} + \delta_t g_i^{(1)} + \delta_t^2 g_i^{(2)} \end{aligned} \tag{2.65}$$

$$O(\delta_t): \qquad (\partial_{t0} + \mathbf{c}_i\cdot\nabla)g_i^{eq} = -\frac{1}{\tau_g}g_i^{(1)} \tag{2.66}$$

$$O(\delta_t^2): \qquad \partial_{t1}g_i^{eq} + (\partial_{t0} + \mathbf{c}_i\cdot\nabla)g_i^{(1)} + \frac{1}{2}(\partial_{t0} + \mathbf{c}_i\cdot\nabla)^2 g_i^{eq} = -\frac{1}{2}g_i^{(2)} \tag{2.67}$$

Merging eq. (2.66) into eq. (2.67) results in eq. (2.68).

$$O(\delta_t^2): \qquad \partial_{t1}g_i^{eq} + (\partial_{t0} + \mathbf{c}_i\cdot\nabla)g_i^{(1)}\left(1 - \frac{1}{2\tau - g}\right) = -\frac{1}{2}g_i^{(2)} \tag{2.68}$$

Up to this point the Chapman-Enskog analysis did not deviate from what Li et al.[81] did. For the next part, they used the internal energy approach eq. (2.63), whereas here $\sum_i g_i = T$ is used. Taking the summations of eq. (2.66) and eq. (2.68) results in the zeroth and first order moments [64].

$$\partial_{t0}(T) + \nabla\cdot(T\mathbf{u}) = 0 \tag{2.69}$$

$$\partial_{t2}(T) + \nabla\cdot\left(1 - \frac{1}{2\tau_g}\right)\sum_i \mathbf{c}_i g_i^{(1)} \tag{2.70}$$

The summation in eq. (2.70) remains since the velocity is taken from the fluid lattice and thus the first order moment is not conserved [64]. Taking the moment summation of eq. (2.66) results in eq. (2.71).

$$\sum_i \mathbf{c}_i g_i^{(1)} = -\tau_g \left[ \partial_{t0} \left( \sum_i \mathbf{c}_i g_i^{eq} \right) + \nabla \cdot \sum_i \mathbf{c}_i \mathbf{c}_i g_i^{eq} \right] \tag{2.71}$$

At this point the Chapman-Enskog analysis deviates strongly from what Li et al have. For Li et al. using $g_i^{eq} = c_v T f_i^{eq}$ result in the following two evaluations eq. (2.72) and eq. (2.73) required to fill into eq. (2.71).

$$\partial_{t0} \left( \sum_i \mathbf{c}_i g_i^{eq} \right) = \partial_{t0}(\rho c_v T \mathbf{u}) \equiv \mathbf{u} \partial_{t0}(\rho c_v T) + \rho c_v T \partial_{t0} \mathbf{u} \tag{2.72}$$

$$\nabla \cdot \sum_i \mathbf{c}_i \mathbf{c}_i g_i^{eq} = \nabla \cdot (\rho c_v T \mathbf{u}\mathbf{u}) + c_v \rho c_s^2 \nabla T + c_v T c_s^2 \nabla \rho \tag{2.73}$$

Using the usual Chapman-Enskog momentum conservation eq. (2.74), they thus obtain the macroscopic thermal equation shown in eq. (2.75).

$$\rho \partial_{t0} \mathbf{u} = -c_s^2 \rho \mathbf{u} \cdot \nabla \mathbf{u} - \nabla (c_s^2 \rho) + \mathbf{F} \tag{2.74}$$

$$\partial_t (\rho c_v T) + \nabla \cdot (\rho c_v T \mathbf{u}) = \nabla \cdot (D \nabla T + \frac{\lambda T \mathbf{F}}{c_s^2 \rho}) \tag{2.75}$$

Comparing eq. (2.75) and the macroscopic temperature equation, Li et al. identified an error term $Er = \frac{\lambda T \mathbf{F}}{c_s^2 \rho}$ and proposed a correction factor. However, when using $\sum_i g_i^{eq} = T$ the eq. (2.72) and eq. (2.73) become eq. (2.76) and eq. (2.77). Those do not require the Chapman-Enskog momentum conservation equation, and thus do not introduce the force specific error.

$$\partial_{t0} \left( \sum_i \mathbf{c}_i g_i^{eq} \right) = \partial_{t0}(T \mathbf{u}) \equiv \mathbf{u} \partial_{t0}(T) + T \partial_{t0} \mathbf{u} \tag{2.76}$$

$$\nabla \cdot \sum_i \mathbf{c}_i \mathbf{c}_i g_i^{eq} = \nabla \cdot (T \mathbf{u}\mathbf{u}) + c_s^2 \nabla T \tag{2.77}$$

There is however an error that remains, which is expressed in eq. (2.78) [64]. In another paper from Li et al. [82] they showed how to correct for that error.

$$Er = \nabla \cdot D \partial_t (T \mathbf{u}) \tag{2.78}$$

### 2.4.4. Summary thermal LBM to be implemented

The approach taken in this thesis is based on the semi-hybrid thermal method proposed by Li et al. [82], shown in eq. (2.79). The phase change term is based on the findings of Házi and Márkus [53, 93], while the semi-hybrid approach is the same as in Kamali et al. [57].

$$g_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) - g_i(\mathbf{x}, t) = -\frac{1}{\tau_g} \left( g_i - g_i^{eq} \right) + C_i + \left( G_i + \frac{1}{2} \frac{\partial G_i}{\partial t} \right) \tag{2.79}$$

$$C_i = \left( 1 - \frac{1}{2\tau_g} \right) \frac{w_i \mathbf{c}_i \cdot \frac{\partial (T\mathbf{u})}{\partial t}}{c_s^2} \tag{2.80}$$

Li et al. [82] provide a detailed analysis of the errors of this method and how to eliminate the lower order error terms. Their source term, $G_i$ eq. (2.81), implementation is different from usual, lacking the $1 - \frac{1}{2\tau_g}$ term of $Q_i$

and making it explicit in time. Otherwise their correction factors would result in an implicit scheme, which would need to be solved iteratively. The source term of the thermal simulation is shown in eq. (2.82).

$$G_i = w_i q \tag{2.81}$$

$$q = \frac{1}{\rho c_v}\nabla\cdot(\lambda\nabla T) - \nabla\cdot(D\nabla T) + T\left[1 - \frac{1}{\rho c_v}\left(\frac{\partial P_{EoS}}{\partial T}\right)_\rho\right]\nabla\cdot\mathbf{u} \tag{2.82}$$

# 3 Implementation Lattice Boltzmann Method

This chapter details how the lattice Boltzmann method simulation was set up. The code package used, and the available features of said package are introduced. Then the necessary extensions required are given such that the simulations can be used for phase change thermal flows. The chapter is concluded with steps required for verification of the code.

## 3.1. LBM simulation frameworks

There are two big opensource LBM code packages that implement LBM as flow solvers: OpenLB and Palabos. OpenLB and Palabos share many of the same functions and source code, and are often called twin code packages. A key difference is that Palabos was backed by a startup called Flowkit, which accelerated its development. Thus Palabos had for a time more features than OpenLB. This allowed it to establish itself in the scientific community as the code package to be used. Flowkit was acquired by Numeca in 2018 [108], and development was essentially stopped with no new updates being released. This allowed OpenLB to catch up feature wise with Palabos.

The University of Geneva took over the hosting and developing of Palabos [71], which, looking at their GitHub activity [38], they initialised late 2019 and started development around May 2020. In late 2019 when a choice needed to be made for which code package to use for this thesis, the OpenLB [61] was chosen due to the uncertainty of support of Palabos. The version of OpenLB used is v1.3-1.

A list of features are given in the user document provided by OpenLB, but relevant features of OpenLB v1.3-1 include:

- Lattice Boltzmann models
  - BGK model for fluids
  - Multiple Relaxation Times (MRT)
  - Porous media model
- Multi physics coupling
  - Shan-Chen two-component fluid
  - Thermal fluid with Boussinesq approximation
- Lattice Structures
  - D2Q9
  - D3Q15
  - D3Q19
  - D3Q27

A few relevant features that are also implemented, but not mentioned in their list are lattice Boltzmann models and structures for advection diffusion lattices, and convection boundaries. The biggest benefit is that OpenLB provides a robust framework in which you can simulate and modify the models easily. It also allows for the simulation to be run on multiple processing cores relative easily.

## 3.2. OpenLB extensions

OpenLB provides a great framework for LBM simulations, but does not provide all the necessary models to simulate phase change flow. Therefore, OpenLB is extended to include additional models. These models are all explained in this chapter, including a new forcing method, the Maxwell construction, scalable wall wettability, multiphase thermal flow coupling, and a semi-hybrid thermal flow coupling, and finally a proper gravity implementation. Finally, any bugs found and missing features that are included are presented.

The source code for all the extensions is provided in this github repository (note that some classes were renamed): `https://github.com/JuliusWeinmiller/ThermalMultiphaseOpenLB`

This was released to the OpenLB community in this forum post: `https://www.openlb.net/forum/topic/thermal-multiphase-flow-openlb-extension/`

In addition, this section details some measures taken regarding the multiphase inlet/outlet. The github repository examples do not contain anything specifically to the inlet/outlet. The github repository for the code used in my thesis are found here: `https://github.com/JuliusWeinmiller/ThesisLBM`

OpenLB has released a new version v1.4, which is not compatible with the code produced with the thesis. However, after releasing the code to the OpenLB community, an offer to directly implement the code into OpenLB was given. Thus hopefully in the near future, the code produced will be part of the core OpenLB framework, which includes any future versions.

### 3.2.1. Adding extensions to OpenLB

All the extensions written are separated from the main OpenLB source code. The main advantages are that it is possible to add these extensions to any future OpenLB versions. Also, by not changing the source code, the chances of introducing bugs to the core framework is reduced. However, it is necessary to point the make compiler to the extension folder such that the extensions can be used.

The extension can be downloaded from GitHub and added as a directory in the OpenLB folder. To make the extensions accessible to the compiler, the extension folder needs to be added to the linked folders. This is done via the makefile `gobal.mk`. In the definition of the variable `SUBDRIS` and `INCLUDEDIRS`, the following needs to be added: "`PathToExtension/ThesisIncludes \`".

In addition, the computation of the equilibrium densities makes usage of the GNU scientific library (GSL). Installation of GSL can be either done on system level or locally by adding it as a directory and pointing to it in the `gobal.mk` makefile. These libraries can be linked to the compiler globally or locally depending on usage. Linking GSL globally is done by adding "`-lgsl -lgslcblas`" to the variable `LIBS`. Linking it locally is done by editing the `makefile` of the program (not the global `makefile`) and adding "`-lgsl -lgslcblas`" to the line `$(CXX) $(foreach file, $(SRC), $(file:.cpp=.o)) $(LDFLAGS) -L$(ROOT)/$(LIBDIR) -l$(LIB) -lz -o $@` after the `-lz` term.

## 3.3. New forcing method

In a previous section 2.3.4, the need for an additional forcing algorithm was explained. This forcing algorithm ensures that the simulation is thermodynamically consistent. The paper by Li. et al [77] describes the method, and this subsection will detail the implementation. In section 4.2, the simulation thermodynamic consistency is verified, which is only possible due to this forcing method.

### 3.3.1. BGK implementation

Li et al. proposed a modification to the Guo forcing method, which introduces a stabilizing term to the forcing method. This forcing method will be called *modified Guo forcing method*. The proposed modification is to add a term, dependent on the local force and pseudopotential, to the velocity used in the forcing scheme

eq. (2.44). This means that this term is added after the collision algorithm has finished.

$$\mathbf{v}_{ModGuo} = \mathbf{v}_{Guo} + \frac{\sigma \mathbf{F}}{\nu \psi^2} \tag{3.1}$$

$$\nu = \tau - 0.5 \tag{3.2}$$

The modified Guo forcing method is implemented into OpenLB via the class `ForcedModifiedGuoBGKdynamics`, which is based on the class `ForcedGuoBGKdynamics`. The exact code for both the header and implementation file can be found in the file `thesisDynamics.h` and `thesisDynamics.hh` respectively. The files `thesisDynamics` were renamed to `modifiedGuoBGKDynamics` in the OpenLB release version.

### Access to pseudopotential value

The correction term is dependent on the local pseudopotential $\psi$, and thus this value needs to be accessed during class function `collide` call. This value is calculated in the multiphase coupling. Thus, it is most efficient to store the calculated pseudopotential in a field and access it during the `collide` function call.

The first part is to add a new field. This field is called `PSI_PSEUDO_RHO` and is added in the file `descriptorFieldExtended.h`. When defining the fluid lattice descriptor in the main file, the `PSI_PSEUDO_RHO` field must be included. The field is initialized with all 0, which is a problem. The usual simulation loop is to collide & stream, and then execute any coupling. In the execute coupling step, the proper pseudopotential are stored into the field. This means that the first iteration the collision and thus forcing method uses 0 for all pseudopotential values. Since in the forcing term, the correction term divides by the pseudopotential, this causes a division by 0 error. To remedy this, the field needs to be manually initialized with either the correct pseudopotential values or with a high value and accept that in the very first step the forcing term correction is not included.

The second part is to store the pseudopotential in the field. The pseudopotential is calculated in the post processor of the multiphase coupling, specifically in the file `shanChenForcedSingleComponenetPostProcessor.hh`. Since that changes the core source code, a new coupling is added called `modifiedShanChenForcedSingleComponenetPostProcessor`. The only difference is the addition of the line at the very end of the `processSubDomain` function where the `setField<descriptors::PSI_PSEUOD_RHO>` function of the `Cell` class is called to store the calculated pseudopotential.

It should be noted that additional couplings have access to the force field. If not properly handled, the next coupling may overwrite the force value stored inside. This happens when using the standard OpenLB thermal coupling, which replaces the multiphase force with the Boussinesq force.

### Implementation in dynamics

With the access to the pseudopotential and force, the velocity can be shifted in the `ForcedModifiedGuoBGKdynamics` as shown in listing 3.1. Since the shift is only for the pseudopotential force, any external force, such as gravity needs to be subtracted from the total force in the calculations. It should be noted that the `FORCE` field is storing the specific force density ($F/\rho$) whereas the velocity shift uses force density $F$, as such, the force gotten from the pseudopotential needs to be multiplied by the density.

Listing 3.1: Code to introduce the velocity shift

```
// Perform normal velocity shift
for (int iVel=0; iVel<DESCRIPTOR::d; ++iVel) {
    u[iVel] += force[iVel] / (T)2.;
}
// Perform normal normal BGK collision
T uSqr = lbHelpers<T,DESCRIPTOR>::bgkCollision(cell, rho, u, _omega);

// Perform velocity shift by Li et al.
```

```
for (int iVel=0; iVel<DESCRIPTOR::d; ++iVel) {
    u[iVel] += _prefactor * (force[iVel] - extForce[iVel]) *rho / psi / psi;
}
// Perform normal Guo forcing
lbHelpers<T,DESCRIPTOR>::addExternalForce(cell, u, _omega, rho);
```

While for the D2Q9 Li et al. [77] found that $\sigma = 0.105$ results in thermodynamic consistent behaviour, this changes for other lattice types and EoS. A.Xu et al. [148] used the modified Guo forcing scheme in combination with MRT dynamics for D3Q15 lattice. They used $\sigma = 0.12$ in combination with $a = 1/100$, usually $a = 2/49$ is used, for the PR equation of state. They mention that a lower $a$ value increases the phase interface thickness of the droplet, resulting in more stable simulations. Zhang et al. [153] used Li's modified forcing scheme in combination with MRT dynamics for D3Q19 lattice. They used a value of $\sigma = 0.3$ in the forcing scheme and $a = 1/49$ in the PR equation of state. In this thesis, the thermodynamic consistency verification tests, see section 4.2, resulted in usage of $\sigma = 0.3$.

There are two that achieved thermodynamic consistency with $\sigma = 0.105$ and one with $\sigma = 0.3$. The ratio between those is roughly 1/3, which is equal to the $c_s^2 = 1/3$ of the lattices used. This in combination with the fact that Li et al called the parameter $\nu$ the kinematic viscosity, but defined it as $\nu = \tau - 0.5$, which is contrary to general BGK viscosity definition of $\nu = c_s^2(\tau - 0.5)$ indicates the possible disparity between the two $\sigma$ found. Since the definition of $\nu$ is different by a constant factor, $\sigma$ will incorporate that and thus there is no need to change the implementation method if the thermodynamic consistency can be sufficiently approximated.

### Optimization multiphase coupling

The OpenLB's code for the calculation of the pseudopotential given in `shanChenForcedSingleComponentPostProcessor` is functioning correctly, but can be optimized. For the calculation of the pseudopotential force, the pseudopotential is needed of all the surrounding nodes. In the original implementation, these are recalculated each time when needed. That means that for a node in the middle of the fluid, its pseudopotential is calculated 19 or 9 times depending on if the lattice used is D3Q19 or D2Q9.

It is possible to pre-calculate the pseudopotential and store it into a field, much like it is already done for the density, shown in listing 3.2. This increases the RAM usage but greatly decreases CPU usage, speeding up processing overall. If the RAM impact is unacceptable, it is possible to directly store into and read from the `PSI_PSEUDO_RHO` field, which is required to save the value into anyway, but reading from it is slower, but still faster than recomputing.

Listing 3.2: Optimizations by precalculating pseudopotential

```
template<typename T, typename DESCRIPTOR>
void CombinedFullShanChenThermalCouplingPostProcessor2D<T,DESCRIPTOR>::
processSubDomain(BlockLattice2D<T,DESCRIPTOR>& blockLattice,
int x0_, int x1_, int y0_, int y1_)
{
    int nx = newX1-newX0+3; // include a one-cell boundary
    int ny = newY1-newY0+3; // include a one-cell boundary
    [...] // other definitions
    BlockData2D<T,T> psiField(nx,ny);
    [...] // other fields definitions
    for (int iX=newX0-1; iX<=newX1+1; ++iX) {
        for (int iY=newY0-1; iY<=newY1+1; ++iY) {
            [...] // velocity and density calculations
            T psi;
            interactionPotential(&psi, &rho);
            psiField.get(iX-offsetX, iY-offsetY) = psi;
            fluidCell.template setField<descriptors::PSI_PSEUDO_RHO>(psi);
        }
    }
```

```
    [...]
    // when psi is needed:
    const T psi = psiField.get(iX-offsetX, iY-offsetY);
    [...]
}
```

### 3.3.2. TRT and MRT implementation

While the BGK simulation with Li's forcing scheme was successfully implemented, simulations showed, see section 4.2.2, that the MRT implementation would help tremendously. Therefore, an attempt was made to implement the Li's forcing into TRT and MRT forcing schemes, with no success. The TRT and MRT collision operator are explained in more detail in section 2.2.4.

First a naive approach using the TRT was attempted; the TRT collision was performed, followed by BGK forcing. That is how it is done in the OpenLB's `ForcedTRTdynamics`. However, the forcing correction term is a function of the relaxation time. Neither the $\tau^+$ nor $\tau^-$ relaxation time of the TRT resulted in a stable simulation. As will be seen later, the MRT version only affects the energy and energy squared, which is the third order scalar momenta, which is odd and thus should only affect the $f^-$ populations. The BGK forcing affects all populations equally, thus leading to instabilities.

Li et al. also provided, in follow up papers [78, 79], their improved forcing scheme for D2Q9 MRT collision. OpenLB provides a `ForcedMRTdynamics` which has both the MRT collision and the addition of forces. The MRT equation with forcing is given in eq. (3.3), with the usual momenta source given in eq. (3.4), which also provides what the momenta represents in the macroscopic domain. For example, since mass is not changed, the forcing term for density momenta is 0. Their modification is given in eq. (3.5).

$$\begin{aligned} \mathbf{m}^* &= \mathbf{m} - \Lambda\left(\mathbf{m} - \mathbf{m}^{eq}\right) + \Delta t \mathbf{S} \\ \mathbf{m} &= \mathbf{M}\mathbf{f} \end{aligned} \tag{3.3}$$

While OpenLB does use the same momenta definition, gotten via the Gram-Schmidt procedure [64], the implemented **M** matrix differs. From `dynamics/mrtLatticeDescriptors.h` the following matrix was extracted for D2Q9 eq. (3.6), while the matrix used by Li et al. as they stated in [80] is different.

$$\mathbf{S} = \begin{bmatrix} 0 & \text{Density} \quad \rho \\ 6\mathbf{u}\cdot\mathbf{F} & \text{Energy} \quad \text{e} \\ -6\mathbf{u}\cdot\mathbf{F} & \text{Energy squared} \quad \zeta \\ F_x & \text{Momentum}_\text{x} \quad \text{j}_\text{x} \\ -F_x & \text{Energy flux}_\text{x} \quad \text{q}_\text{x} \\ F_y & \text{Momentum}_\text{y} \quad \text{j}_\text{y} \\ -F_y & \text{Energy flux}_\text{y} \quad \text{q}_\text{y} \\ 2\left(u_xF_x - u_yF_y\right) & \text{Diagonal stress tensor} \quad \text{p}_\text{xx} \\ \left(u_xF_y + u_yF_x\right) & \text{Offdiagonal stress tensor} \quad \text{p}_\text{xy} \end{bmatrix} \tag{3.4}$$

$$\mathbf{S}_{Li} = \begin{bmatrix} 0 \\ 6\mathbf{u}\cdot\mathbf{F} + \frac{12\sigma|\mathbf{F}|^2}{\psi^2\Delta t(\tau_e - 0.5)} \\ -6\mathbf{u}\cdot\mathbf{F} - \frac{12\sigma|\mathbf{F}|^2}{\psi^2\Delta t(\tau_\zeta - 0.5)} \\ F_x \\ -F_x \\ F_y \\ -F_y \\ 2\left(u_xF_x - u_yF_y\right) \\ \left(u_xF_y + u_yF_x\right) \end{bmatrix} \tag{3.5}$$

$$\mathbf{M}_{openLB} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & 2 & -1 & 2 & -1 & 2 & -1 & 2 & -1 \\ 4 & 1 & -2 & 1 & -2 & 1 & -2 & 1 & -2 \\ 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 1 & -2 & 1 & 0 \\ 0 & 1 & 0 & -1 & -1 & -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 & 2 & -1 & 0 & 1 & -2 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} \tag{3.6}$$

$$\mathbf{M}_{Li} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{3.7}$$

For the implementation, the simplest approach is to split the source term into the force addition and correction. The standard MRT functions of OpenLB can then be used for the collision `mrtHelpers<T,DESCRIPTOR>::mrtCo` and the forcing addition by `mrtHelpers<T,DESCRIPTOR>::addExternalForce([...])`. An additional note is that the `mrtHelpers<T,DESCRIPTOR>::addExternalForce([...])` adds the force in a computationally efficient manner. Reconstructing it seems to recover the MRT Guo forcing.

Then the Li correction is manually added. Since Li's correction only affects the energy and energy squared, only the relevant columns are needed. However, the resultant simulations were unstable and an initial circular droplet formed a square before crashing. This could be due to mismatching $\mathbf{M}$, which should not matter, or a different error. In the two papers they presented [78, 79], they have mismatching correction terms, with one missing the factor of 12. Since that is a factor, it should be correctable by changing $\sigma$. It could also be that the force source is actually different, and a mistake was made during the reconstructing of how OpenLB has implemented the `addExternalForce` function.

Not enough time remained to fully explore the MRT implementation of Li. et al's improved forcing. However, this should provide a guideline to how to either implement the MRT or a modified TRT dynamics into OpenLB.

### 3.3.3. Recommendation new forcing method

Possible recommendations to this class are as follows:

- Base the class of `ForcedGuoBGKdynamics` and not `BasicDynamics`. While there should be no computational difference, however the functions of `computeU` and `computeRhoU` are currently duplicated.
- Implement the MRT version presented by Li et al. [78] and [153] to reduce the spurious currents.
- Address the coalescence problem outlined in section 2.3.1. This is a far reaching goal which is not easily addressed.

## 3.4. Maxwell construction

The thermodynamic consistency of the new forcing method is compared to the theoretical densities given by the equation of state. The method to get those densities is the Maxwell construction, as was introduced in section 2.3.3. This feature should give the liquid and vapour densities for a given temperature ratio and equation of state, which is given by the interaction potential.

The Maxwell construction is coded both in python and c++. The Python version exists for verification and ease of producing figures. The c++ implementation is based on the python code and is used for the getting the densities for simulations. For better understanding of the c++ implementation, first the python code is explained. The implementations are verified in section 4.1.

### 3.4.1. Python implementation

The simplest implementation is given in the file `ExampleMaxwellConstruction.py` which calculates the vapour and liquid densities for a given pressure function dependent on specific volume and temperature ratio. The code is given in listing 3.3, and is briefly explained.

The main component is the for loop going through various temperature ratios. In this loop, it first sets up the pressure function to be useful for the next steps. The Maxwell construction works with the specific volume, which is the inverse of the density. The pressure function, both in OpenLB and in the example, is a function of density and thus requires a quick input conversion.

The remaining code is preparing for and executing a binomial search. The preparation is everything in the for loop before the while loop, which is the execution. The binomial search is looking for the isobaric line which intersects the pressure curve three times such that the two areas enclosed by the curve and line are equal. The equal area can also be seen as; the integral from the left intersect to the right intersect of the pressure function minus the pressure offset is equal to zero.

The two intersects of interest can be bounded by the location of the function's minimum and maximum. To find the minimum and maximum location, the scientific library Scipy is used. Scipy has only a minimum finder, however the maximum of a function is the minimum of the negative function.

The binomial search logarithm requires an initial guess and step size. The pressure range must be positive, be below the maximum pressure and above the minimum pressure. The step size must be large enough to cover the search space between the minimum and maximum pressure, and has to not allow stepping outside of this space. With these parameters an initial pressure guess and step size is generated.

With the preparation complete, the binomial search is executed. This execution is the while loop inside the

for loop. It first generates an offset pressure function, which is just the normal EOS pressure function minus the Maxwell construction pressure guess. Then the left and right most intersects are found using the scipy library. The intersects are used as the bounds in calculating the integral of the pressure offset function. If the integral is positive, then the pressure guess is too low, i.e. the area below the pressure guess line is smaller than the area above the pressure guess line. Since the guess is too low, the current pressure step size is added the guess. The step size is halved and the steps from the pressure offset generation is repeated till the area difference is less than the required accuracy. The left and right intersects are then the specific volumes, i.e. the inverse density, of the vapour and liquid at equilibrium.

Listing 3.3: Maxwell construction example python code

```
import numpy as np
import scipy.optimize as opt
import scipy.integrate as integrate

# pressure of the LBM Carnahan Starling EOS
def pressure(rho, tr, a=1.0, b=4.0):
    R = 1.0
    c = b*rho/4.
    tc = 0.18727/0.4963*a/b/R
    t = tr * tc
    return rho*R*t*((1.+c+c*c-c*c*c)/(1.-c)/(1.-c)/(1.-c))-a*rho*rho

for tr in np.linspace(0.4,0.998,100):

    P = lambda v: pressure(1/v, tr)      # make function dependent on specific volume

    guessLeft, guessRight = 4, 9 # guess is dependent on EOS used

    # silently get the minimum and maximum of the pressure function
    VminX = opt.fmin(                P, guessLeft,  disp=False)
    VmaxX = opt.fmin( lambda x: -P(x), guessRight, disp=False)

    # Generate a valid initial guess and step size
    PressureGuess = P(VmaxX) /2 + 1e-10
    step = PressureGuess/2.
    if PressureGuess< P(VminX):
        PressureGuess = P(VmaxX)/2 + P(VminX)/2
        step = (PressureGuess - P(VminX) ) /2.

    # Change the pressure guess such that the area between
    # left intersect and right intersect is zero
    diffArea = 1
    accuracy = 1e-8
    lbound, rbound = 1.01, 50000         # bounds are dependent on EOS used

    while abs(diffArea) > accuracy:
        offsetFunction = lambda V: P(V) - PressureGuess

        intersectLeft = opt.brentq( offsetFunction, lbound, VminX )
        intersectRight = opt.brentq( offsetFunction, VmaxX, rbound )

        diffArea = integrate.quad(offsetFunction, intersectLeft, intersectRight)[0]

        if diffArea<0:
            PressureGuess-=step
        else:
```

```
            PressureGuess+=step
        step/=2

    print("Rho vapour/liquid: {:.6f} / {:.3f}".format(1/intersectRight, 1/
        ↪ intersectLeft))
```

### 3.4.2. C++ implementation

The principle presented in section 3.4.1 is then included into the OpenLB framework. This is done with the class `MaxwellConstruction`. The declaration and definition are given in the files `maxwellConstruction.h` and `maxwellConstruction.h`.

To minimize the chance of mistakes occurring during simulation setup, the same interaction potential object used in the pseudopotential method coupling is used in the Maxwell construction. To allow this, the interaction potential class was extended with a hook to compute pressure. The extended interaction potential is elaborated in section 3.6.

The GNU scientific library, GSL, is included for the root finding, minimization and integration functions. For the compiler to find the library, two packages gls and glscblas need to be linked. This is done by editing the Link section of the `Makefile`, and adding `-lgsl -lgslcblas`, for example see listing 3.4.

Listing 3.4: Makefile GSL link

```
link: $(OUTPUT)

$(OUTPUT): $(OBJECTS) $(ROOT)/$(LIBDIR)/lib$(LIB).a
    @echo Link $@
    $(CXX) $(foreach file, $(SRC), $(file:.cpp=.o)) $(LDFLAGS) -L$(ROOT)/$(LIBDIR) -
        ↪ l$(LIB) -lz -lgsl -lgslcblas -o $@
```

Executing the Maxwell construction algorithm is done via `operator()`. A minimalistic example of the setup and usage is given in listing 3.5.

Listing 3.5: Maxwell construction C++ examples

```
double rho[2];
double temperatureRatio = 0.7;
MaxwellConstruction<double, decltype(interactionPotential)> maxwellConstruction(
    ↪ interactionPotential );
maxwellConstruction(rho, temperatureRatio);
double densityLatticeVapour = rho[1];
double densityLatticeLiquid = rho[0];
```

There are a few differences from the python implementation to the c++ implementations. To find the minimum, the GSL golden section algorithm is used, which requires a bounded search area in addition to the initial guess. In addition, the GSL functions used require static functions to be passed. As such, the necessary functions are made static and the required parameters are passed along as a struct.

Currently the `MaxwellConstruction` class is using a template for the interaction potential type. This is because GSL requires a static function to be passed. Since in c++ virtual static functions are not possible, the base class `InteractionPotentialExtended` does not contain the `computePressure` function required in the `MaxwellConstruction`. The static `computePressure` function are individually given for derived classes. As such a template was used to circumvent the missing `computePressure` function in the base class.

Possible recommendations for this extensions are:

- A possible error source is the templated interaction potential. Removing the need to pass the interaction potential type in the declaration can help to reduce it. Right now it is possible to pass along an object which does not have the right function set.

## 3.5. Thermal multiphase flow

This section provides the implementation of the phase change mechanics detailed in section 2.4. At the moment, only isothermal multiphase flow simulations are possible. The thermal implementation based on the Boussinesq approximation does not work with multiphase flow. Thus a complete new thermal coupling is required.

The implementation is split into three sections: the advection diffusion dynamics, which covers the dynamics implementation into OpenLB, the coupling between the fluid and thermal lattice, which details how each lattice interacts with the other, and how the fluid properties in the interface are calculated.

### 3.5.1. Advection diffusion dynamics

Similar to the new forcing implementation, the thermal dynamics requires a new implementation as well. This implementation is given in `thesisAdvectionDiffusionDynamics.hh` and `thesisAdvection-DiffusionDynamics.h`. The thermal dynamics to be implemented are summarized in section 2.4.4, and repeated below for convenience, and follows the 2017 paper by Li et al.[82].

$$g_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) - g_i(\mathbf{x},t) = -\frac{1}{\tau_g}\left(g_i - g_i^{eq}\right) + C_i + \left(G_i + \frac{1}{2}\frac{\partial G_i}{\partial t}\right)$$

$$C_i = \left(1 - \frac{1}{2\tau_g}\right)\frac{w_i \mathbf{c}_i \cdot \frac{\partial (T\mathbf{u})}{\partial t}}{c_s^2}$$

$$G_i = w_i \phi$$

Li et al. proposed dynamics for both the BGK as well as MRT collision operators. Both are implemented into the simulation framework. OpenLB's thermal lattices are D3Q7 and D2Q5 based on this series of papers [63, 75, 87], which is of a lower isotropy than the D2Q9 given by the paper. It requires MRT dynamics on such pruned lattices to match the accuracy BGK dynamics of the extended lattice for thermal flow. As such, the MRT dynamics presented by Li et al was rewritten for usage with OpenLB lattice. An introduction to the MRT collision operator is given in section 2.2.4.

For a complete implementation of Li's thermal dynamics, three new fields are required. The first field, named `PHI_THERMAL`, is used to communicate with the thermal post processor, which calculates the source term. The explanation of how the source term is computed numerically is given in section 3.5.2. The second and third field are required for the temporal derivative of the source `PREV_PHI` and correction terms `PREV_T_V`.

Since the lattice time step is equal to 1, the temporal derivatives are just the difference between the current and previous values. The implementation of the source term into the dynamics is straight forward, but the $T\mathbf{u}$ correction term is a vector, thus requiring calculating the derivative for each dimension. The specific code is given in listing E.5 and listing E.6.

#### Thermal BGK dynamics

The class is heavily based on OpenLB's `SourcedAdvectionDiffusionBGKdynamics`, provided in the `dynamics/advectionDiffusionDynamics.h` file. The `computeEquilibrium` function already has the first order method by default, and is in line with the method by Li et al.

The collision function follows the same layout as OpenLB's `SourcedAdvectionDiffusionBGKdynam-ics` collision function. First the derivatives are calculated, followed by performing the BGK collision. Then

the source term and correction factors are added to the populations. The core of the previous steps are shown in listing 3.6. The collide function ends by updating the statistics as is done in OpenLB and updating the fields relevant to the derivative calculations.

Listing 3.6: Summarized derivative implementation

```
[...]
for (int iD=0; iD< DESCRIPTOR::d; ++iD){
    derivTu[iD] = temperature*u[iD] - prev_Tu[iD];
}
T derivPhi = phi - prev_phi;

T uSqr = lbHelpers<T,DESCRIPTOR>::bgkCollision( cell, temperature, u, _omega );

for (int iPop=0; iPop < DESCRIPTOR::q; ++iPop) {
    [...]
    cell[iPop]+= weight*(phi + 0.5*derivPhi);
    cell[iPop]+= (1-0.5*omega)*weight*cdTu*descriptors::invCs2<T,DESCRIPTOR>();
}
[...]
```

### Thermal MRT dynamics

This class is based on the OpenLB's `AdvectionDiffusionMRTdynamics` and is named `thesisAdvectionDiffusionMRTdynamics`. The `computeEquilibrium` of that class is however second order, and thus needs to be changed to the first order method listing 3.7. Additionally, OpenLB has in v1.3-1 a bug which results in erroneous calculations when using D3Q7, see section 3.11 how to fix it.

Listing 3.7: Equilibrium first order of thesis AD dynamics

```
template<typename T, typename DESCRIPTOR>
T thesisAdvectionDiffusionMRTdynamics<T, DESCRIPTOR>::computeEquilibrium(int iPop, T
    ↪ rho,
const T u[DESCRIPTOR::d], T uSqr) const
{
    return lbHelpers<T, DESCRIPTOR>::equilibriumFirstOrder(iPop, rho, u);
}
```

Since the example class `AdvectionDiffusionMRTdynamics` does not feature the addition of sources, the sourcing had to be implemented separately. Following the paper, the MRT AD equation is shown in eq. (3.8) for velocity space and in eq. (3.11) in momentum space.

$$g_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) = g_i(\mathbf{x},t) - \bar{\Lambda}_{ij}\left(g_i - g_i^{eq}\right) + \Delta t S_i' \tag{3.8}$$

$$\bar{\Lambda}_{ij} = \left(\mathbf{M}^{-1}\Lambda\mathbf{M}\right)_{ij} \tag{3.9}$$

$$\Lambda = \mathrm{diag}(\omega_0, \omega_1, ..., \omega_q) \tag{3.10}$$

$$\mathbf{m}^* = \mathbf{m} - \Lambda\left(\mathbf{m} - \mathbf{m}^{eq}\right) + \Delta t\mathbf{S} \tag{3.11}$$

$$\mathbf{S}^9 = (S_0, 0, 0, 0, 0, 0, 0, 0, 0)^T \tag{3.12}$$

$$S_0 = \phi + \frac{1}{2}\partial_t\phi \tag{3.13}$$

For the D2Q9, the source is given by Li et al as $\mathbf{S}^9$. For D2Q5 and D3Q7 this source is taken as shown below, with $S_0$ staying the same as in eq. (3.13), where $\partial_t\phi$ is shorthand for the temporal derivate of the source term.

$\Lambda$ in eq. (3.10) is the relaxation frequency diagonal, where in general $\omega = 1/\tau$.

$$\mathbf{S}^7 = (S_0, 0, 0, 0, 0, 0, 0)^T \tag{3.14}$$

$$\mathbf{S}^5 = (S_0, 0, 0, 0, 0)^T \tag{3.15}$$

For the D2Q9, the momenta are shown in eq. (3.16) [82] and for the OpenLB implementation of D2Q5 [75] it is shown in eq. (3.17). More specifically, OpenLB uses the standard a = 2/3 of the generic D2Q5, where a is a tuning coefficient. The momenta recovered for D2Q7 is shown in eq. (3.18) [63].

$$\mathbf{m}^{eq,9} = T(1, -2, 1, u_x, -u_x, u_y, -u_y, 0, 0)^T \tag{3.16}$$

$$\mathbf{m}^{eq,5} = T(1, u_x, u_y, 2/3, 0) \tag{3.17}$$

$$\mathbf{m}^{eq,7} = T(1, u_x, u_y, u_z, 3/4, 0, 0) \tag{3.18}$$

This becomes important since in the next step the implementation of the thermal model deviates from Li et al.'s paper. Their next step is to correct for the $T\mathbf{u}$ error, which was done by means of temporal derivative in the BGK dynamics. To correct for that error, Li et al. managed to numerically exploit the MRT collision procedure to express the derivative in terms of the higher order momenta present in the D2Q9 lattice. Thus removing the need to compute the derivative and store the previous vector of $T\mathbf{u}$.

However, with the pruned lattice of D2Q5 and D3Q7, these higher order momenta are not able to be computed, thus the error correction is done the same way for the BGK, which is the temporal derivative. Li et al. have expressed the implementation as an intermediate step, shown in eq. (3.19). For D2Q9, the third and fifth momenta correspond to the first order moments.

$$\begin{aligned} m^*_{3,new} &= m^*_3 + \Delta t\left(1 - \frac{\omega_3}{2}\right)\partial_t(Tu_x) \\ m^*_{5,new} &= m^*_5 + \Delta t\left(1 - \frac{\omega_5}{2}\right)\partial_t(Tu_y) \end{aligned} \tag{3.19}$$

For the D2Q5 and D3Q7, the first order moments, which required correcting, are the second and third momenta, and second, third and fourth momenta respectively. The resultant correction for D2Q5 is shown in eq. (3.20).

$$\begin{aligned} m^*_{2,new} &= m^*_2 + \Delta t\left(1 - \frac{\omega_2}{2}\right)\partial_t(Tu_x) \\ m^*_{3,new} &= m^*_3 + \Delta t\left(1 - \frac{\omega_3}{2}\right)\partial_t(Tu_y) \end{aligned} \tag{3.20}$$

Next is shown how to implement this new correction for the MRT dynamics class. The constructor additionally needs to pre-calculates and stores the $\mathbf{M}^{-1}(1 - \Lambda/2)$ matrix, which is required for the $T\mathbf{u}$ correction, which is shown in listing 3.8.

Listing 3.8: Precalculation of the MRT D2Q5 correction matrix

```
for (int iPop = 0; iPop < DESCRIPTOR::q; ++iPop) {
  for (int jPop = 0; jPop < DESCRIPTOR::q; ++jPop) {
    invM_correct[iPop][jPop] = T();
    for (int kPop = 0; kPop < DESCRIPTOR::q; ++kPop) {
      if (kPop == jPop) {
        invM_correct[iPop][jPop] += descriptors::invM<T,DESCRIPTOR>(iPop,kPop)
            ↪ * (1 - rt[kPop]/2. );
      }
    }
  }
}
```

The correction matrix containing $\mathbf{M}^{-1}(1-\Lambda/2)$ matches with momenta. Since only the first order moments need correcting, only the first 2/3 columns, depending on if the simulation is 2D or 3D, after the first column (the 0th order moment) are of interest. The column index coincide with the shear index. The rest of the columns are multiplied by 0. To save on calculation, this is expressly implemented, as shown in listing 3.9, which is executed after performing the basic MRT collision.

Listing 3.9: Correcting first order moments in MRT D2Q5

```
for (int iPop = 0; iPop < DESCRIPTOR::q; ++iPop){
  for (int iShear = 0; iShear < descriptors::shearIndexes<DESCRIPTOR>(); ++iShear) {
    cell[iPop] += invM_correct[iPop][descriptors::shearViscIndexes<DESCRIPTOR>(
        ↪ iShear)] * derivTv[iShear] ;
  }
}
```

The source, **S**, in eq. (3.11) is directly defined in moment space, and as such requires only a conversion into velocity space. This is performed as shown in listing 3.10. In terms of order, it can either be before or after the $T\mathbf{u}$ correction, but needs to be after the MRT collision.

Listing 3.10: Performing the addition of the source term for MRT D2Q5

```
for (int iPop = 0; iPop < DESCRIPTOR::q; ++iPop){
    cell[iPop] += descriptors::invM<T, DESCRIPTOR>(iPop,0)*(phi + 0.5 * derivPhi);
}
```

#### Recommendation

There is a bug in OpenLB's D3Q7 MRT collision matrix, which leads to erroneous simulation results. A more detailed analysis of this bug and how to fix it is given in section 3.11.1 *D3Q7 MRT collision matrix velocity set error*. Till then, the 3D MRT thermal simulation are erroneous.

### 3.5.2. Advection diffusion source term calculation

Since the calculation of the source term for the advection diffusion function differs drastically from what OpenLB has, the source term calculation is explained in this section separately. A summary of the implemented term is given in section 2.4.4. The source term can be split into two main parts: the diffusion correction eq. (3.21) and the phase change term eq. (3.22).

$$q_D = \frac{1}{\rho c_v}\nabla\cdot(\lambda\nabla T) - \nabla\cdot(D\nabla T) \tag{3.21}$$

$$q_L = T\left[1 - \frac{1}{\rho c_v}\left(\frac{\partial P_{EoS}}{\partial T}\right)_\rho\right]\nabla\cdot\mathbf{u} \tag{3.22}$$

#### Diffusion correction

The diffusion correction is needed due to the limit of using LBM to have two differing diffusivities for both fluid and thermal lattice. The term containing the LBM diffusivity $D = \frac{\lambda}{\rho c_v} = (\tau_g - 1/2)c_s^2$, needs to be subtracted. Since a constant diffusivity is applied in the simulation, the term that is subtracted is simplified to $D\nabla^2 T$. The actual diffusivity is then added to the source term. Since the thermal conductivity $\lambda$ changes between the liquid and vapour, the term inside the first nabla is expanded.

$$q_D = \frac{1}{\rho c_v}\nabla\cdot(\lambda\nabla T) - \nabla\cdot(D\nabla T) \tag{3.23}$$

$$\frac{1}{\rho c_v}\nabla\cdot(\lambda\nabla T) - D\nabla^2 T \tag{3.24}$$

$$\frac{\lambda}{\rho c_v}\nabla^2 T + \frac{1}{\rho c_v}\nabla\lambda\nabla T - D\nabla^2 T \tag{3.25}$$

$$\left(\frac{\lambda}{\rho c_v} - D\right)\nabla^2 T + \frac{1}{\rho c_v}\nabla\lambda\nabla T \tag{3.26}$$

Since section 3.5.4 assumes a linear gradient between the fluid properties, the term $\nabla\lambda$ can be numerically optimized. The derivative of the thermal conductivity with respect to the density is constant, and thus the computation of the thermal conductivity can be saved.

$$\nabla\lambda = \frac{d\lambda}{d\rho}\nabla\rho \tag{3.27}$$

The calculations for the thermal and density gradients, $\nabla T$, $\nabla^2 T$ and $\nabla\rho$ makes use of the central difference method, shown in eqs. (3.28) to (3.30), where the summation happens over the dimensions $d$ for x and y, in 2D and x, y, z for 3D. Since the lattice spatial step is equal to 1, the $\Delta x$ in the denominator is omitted.

$$\nabla\rho_{\mathbf{x}} = \sum_d \frac{-\rho_{\mathbf{x}-d} + \rho_{\mathbf{x}+d}}{2} \tag{3.28}$$

$$\nabla T_{\mathbf{x}} = \sum_d \frac{-T_{\mathbf{x}-d} + T_{\mathbf{x}+d}}{2} \tag{3.29}$$

$$\nabla^2 T_{\mathbf{x}} = \sum_d T_{\mathbf{x}-d} - 2T_{\mathbf{x}} + T_{\mathbf{x}+d} \tag{3.30}$$

Using the central difference scheme makes the thermal simulation not a full LBM simulation but a hybrid. Since the tools used to implement this hybrid scheme utilizes the LBM source, the complete simulation is more of a semi-hybrid approach. A drawback to using the central difference scheme is that it inherits the instabilities of said scheme, such as the checkerboard instabilities.

### Phase change correction

The phase change term contains the thermal source due to latent heat and a correction term $T\nabla\cdot\mathbf{u}$, which originates from the difference between what LBM solves compared to the macroscopic equation, as was explained in more detail in section 2.4.2. The derivative of the pressure with respect to temperature at constant density is provided by the extended interaction potential class via `computeDerivativeConstRho` and is discussed in section 3.6.

The divergence of the velocity, $\nabla\cdot\mathbf{u}$, is calculated using the central difference scheme as well, shown in eq. (3.31). The notation $\mathbf{u}_d$ means the velocity in the $d$ dimension.

$$\nabla\cdot\mathbf{u}_{\mathbf{x}} = \sum_d \frac{-\mathbf{u}_{d,\mathbf{x}-d} + \mathbf{u}_{d,\mathbf{x}+d}}{2} \tag{3.31}$$

For the implementation into OpenLB, special care needs to be taken for the unit scale used in latent heat term the phase change term. A detailed analysis on how to do that can be found in section 5.3.2. The section is shortly summarized: OpenLB uses a different unit scale between the EoS and temperature. The simplest approach, requiring the least amount of code and conversion factors, is to use a non-dimensionalized specific $R$ in the equation of state, i.e. $R \neq 1$. The temperature used in the latent heat term should also be converted

to the temperature ratio. The resultant phase change term is given in eq. (3.32), where $Tr(T)$ is a function converting the OpenLB temperature to temperature ratio.

$$q_L = \left[ T - \frac{Tr(T)}{\rho c_v} \left( \frac{\partial P_{EoS}}{\partial T} \right)_\rho \right] \nabla \cdot \mathbf{u} \tag{3.32}$$

#### Advection diffusion instability

The "semi-hybrid" implementation effectively is the explicit scheme of the advection diffusion problem. As such it inherits the stability issues as well [9]. Under certain conditions, instabilities arise, such as the checkerboard shown in fig. 3.1, leading to simulation crashes.

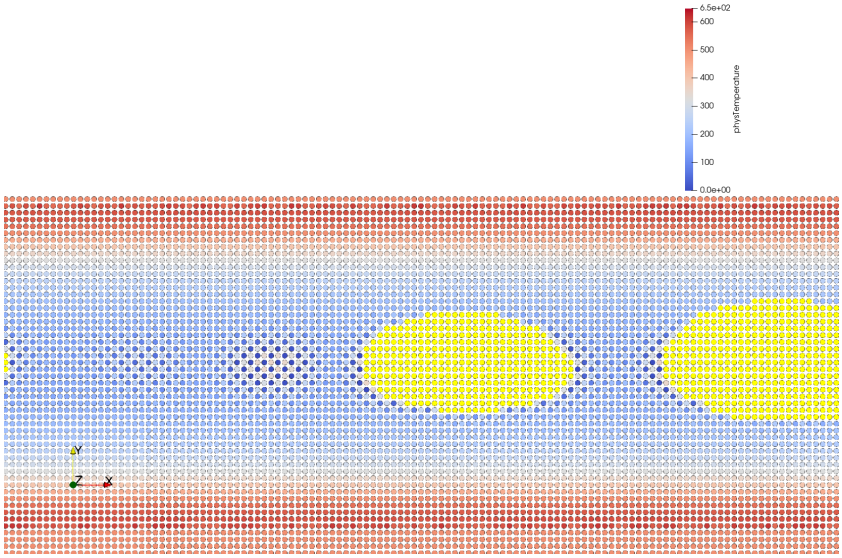



Figure 3.1: Checkerboard instabilty due to the explicit advection diffusion implementation

Li et al. [79] proposed early on in 2015 an pure hybrid advection diffusion solver based on the fourth order Runge-Kutta scheme. Gong and Chen followed suit in 2018 [45], where they previously used the default LBM DDF [41, 43].

### 3.5.3. Thermal and fluid lattice coupling

The impact of the temperature on the fluid lattice is only in the pseudopotential calculation of the fluid post processor, leaving the dynamics untouched. Instead of calculating the interaction potential as a function of only rho, temperature is used as well. Using the code snippet listing 3.2 as example, the line `interactionPotential(&psi, &rho);` is replaced with `interactionPotential(&psi, &rho, &temperatureRatio);`. The extended interaction potential, see section 3.6, uses the temperature ratio instead of lattice temperature for the calculations, thus the post processor needs access to the local temperature and converts it to temperature ratio.

To access the temperature, the same method as for OpenLB's `navierStokesAdvectionDiffusionCouplingPostProcessor` is used. They passed the thermal lattice as a partner lattice into the post processor generator, thus allowing the post processor to access the thermal lattice as a partner lattice. The temperature is calculated as normal via the `computeRho` function. The temperature ratio is gotten via the fluid properties, see section 3.5.4.

### 3.5.4. Fluid properties

The thermal coupling requires knowing certain fluid properties, e.g. thermal conductivity. These are different dependingly if the fluid is a vapour or a liquid. In the simulation there is only one component, and to determine whether a node is fluid or vapour the density is used. Thus fluid properties are a function of density.

To retrieve the relevant fluid properties, the properties are expressed in terms of a marker function [31, 41, 43], shown in eq. (3.33), where $\chi$ is a placeholder for the relevant fluid property. The marker function usually is zero for vapour, and one for liquid [152]. A simple and commonly used marker function is the linear scaling, shown in eq. (3.34), but more complex functions are used as well eq. (3.35) [31].

$$\chi = (1-\phi)\chi_{vap} + \phi\chi_{liq} \tag{3.33}$$

$$\phi(\rho) = \frac{\rho - \rho_{vap}}{\rho_{liq} - \rho_{vap}} \tag{3.34}$$

$$\chi = \frac{\chi_{liq} + \chi_{vap}}{2} + \frac{\chi_{liq} + \chi_{vap}}{2} \tanh\left(\frac{2\left(\rho - (\rho_{liq} + \rho_{vap})/2\right)}{\rho_{liq} + \rho_{vap}}\right) \tag{3.35}$$

The fluid properties and the linear marker function are combined into one class `fluidProperties.h` and `fluidProperties.hh`. This allows setting up the fluid properties once and passing them on. The class also contains a `print & write` function to better keep track of the used simulation parameters. A quick verification function `verify` is called in the constructor to ensure that the functions return the correct fluid properties for both the vapour and liquid density endpoints. The class function `scaleProperty` contains the relevant marker function.

## 3.6. Extended interaction potential

The interaction potential class in OpenLB defines the pseudopotential function. That can be one of the original pseudopotential functions by Chen and Shan, or an implementation of an equation of state.

In the section 3.4.2 it was stated that, for the Maxwell construction, a direct call to the pressure computation is needed. In addition, the thermal multiphase coupling detailed in section 3.5 requires the derivate of the pressure function with respect to temperature. In the same computation, the equation of state's critical temperature is required. These three features are added to the base interaction potential class, creating an extended version.

As usual, the declaration can be found in `interactionPotentialExtended.h` and definition in `interactionPotentialExtended.hh`. These files are based on the classes from the OpenLB's `interactionPotential.h` and `interactionPotential.hh`. Since the additional functions beyond the standard call operator need to be called by the modules, a base class is introduced such that the additional functions can be found when passed to the modules. The main operational difference of the call operator, between this and OpenLB's implementation, is that, for simplicity's sake, instead of temperature, the temperature ratio is passed to the call function.

The first additional capability is for the `MaxwellConstruction` class and consists of the functions `computePressure`, `getCriticalDensity` and `getParameters`, and the struct `staticPressureParameters`. The function `computePressure` computes the pressure the same way as with for pseudopotential via the call operator. The function is static, since that is required by the GNU Scientific Library. The relevant parameters are passed along via the `staticPressureParameters` struct, which is gotten via the `getParameters` function. In c++ virtual static functions are not possible, and thus the `computePressure` function is not available to the base class. See section 3.4.2 for a more detailed explanation of how that impacts the usage of the `MaxwellConstruction` class.

The second function set is for the thermal coupling, which consists of the functions `computeDerivativeConstRho` and `getCriticalTemperature`. The `computeDerivativeConstRho` function computes the derivative of the pressure with respect to the temperature, which is used in the source term calculations of the phase change. The derivatives for both the CS and PR EoS are detailed in eq. (3.36) and eq. (3.38). The `getCriticalTemperature` returns the critical temperature, which is required from converting temperature between LBM scale to EOS scale.

$$\left(\frac{\partial p_{CS}}{\partial T}\right)_{\rho} = \rho R \frac{1 + c + c^2 - c^3}{(1-c)^3} \tag{3.36}$$

$$c = b\rho/4 \tag{3.37}$$

$$\left(\frac{\partial p_{PR}}{\partial T}\right)_{\rho} = \frac{\rho R}{1-b\rho} - \frac{a\rho^2}{1+2b\rho-b^2\rho^2}\frac{\partial \alpha}{\partial T} \tag{3.38}$$

$$c = \left(0.37464 + 1.54226\omega_{EoS} - 0.26992\omega_{EoS}^2\right) \tag{3.39}$$

$$\frac{\partial \alpha}{\partial T} = \frac{c^2}{T_c} - \frac{c + c^2}{\sqrt{T T_c}} \tag{3.40}$$

## 3.7. Gravity coupling

Since previously the gravity in OpenLB was introduced via the Boussinesq approximation, which cannot be used, a new gravity coupling needs to be added.

The implementation of gravity follows OpenLB's existing style of gravity implementation featured in `navierStokesAdvectionDiffusionCouplingPostProcessor`. They require that the direction and magnitude of the gravitational acceleration are passed separately. During the construction of the post processor, the direction is normalized. Since the gravity magnitude does not change, it allows for a simple and safe way to provide the direction.

The force required in the fluid dynamics are density normalized. As such, the force being added to the field are implemented as given in eq. (3.41), where $\rho_0$ is the reference density equal to 0, $\rho_l$ or $\rho_{avg}$ depending on which implementation method explained in section 2.3.6 (full, buoyant or zero momentum addition) used. Following [41, 45, 82], $\rho_0 = \rho_{avg}$ is used.

$$\mathbf{F}_g(\mathbf{x}) = \left(1 - \frac{\rho_0}{\rho(\mathbf{x})}\right)\mathbf{g} \tag{3.41}$$

## 3.8. Combined multiphase, thermal and gravity coupling

The coupling for multiphase pseudopotential, thermal and gravity are implemented separately from each other. This means that three coupling and post processors are added to the simulation. Each post processor re-calculates the fluid lattice density, making the calculations redundant. As such, the three couplings are merged together into one combined coupling post processor, resulting in `combinedFullShanChenThermalCouplingPostProcessor`.

Combining all the couplings into one also makes the programming safer, with less chance of a coupling accidentally overwriting the field value which was already previously calculated. As such, all thermal simulation performed in this thesis use the combined coupling. In case gravity is not wanted, a gravitational magnitude of 0 is used.

## 3.9. Boundary condition

In OpenLB, the simulation domain boundaries are best handled via the internal boundary manager. This allows to impose a boundary on the geometry via material, while the manager ensures proper handling of the implementation, which can get complicated due to edges and corners.

For the walls, the no-slip bounce back method is used, since that is the only supported wall type that supports setting the wettability, see section 3.10. For the open boundaries, according to OpenLB's documentation, it supports velocity and pressure boundaries. A convection boundary is implemented which approximates a Neumann boundary condition, which is not mentioned in OpenLB's documentation. While it is meant for advection-diffusion problems it can also be used for the fluid.

Pseudopotential LBM has a unique problem for defining inlet and outlet conditions, since pressure boundary conditions are not very useful. In LBM density and pressure are linked via the constant lattice speed. Therefore, setting a pressure also sets the density. The pressure boundary conditions does not allow for setting the velocity [61].

For the outlet, it is required that it can handle a multiphase flow. Setting it to a constant pressure, and thus density, would result in defining the phase at the outlet. A velocity boundary can handle a multiphase outflow including droplets, but any droplets get smeared across the interface. A convective boundary performs the best for droplets exiting the simulation domain.

To set a constant massflow for the inlet both the velocity and density, and by extension pressure, need to be set. As such, both velocity and pressure boundary managers are not suitable for the inlet. Relaxing the requirement of constant massflow, it is possible to set the density, i.e. phase, at the inlet with the pressure boundary.

It is also possible to manually set both the inlet density and velocity, without using the internal boundary manager, see https://www.openlb.net/forum/topic/setting-both-density-and-velocity-at-boundary/. This needs a convective outlet to not over constrain the simulation.

### Issues regarding inlet

Setting both velocity and pressure manually and circumventing the boundary manager has a unique set of problems, since no empty nodes are created. The empty node is a simulation node which has no dynamics associated with it. They are necessary such that numerical methods using data from surrounding nodes, such as the pseudopotential and the semi-hybrid thermal implementation, work. Correct functioning of the boundary dynamics is handled by the boundary manager. In case that is not done, the simulation runs into a segmentation error.

As such the inlet material must be excluded from any coupling. For the multiphase flow, that means that the inlet is not experiencing the pseudopotential force, which is not a problem since the density and velocity are manually set. For thermal flow, this means that the thermal lattice velocity field is not updated at the inlet. This can result in very negative temperatures. However is easily remedied by manually updating the velocity field when updating the inlet velocity, as shown in listing 3.11.

Listing 3.11: "Manual inlet velocity coupling"

```
void setBoundaryValues( [...] )
{   [...]
    // Updating inlet velocity, which is defined in InPoiseuilleU
    [...]
    ADlattice.defineField<descriptors::VELOCITY>(superGeometry, materialInlet,
        ↪ InPoiseuilleU);
}
```

An example of a single phase channel flow is given in figs. 3.2 to 3.4, where fig. 3.2 is showing the channel resulting velocity field. The inlet is set to a constant 273 Kelvin and the walls at a constant 293K. With correct

implementation the resultant temperature field is as in top of fig. 3.3. Without correct implementation the temperature field, see fig. 3.4, looks similar, but the inlet temperature drops to around 230K, a 40K difference. Also visible is the inlet thermal constant temperature, which has the correct inlet temperature, but does not interact with the surrounding due to lack of coupling. Note that fig. 3.3 and fig. 3.4 show the exact same simulations, just with a different scale.

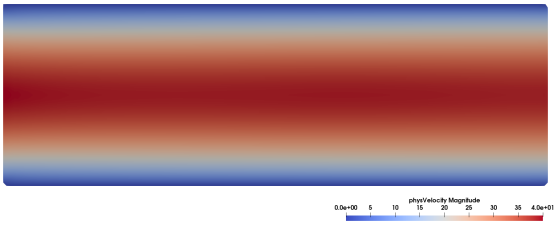



Figure 3.2: Velocity field of single phase thermal 3D channel flow, using manual velocity and pressure inlet and convection outlet. Slice is showing the middle cross-section of the channel.

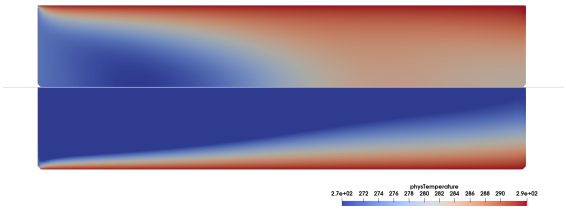



Figure 3.3: Temperature field of single phase thermal 3D channel flow using manual velocity and pressure inlet and convection outlet. Slice is showing the middle cross-section of the channel. Proper inlet coupling (top) and improper (bottom). The scale is based on the min/max temperatures of proper inlet simulation

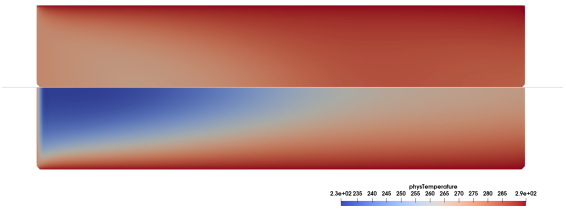



Figure 3.4: Temperature field of single phase thermal 3D channel flow using manual velocity and pressure inlet and convection outlet. Slice is showing the middle cross-section of the channel. Proper inlet coupling (top) and improper (bottom). The scale is based on the min/max temperatures of improper inlet simulation

Another problem which occurs at the inlet is no fluid flow even with inflow velocity. This is due to setting a too low density at the inlet. Even with a high inflow velocity, the simulation experiences a reverse flow. Since the density at the inlet is below the saturation density, it attracts vapour from the surrounding nodes. This increases the density at the inlet nodes. At the next timestep, the density and velocity at the inlet is manually set, thus removing the excess density. This means that there is always a force acting into the direction of the inlet, resulting in no fluid flow. With a convection boundary and pressure outlet, this results in a reverse flow, and with a velocity outlet in unstable simulation. The solution is very easy, which is to pick a density at or slightly above the saturation density.

A potential error source, for a multiphase inlet bounded by walls, is the wall wettability. The simplest implementation is a constant pressure, i.e. density, across the inlet, thus not accounting for the impact of the wall wettability. This results that the fluid near the wall nodes are expelled away, shown in fig. 3.5. This results in some compression artefacts. This is visible in both the velocity field fig. 3.6 and in the thermal field fig. 3.7, where this compression results in a streak of temperature. The impact of the temperature artefact can be mitigated by extending the constant temperature field of the inlet into the simulation domain.

### Issues regarding outlet

The outlet of the VLM heating chamber transitions to the inlet of the nozzle. During nominal operations, that flow is choked. Since the massflow is known at the throat, it is tempting to set a massflow constrained velocity outlet. To calculate the velocity, the density is needed. Two densities can be used: vapour density, or average outlet density.

The argument for using the average outlet density is that it more accurately describes the impact of multiphase flow on the heating chamber. However, the simulation is at risk of getting stuck with a blocked throat.

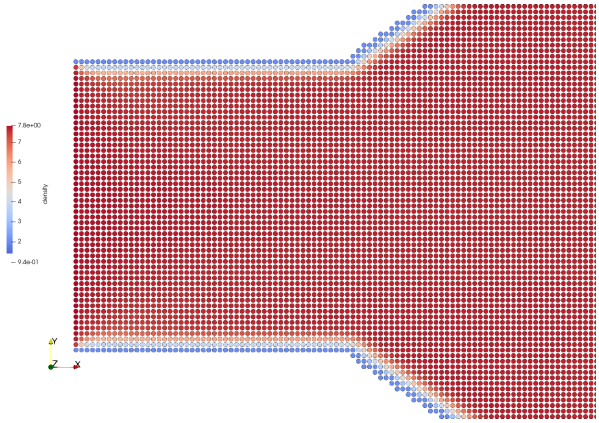


Figure 3.5: Density compression at inlet due to wall wettability

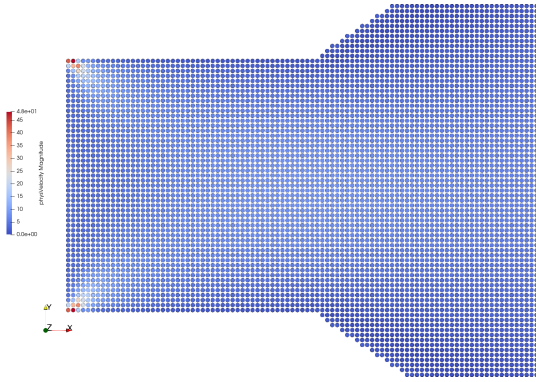


Figure 3.6: Velocity artefact at inlet due to compression

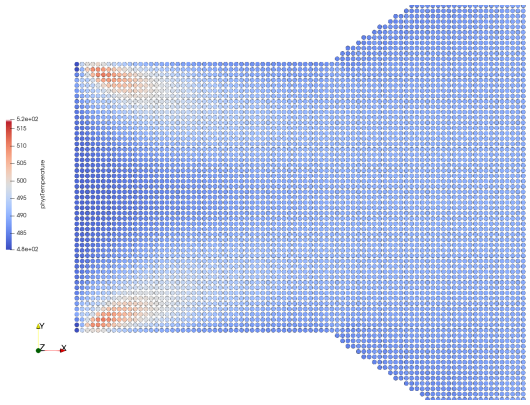


Figure 3.7: Temperature artefact at inlet due to compression

Once a droplet large enough is near the outlet fig. 3.8, it gets sucked into the outlet fig. 3.9, a erroneous phenomena which is also documented in literature [90]. Since it fully covers the outlet fig. 3.10, the velocity drastically falls, and the outlet remains blocked throughout the rest of the simulation fig. 3.11.

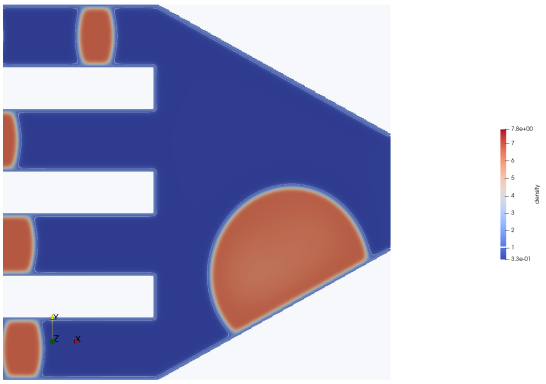


Figure 3.8: Multiphase velocity outlet behaviour: initial condition with unblocked throat, $\Delta t = 50e3$

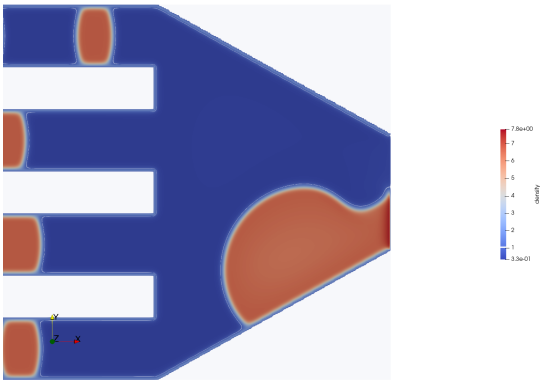


Figure 3.9: Multiphase velocity outlet behaviour: suction effect of velocity boundary, $\Delta t = 60e3$

The other option for a velocity boundary is to assume vapour density and thus a constant velocity. In this case two abnormalities occur: the suction effect and liquid stretching. The suction effect [90] was already mentioned and shown in fig. 3.9. The difference is that instead of slowing down the outflow velocity, the droplet speed increases, and thus the droplet vanishes nearly instantly. This liquid stretching is shown in fig. 3.12 and fig. 3.13.

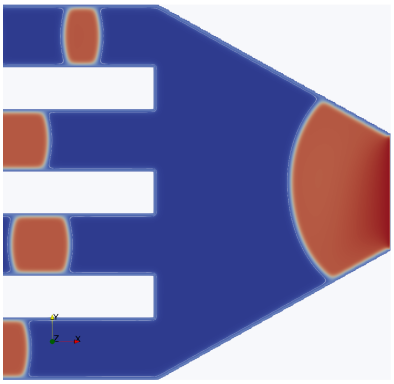


Figure 3.10: Multiphase velocity outlet behaviour: fully blocked outlet, $\Delta t = 80e3$

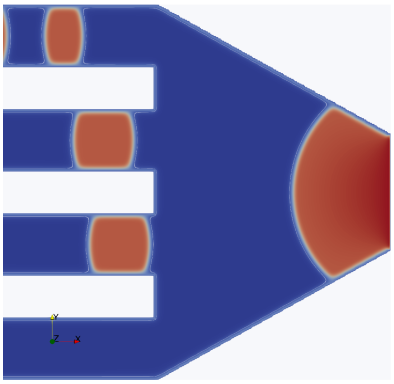


Figure 3.11: Multiphase velocity outlet behaviour: fully blocked outlet, $\Delta t = 146e3$

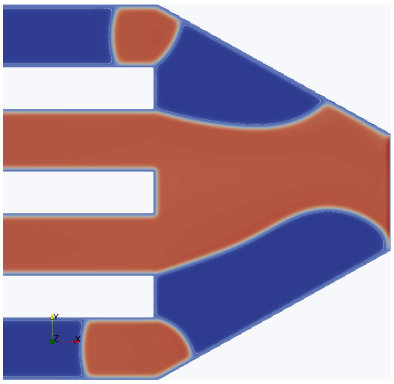


Figure 3.12: Multiphase velocity outlet behaviour: liquid stretching initial, $\Delta t = 40e3$

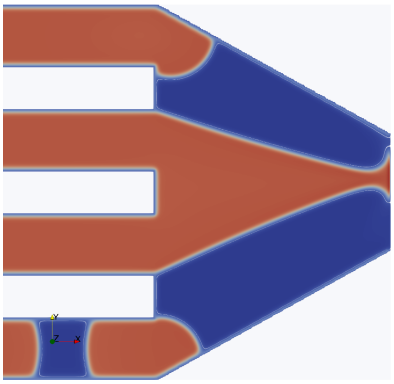


Figure 3.13: Multiphase velocity outlet behaviour: liquid stretching final, $\Delta t = 55e3$

## 3.10. Setting wall wettability

OpenLB's `BounceBack` class for no-slip wall has an option to implement a fixed wall density to use in combination with the pseudopotential method to set wall wettability. The closer the fixed wall density is to the liquid density, the more hydrophilic the wall becomes. This may result in some instability during the first few simulation steps due to a steep density gradient. For example, a hydrophilic wall (high wall density) surrounded by vapour or a hydrophobic wall (low wall density) surrounded by liquid.

A work around is to initialize the simulation domain with density equal to the wall density. However, this of course leads to phase separation and is not representative of actual initial conditions. This is shown in figs. 3.14 to 3.17, which also shows in fig. 3.15 the Plateau–Rayleigh instability, i.e. the beading of the fluid thread, in the channels themselves.

Figure 3.14: Multichannel initial conditions $\rho = \rho_w$ progression $\Delta t = 0$

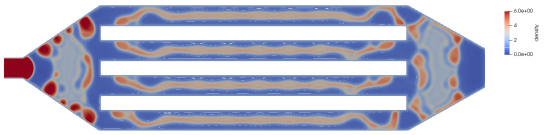


Figure 3.15: Multichannel initial conditions $\rho = \rho_w$ progression $\Delta t = 10e3$

To ensure stable simulations with correct initial conditions, two methods are thought of. The first is to initialize a smooth interface between the wall and fluid. The second is a smooth ramp-up of the wall density from the initial fluid density to the required wall density.

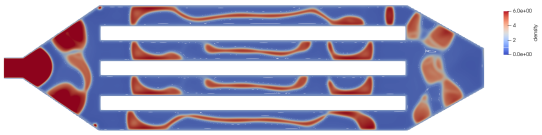


Figure 3.16: Multichannel initial conditions $\rho = \rho_w$ progression $\Delta t = 20e3$

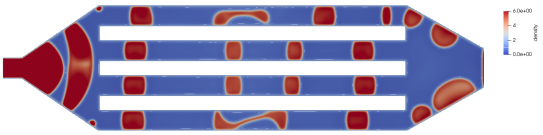


Figure 3.17: Multichannel initial conditions $\rho = \rho_w$ progression $\Delta t = 30e3$

The simplest implementation is the changing wall density. However, OpenLB's `BounceBack` class does not feature a function to change the density once set. Therefore a simple function, `setDensity` is added which overwrites the previous wall density, with appendix E.1 giving the details regarding the implementation. A start regarding the smooth density gradient can be found from the discussion found here: https://www.openlb.net/forum/topic/indicatorlayer2d-usage/.

This variable wall density allows updating the wall density over the first couple hundred timesteps of the simulation. This enabled the simulation of wall wettability at high fictitious wall densities without adding density to the vapour phase so that the simulation is stable, which is shown in section 4.4.

## 3.11. OpenLB fixes

While working with the OpenLB source code, a few issues were encountered which were corrected. They can be grouped into bugs and missing features. The various fixes are presented below.

### 3.11.1. Bugs

Bugs are errors in existing features that have been released. Three bugs have been identified and reported via the forum. For each bug report, the bug was acknowledged by the developers and was stated that it will be fixed in the upcoming release of OpenLB (current version is 1.3-1). The bugs found are presented as follows: the url to the bug report, a short version of the bug, and how this bug would affect the simulations performed in this thesis.

A quick update regarding bug fixes: In OpenLB v1.4 [62] (released November 18$^{th}$ 2020) the reported bugs were all corrected. In addition, a new bug was found after the release, the D3Q7 MRT collision matrix bug, which is of course not yet fixed. However, it was acknowledged as a bug and it should be fixed in the next release.

#### D3Q7 equilibrium calculation error

https://www.openlb.net/forum/topic/missing-dimension-in-c_u-calcuation-d3q7/

In the `dynamics/lbHelpersD3Q7.h` file, in the function `equilibrium` the `c_u` is calculated as follows:

```
T c_u = descriptors::c<L>(iPop,0)*u[0] + descriptors::c<L>(iPop,1)*u[1];
```

The third dimension is missing in that calculation. It should be as in the below function `equilibriumFirstOrder`:

```
T c_u = descriptors::c<L>(iPop,0)*u[0] + descriptors::c<L>(iPop,1)*u[1] +
    ↪ descriptors::c<L>(iPop,2)*u[2];
```

When using the D3Q7 lattice, which is done in this thesis for the 3D thermal lattice, the equilibrium distribution function and by extension the simulations fail. However, by default the BGK version uses the `equilibriumFirstOrder` and only when using the MRT collision operator does this bug become important.

### Sourced advection-diffusion dynamics density calculation error

https://www.openlb.net/forum/topic/density-calculation-error-in-sourced-ad-bgk-dynamics-computerhou/

In the file `src/dynamics/advectionDiffusionDynamics.hh` for the class `SourcedAdvectionDiffusionBGKdynamics` in the function `computeRhoU`. The density is calculated as follows:

```
{
    this->_momenta.computeRhoU( cell, rho, u );
    const T* source = cell.template getFieldPointer<descriptors::SOURCE>();
    rho += 0.5 * source[0] * _omegaMod;
}
```

In the function `computeRho` of the same class, the density is calculated as:

```
{
    const T* source = cell.template getFieldPointer<descriptors::SOURCE>();
    return this->_momenta.computeRho( cell ) + 0.5 * source[0];
}
```

The density calculation of `computeRho` is correct, while in `computeRhoU` the density has an erroneous `*_omegaMod` term.

This bug is also affecting the thermal lattice. This time the custom thermal dynamics written to include phase change sources is based on the `SourcedAdvectionDiffusionBGKdynamics`. As such, if not corrected, the simulations would be faulty.

### Minimum value finder not working in 2D

https://www.openlb.net/forum/topic/supermin2d-not-finding-minimum/

In file `superLatticeIntegralF2D.hh` the function `SuperMin2D` is not working. Using this function results in a return of 0.

The initial minimum is set with the output as 0: (line 96) `output[i]=T();` This is only updated when the absolute value of another minimum is lower than the initial minimum

```
if (fabs(outputTmp[i]) < output[i]) {
    output[i] = fabs(outputTmp[i]);
}
```

This basically reads as: `if( abs(x) < 0) update` , which is never true.

In the `SuperMin3D` function the output is initialized differently `output[i] = std::numeric_limits<W>::max();`. And that minimization works. Replacing `SuperMin2D` initialization with `SuperMin3D` makes the function work as expected.

For the verification tests the minimum density, i.e. pressure, was often recorded. As such this bug resulted in erroneous results evaluations.

### D3Q7 MRT collision matrix velocity set error

https://www.openlb.net/forum/topic/d3q7-mrt-velocity-directions/

The file `mrtLatticeDescriptors.h` contains all the MRT collision matrix information. That includes the matrix $\mathbf{M}$, $\mathbf{M}^{-1}$, and the number and locations of the shear indices. As per their definition listing 3.12, they change the velocity directions for optimization purposes.

Listing 3.12: "OpenLB definition for velocity directions"

```
/// Descriptors for the 2D and 3D lattices.
/** \warning Attention: The lattice directions must always be ordered in
* such a way that c[i] = -c[i+(q-1)/2] for i=1..(q-1)/2, and c[0] = 0 must
* be the rest velocity. Furthermore, the velocities c[i] for i=1..(q-1)/2
* must verify
*  - in 2D: (c[i][0]<0) || (c[i][0]==0 && c[i][1]<0)
*  - in 3D: (c[i][0]<0) || (c[i][0]==0 && c[i][1]<0)
*                       || (c[i][0]==0 && c[i][1]==0 && c[i][2]<0)
* Otherwise some of the code will work erroneously, because the
* aformentioned relations are taken as given to enable a few
* optimizations.
*/
```

For the D3Q7, the collision matrix is defined by listing 3.13, which does not conform to their velocity direction. The proper definition of the matrix is given in listing 3.14, as stated in the bug report by the developers.

Listing 3.13: "OpenLV v1.4 D3Q7 M matrix"

```
template <>
constexpr Fraction M<3,7>[7][7] = {
    //  Li, Yang et al 2016: The directions are modified for the OpenLB definition
    {1,  1,  1,  1,  1,  1,  1},
    {0,  1,  0, -1,  0,  0,  0},
    {0,  0, -1,  0,  0,  1,  0},
    {0,  0,  0,  0,  1,  0, -1},
    {6, -1, -1, -1, -1, -1, -1},
    {0,  2, -1,  2, -1, -1, -1},
    {0,  0,  1,  0, -1,  1, -1}
};
```

Listing 3.14: "OpenLV proper D3Q7 M matrix"

```
template <>
constexpr Fraction M<3,7>[7][7] = {
    {1,  1,  1,  1,  1,  1,  1},
    {0, -1,  0,  0,  1,  0,  0},
    {0,  0, -1,  0,  0,  1,  0},
    {0,  0,  0, -1,  0,  0,  1},
    {6, -1, -1, -1, -1, -1, -1},
    {0,  2, -1, -1,  2, -1, -1},
    {0,  0,  1, -1,  0,  1, -1}
};
```

The impact this has is that the shear index locations are off compared to the $\mathbf{M}^{-1}$. This makes the D3Q7 matrix currently unusable in its current state. Therefore, **any thermal simulations in 3D should use the BGK collision operator**.

This can be seen when comparing the BGK and MRT simulations of the 3D $D^2$ law verification test. The significance is shown in fig. 3.18, where the left half is the BGK simulation and the right half the MRT.

The solution would be to overwrite OpenLB's current $\mathbf{M}$ matrix, and calculate the inverse $\mathbf{M}^{-1}$ via a software package and overwriting that matrix as well.

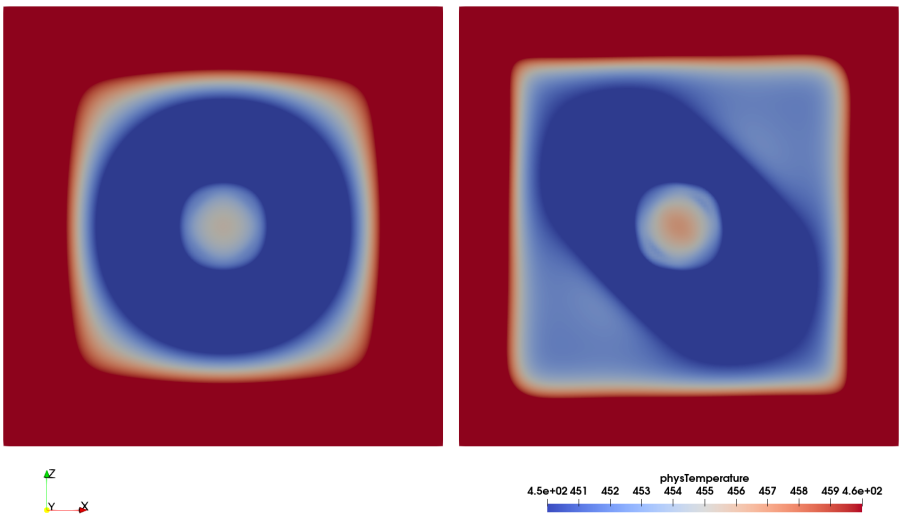



Figure 3.18: Impact of the D3Q7 MRT collision matrix bug on simulations (left: BGK, right: MRT)

### 3.11.2. Missing features

Missing features are features that do not explicitly exist, but an implementation similar to the feature does exist elsewhere in the OpenLB framework.

https://www.openlb.net/forum/topic/is-there-a-superaverage2d/

A missing feature is the `SuperAverage2D` function, which computes the average tracked value across a material. This feature exists for its three dimensional counterpart `SuperAverage3D`. This missing feature was implemented by copying the relevant 3D classes and removing the third dimension. This resulted in the files `superAverage2D.h` & `.hh`, `blockAverage2D.h` & `.hh` and `blockIntegral2D.h` & `.hh`. The `SuperAverage2D` function is used in the output definition of 2D channel flows.

Another missing feature is the `write` function of the `ThermalUnitConverter` class, which does exist for the `UnitConverter` class. This function was added in the file `miscFeatures.h` and is used to create a file which documents the conversion magnitudes used in the simulation.

A quick update regarding the missing features. In OpenLB v1.4 (released November 18$^{th}$ 2020) the mentioned features are not available.

## 3.12. Summary LBM implementation

In this section, a short summary is given for this chapter. The extensions to the OpenLB framework is summarized, with a short overview what the limitations were and how they were overcome. Then a short discussion regarding the boundary methods are given. The summary is concluded with a short reflection regarding the applicability of LBM to the problem.

### 3.12.1. OpenLB extension

The OpenLB simulation library was successfully extended with a several features, which are summarized below:

- A modified Guo forcing BGK dynamics, including custom coupling
- An extended interaction potential class

- Maxwell construction calculation class
- A thermal DDF including latent heat which works in combination with the pseudopotential
- Extending the thermal DDF with a hybrid source term
- A fluid properties helper class
- Proper gravity coupling
- Variable wall wettability

### Modified forcing scheme

Using only the original forcing schemes, there were several limitations. The Guo forcing algorithm is ill suited to combine with the pseudopotential method, resulting in crashes below a temperature ratio of around 0.8. Other forcing schemes, such as the Shan-Chen and exact difference method, can simulate lower temperature ratios up to around 0.5, depending on the EoS used. Their forcing term depends on the relaxation time, thus restricting them to 0.9 to 1.1 for adequate performance, which limits viscosity choice. Even then, the resultant simulations are not thermodynamic consistent, and thus the simulation densities diverged from theoretical values.

The modified Guo forcing BGK dynamics was needed to remedy those limitations; ensure thermodynamic consistency and make simulations independent of the relaxation time. With the modified Guo forcing scheme, high density ratios, unachievable before, can be simulated, while remaining independent of the relaxation time and ensuring thermodynamic consistency. However, the modified Guo forcing scheme contains a constant that needs to be numerically determined. A different equation of state, and different values used for that equation of state, changes that constant.

### Thermal implementation

The original thermal implementation of OpenLB was intended for small temperature changes where the Boussinesq approximation is valid. The core thermal coupling, without the Boussinesq part, can be used to simulate the temperature flow. However, the recovered macroscopic thermal equation is not fully correct, and includes an error term, $T\mathbf{u}$, scaling with velocity.

There are two stages to the implementation that was performed in this thesis. The first stage implemented the latent heat, and corrected for the $T\mathbf{u}$ error term. The correction and latent heat are introduced to the thermal simulation via a source term. This resulted in a fully functional thermal multiphase LBM DDF simulation. However, this implementation has the limitation that the thermal diffusivity is constant across the simulation domain. In addition, the diffusivity is limited to be in the same range as the viscosity, since the same lattice spacing and timestep is used.

The second stage was to remedy the thermal diffusivity deficiencies. For that, the source term is modified to subtract the LBM diffusivity and adding a custom diffusivity. The custom diffusivity is calculated based on the local density. The source term contribution is then calculated via the central difference method. This effectively introduces an explicit Euler scheme via the source term, i.e. a semi-hybrid scheme. This allows for choosing a greater range of diffusivity, but the classical checkerboard instability is still limiting the choice.

### Remaining implementations

For the calculation of the Maxwell construction and equilibrium densities, a hook into the equation of state pressure is needed. The latent heat calculations requires the derivative of the equation of state pressure with respect to the temperature at a constant volume. This exceeds the functionality of the given OpenLB interaction potential class. Thus these functionalities were added in an extended interaction potential class.

The thermal conductivity, and specific heat capacity are different for vapour and liquid. To store and calculate these properties compactly, a class to handle those calls are created. This class also contains an efficient calculation method for the temperature ratio.

The gravity implementation of OpenLB was based on the Boussinesq approximation, which is no longer needed, since the density difference due to temperature are now simulated directly. Implementation of the gravity model is therefore very straight forward.

For the wall wettability the wall has a fictitious density. If the initial domain density and the wall density are too different, it may cause instabilities and crashes. As such, a function to change the wall density was added. At simulation start, the density slowly changes from the surrounding to the wanted wall density. This avoids the instabilities and therefore allows for a wider range of initial conditions.

### 3.12.2. Boundary methods

Little was added to the boundary methods in OpenLB. The limitations of the available boundary conditions are explored, which are quite severe for multiphase flows. That is however a flaw of the LBM, and not a fault of the OpenLB. In fact, little was found in literature regarding proper handling of multiphase flow, where similar channel simulations either accepted the inaccuracies, or only simulated up to the point where only single phase flow exists the domain.

The best method found for the inlet is to manually set the density and velocity, and adding custom coupling to the thermal lattice. The inlet suffers from a compressive effect due to the wall wettability, and mitigation methods are explored, but not implemented.

The outlet has difficulty dealing with multiphase flow, where the convective boundary handles it the best. That type of boundary however leaves little over the behaviour of the simulation domain. Pressure boundaries are unsuited, and velocity based boundaries can be used, if the inaccuracies of suction and bridging are acceptable.

### 3.12.3. Reflection OpenLB

At the start of the thesis, it was assumed that since OpenLB already has multiphase flow via the pseudopotential method and thermal flow simulation, the combination would be relatively straight forward, with only the phase change needed to be added. However, it turned out different than anticipated.

The approach to OpenLB was to learn by doing. Starting with single channel flow and building up the complexity to the multiphase thermal multichannel problem. This allowed tackling issues regarding the thermodynamic inconsistency separate, but in parallel, from the thermal model issue of the erroneous paper which was used.

In hindsight, OpenLB did not have the feature set necessary to simulate phase change multiphase flow. However, the complexity of adding such functionality is also not that difficult, if one knows what to do. The implementation of the models detailed in this chapter were quite straight forward, if some pitfalls would have been avoided.

A few things which would have been helpful is to know is how the units in OpenLB are handled. For example, the force in OpenLB is saved as acceleration, whereas in papers it usually as force density. Both call it force however, which may cause some confusion. In addition, the thermal handling in OpenLB requires special attention, and correct handling is explained in section 5.3.

The mismatch of simulation framework and paper, as well as that papers are also sometimes just wrong, requires more than just surface knowledge. Thus this thesis requires in depth knowledge in both the lattice Boltzmann method and OpenLB, such that methods proposed in papers are verified, adapted to OpenLB. This means that a large part of this, and potential future thesis, should be allocated to truly and fundamentally understanding LBM and OpenLB.

The approach taken, learn by doing, is most likely still the correct approach. It allows learning both LBM and OpenLB. However, in addition, more theoretical studying portions should be added. For example, after simulating a few cases with the pseudopotential method, the Chapman-Enskog analysis of the pseudopotential method should be performed to understand the error sources and how they are corrected for. Doing this

would have prevented the thermal bug discussed in section 5.3.

Using LBM for the problem at hand was not inherently a problem. The computational speed, ease of the geometry creation, as well as simulation stability for multiphase flow went without major issues. The biggest unsolved issue is the inlet and outlet boundary handling. In addition, the handling of the mismatch between fluid and thermal diffusivity can be improved. A practical issue that needs to be solved is the metastability of the pseudopotential method.

# 4 Verification Isothermal LBM

This chapter details the verification of the simulations, focusing on the isothermal aspects. The verification was split for better readability. The isothermal verification cases detailed in this chapter are: the Maxwell construction, the thermodynamic consistency, Young-Laplace law, and wall wettability.

## 4.1. Maxwell construction verification

The Maxwell construction verification is done in three steps. Initially the simple Maxwell construction example code written in python is verified against other examples from literature. Then the more complex python implementation is compared to the simple implementation. Then the c++ code implementation is compared to the previous results. All of these should give the same equilibrium densities.

For the more complex implementation, the python as well as the c++ implementation requires the additional verification of the pressure derivative with respect to temperature at constant density. This derivative has been written in an analytical form, and as such a numerical derivative of the pressure function suffices to verify the analytical derivative.

Since the Maxwell construction and derivatives are physical, without simulations needed, the only error sources should originate from floating point errors, root finding accuracy and integration accuracy. Since both python and c++ use double precision, any difference should be very minimal, such that when plotted, the results should be perfectly overlapping.

### Simple python implementation

The densities generated by the simple Maxwell construction example code listing 3.3, see fig. 4.2, written in python is compared to the Maxwell construction presented by Li et al [78], see fig. 4.1. Since the Maxwell construction is not impacted by the simulation, the simulation parameters used by them are not included, the equation of state parameters used are shown. The lowest temperature ratio shown by them is 0.49. The generated densities for both vapour and liquid branch match up, and thus the code is considered verified.

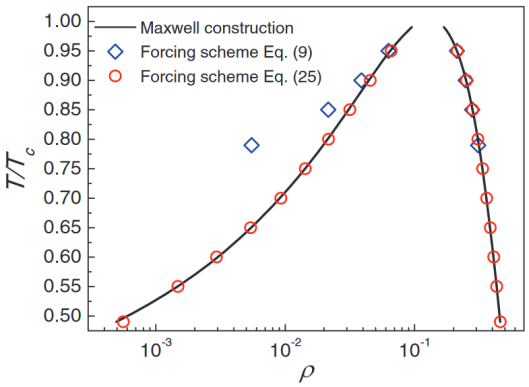



Figure 4.1: Maxwell construction Carnahan-Starling equation of state with a=0.5, b=4, R=1 taken from Li. et al [78]

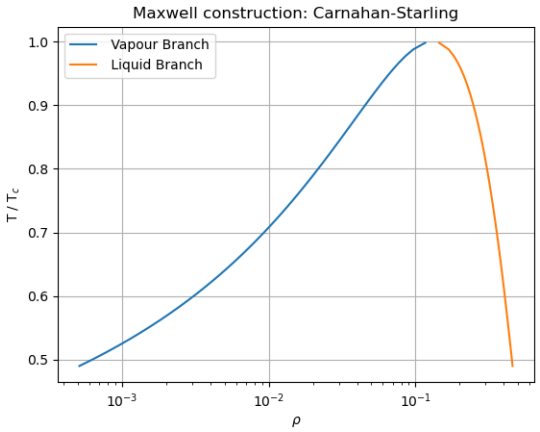



Figure 4.2: Maxwell construction Carnahan-Starling equation of state with a=0.5, b=4, R=1 generated from `ExampleMaxwellConstruction.py`

## Complex python implementation

The code in the c++ is object oriented and as an intermedium step to implementation, the simple python code is put into a class framework, which can also encompass any features that may need to be added. The implementation is verified by comparing the Maxwell construction of the simple code, see fig. 4.2 with the Maxwell construction of the complex code, see fig. 4.3. They match up as expected, showing that the class framework is working as intended.

Due to the class framework, it is possible to use different pressure functions, i.e. different equation of states see section 2.3.2, while the exact same code is used for the Maxwell construction. This means that any additional Maxwell constructions based on that class are verified. For example, the Peng Robinson EoS which is shown in fig. 4.4.

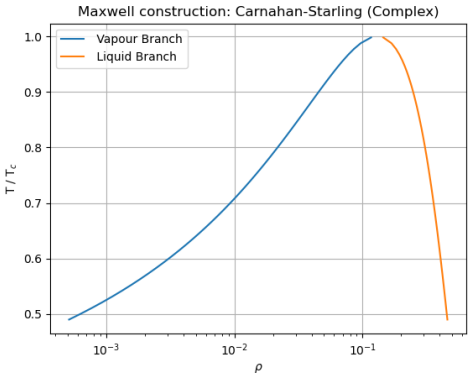



Figure 4.3: Maxwell construction Carnahan-Starling equation of state with a=0.5, b=4, R=1 generated from complex implementation `eos.py`

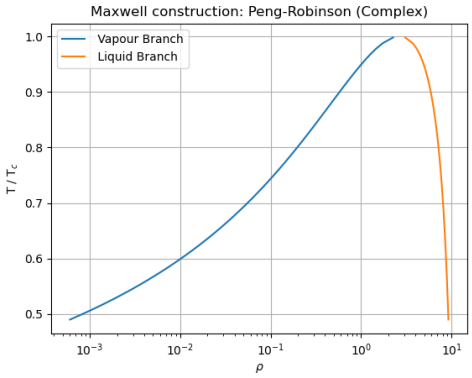



Figure 4.4: Maxwell construction Peng-Robinson equation of state with a=2.0/49, b=2/21, $\omega_{EoS} = 0.3443$, R=1 generated from complex implementation `eos.py`

## OpenLB InteractionPotential implementation

The code in `maxwellConstruction.cpp` calculates and saves the equilibrium densities using the c++ code. The saved data is overlaid onto the densities generated by the python code. The resulting match up for both the CS and PR EoS are shown in fig. 4.5 for the CS EoS and fig. 4.6 for the PR EoS. The resultant densities agree perfectly, meaning that the c++ implementation is verified.

To highlight, fig. 4.7 is shown, showing the difference between the python and c++ implementation. The erratic error rate around the zero point shows that there is no systematic error between the two implementation. The error difference increasing for higher temperature ratio is due to the smaller areas in involved at high temperature ratios, thus causing small differences in the root finding leading to larger differences in the area calculation.

## Pressure derivative implementation

In the thermal phase change algorithm, there is a term containing the pressure derivative with respect to temperature at constant volume. This derivative is implemented analytically to save computational costs. To verify that the analytical implementation is correct, it is compared to its numerical derivative. The derivatives for the CS and PR EoS both at a constant $\rho = 0.1$ is shown in fig. 4.8 and fig. 4.9 respectively.

The derivatives of the c++ implementation is compared to the derivatives of the verified python code. The resultant comparisons are shown in fig. 4.10 and fig. 4.11. As can be seen, the c++ and python derivatives agree perfectly.

With this verification successfully concluded, the Maxwell construction module of both c++ and python is correctly implemented.

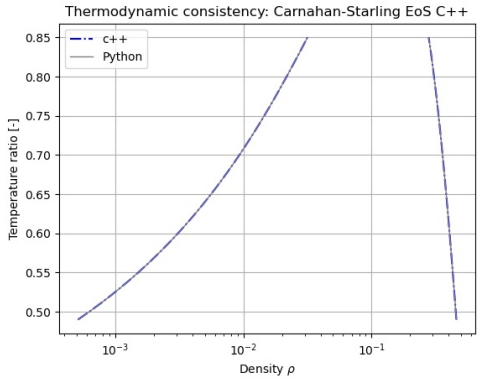



Figure 4.5: Maxwell construction Carnahan-Starling equation of state with a=0.5, b=4, R=1 generated from c++ implementation

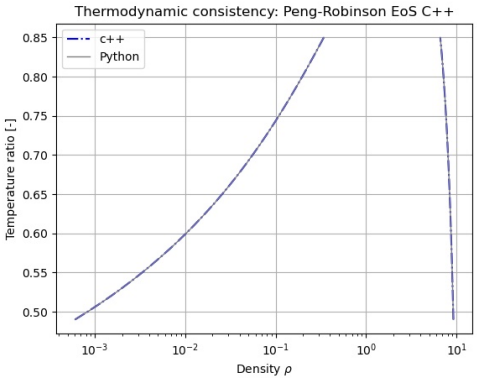



Figure 4.6: Maxwell construction Peng-Robinson equation of state with a=0.5/49, b=2/21, $\omega_{EoS}$ = 0.3443, R=1 generated from c++ implementation

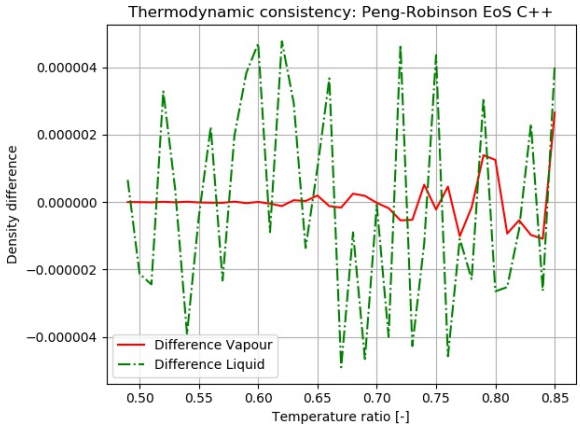



Figure 4.7: Maxwell construction Peng-Robinson equation of state with a=0.5, b=2/21, $\omega_{EoS}$ = 0.3443, R=1 difference between python and c++ implementation

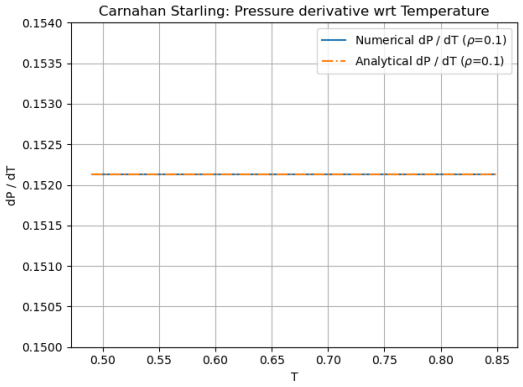



Figure 4.8: Analytical and numerical pressure derivative Carnahan-Starling equation of state with a=0.5, b=4, R=1

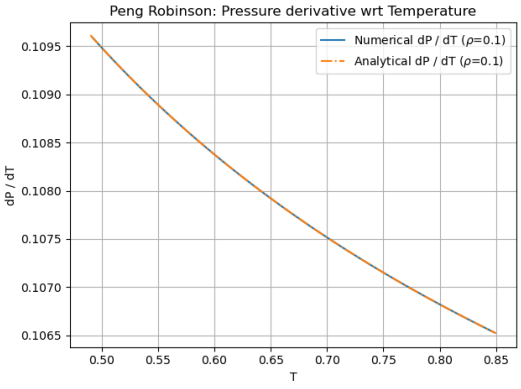



Figure 4.9: Analytical and numerical pressure derivative of Peng-Robinson equation of state with a=0.5/49, b=2/21, $\omega_{EoS}$ = 0.3443, R=1

## 4.2. Thermodynamic consistency verification

The importance of the thermodynamic consistency of the simulations when using the pseudopotential method was previously discussed in section 2.3.1. For a simulation to be thermodynamically consistent, the resultant

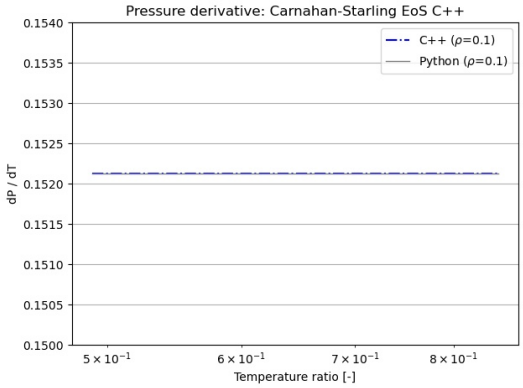



Figure 4.10: Pressure derivative of Carnahan-Starling equation of state with a=0.5, b=4, R=1

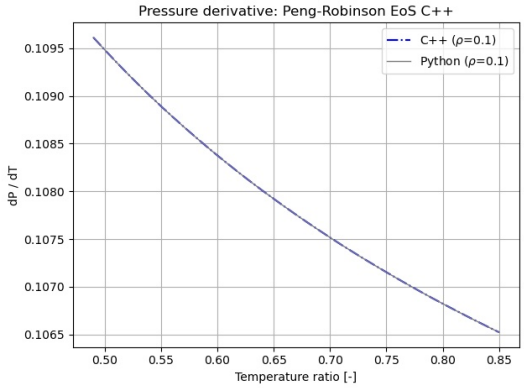



Figure 4.11: Pressure derivative of Peng-Robinson equation of state with a=0.5/49, b=2/21, $\omega_{EoS} = 0.3443$, R=1

densities must agree with the predicted densities by the Maxwell construction, see section 2.3.3. To enable the simulations to do so, a new forcing scheme is used, based on the theory of section 2.3.4 and implemented as shown in section 3.3.

This thermodynamic consistency test verifies both the correct implementation of the new forcing scheme and thermodynamically consistent behaviour. The test results are compared to the predicted values given by the Maxwell construction, which is in turn verified by comparison to literature.

## Simulation setup

For the simulations, a domain of $L \times L \times L$ is set up with periodic boundaries with the constant simulations parameters as listed in table 4.1. The resultant geometry for 2D is shown in fig. 4.12, with a zoomed in section shown in fig. 4.13. Since the grid is uniform, with periodic boundary, there are no ghost nodes (geometry = 0), nor walls (geometry = 2). The remaining nodes are fluid nodes (geometry = 1) on a regular grid. While the 3D geometry has fewer nodes in a direction, the total nodes are more. The distance between two nodes is the characteristic length divided by the resolution, $\delta x = L/N$.

The characteristic length is used to convert from simulation back to physical/real units. The spacial and temporal resolution, relaxation time, characteristic length and speed, vapour density multiplier, and maximum timesteps were found from previous simulations to be a good setup parameters. The characteristic speed is given in place of characteristic time since it allows better understanding of what maximum allowable simulation speed is possible without breaking any assumptions.

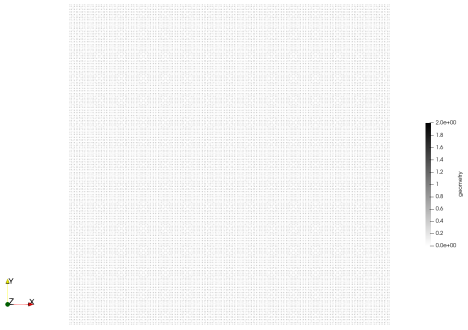


Figure 4.12: 2D consistency geometry

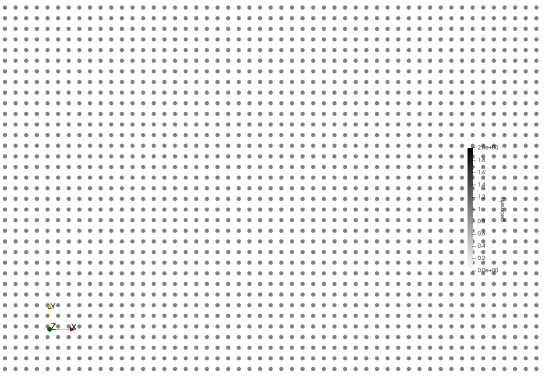


Figure 4.13: 2D consistency geometry zoomed in section

A sphere of radius $R = L/6$ with an interface width of $R/4$ is set up in the centre with the expected liquid density. The surrounding is filled with vapour, multiplied with a factor, and random noise of 10% of local

density is added to avoid crashes due to oscillations. The Peng-Robinson equation of state is used for the pseudopotential with the commonly used non-dimensionalized parameters, see section 2.3.2. BGK dynamics in combination with the modified Guo forcing scheme is used. The parameter $\sigma$ used in the forcing scheme needs to be determined numerically.

The outputs of the simulation are the vapour and liquid densities which are taken as the minimum and maximum values once the sphere has come to equilibrium after a certain time period. The parameter that is varied is the temperature ratio of the system.

Table 4.1: Thermodynamic consistency simulation setup parameters

| **Parameter** | **Symbol** | **Value 3D** | **Value 2D** | **Unit** |
|---|---|---|---|---|
| Spatial resolution | N | 75 | 200 | - |
| Temporal resolution | $N_t$ | $7N$ | $7N$ | - |
| Relaxation time | $\tau$ | 0.8 | 0.8 | - |
| Characteristic length | L | $100 \cdot 10^{-6}$ | $100 \cdot 10^{-6}$ | $m$ |
| Characteristic speed | U | 10 | 30 | $m \cdot s^{-1}$ |
| Physical kinematic viscosity | $\nu$ | $1 \cdot 10^{-5}$ | $1 \cdot 10^{-5}$ | $m^2 \cdot s^{-1}$ |
| Physical density | $\rho$ | 1 | 1 | $kg \cdot m^{-3}$ |
| Vapour density multiplier | | 2 | 1.1 | - |
| Maximum timesteps | | $10 \cdot 10^3$ | $20 \cdot 10^3$ | $\Delta t$ |
| EoS parameter a | $a$ | $\frac{2}{49}$ | $\frac{2}{49}$ | - |
| EoS parameter b | $b$ | $\frac{2}{21}$ | $\frac{2}{21}$ | - |
| EoS acentric factor | $\omega_{EoS}$ | 0.3443 | 0.3443 | - |

## Achieving thermodynamic consistency

At first, the same simulation values as used by Li et al. [77] who introduced the method were used. Using the values of $\sigma = 0.105$ resulted in unstable simulations. In section 2.3.4, it was mentioned that there may be a printing error which may result in an offset of a factor $\frac{1}{c_s^2} = 3$ to the parameter $\sigma$. Following Zhang et al. [153], $\sigma = 0.3$ was used, which resulted in stable simulations.

At lower temperature ratio, the interface became unstable due to high density ratios. As such the same approach of Xu et al. [148] was used to stabilize the simulations. A lower $a$ value was used in the equation of state to spread the interface over multiple nodes [78]. According to the equation of state, this has little impact on the predicted densities, however it was found that at lower $a$ values, thermodynamic consistency diverges, which can be partly corrected by adapting the $\sigma$ parameter.

The impact of $a$ and $\sigma$ was studied on the thermodynamic consistency over various values. A few examples of the final simulations state are given below. Afterwards the results of the impact analysis of $a$ and $\sigma$ will be presented.

## Simulation results

The final state of the droplet for the simulation series of $\sigma = 0.3$, a temperature ratio of 0.8, with a varying parameter $a$ is shown in figs. 4.14 to 4.17 for the 3D simulations. A similar series of $\sigma = 0.31$ is excluded since visually there is little difference.

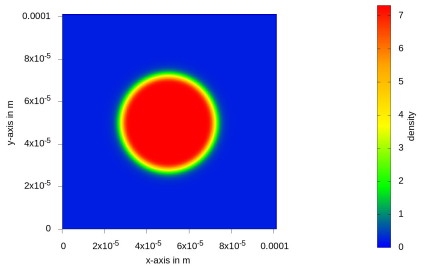



Figure 4.14: 3D thermodynamic consistency result, slice in the z-axis halfway through domain, $\sigma = 0.3$, $Tr = 0.8$, $a = 2.0/49$

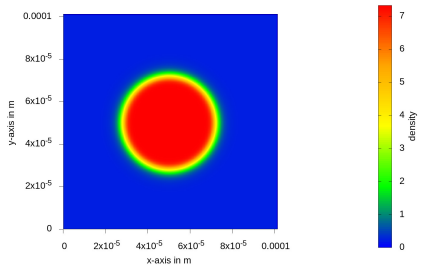



Figure 4.15: 3D thermodynamic consistency result, slice in the z-axis halfway through domain, $\sigma = 0.3$, $Tr = 0.8$, $a = 1.5/49$

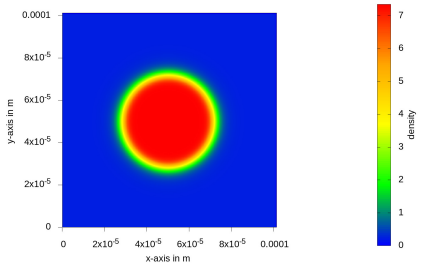



Figure 4.16: 3D thermodynamic consistency result, slice in the z-axis halfway through domain, $\sigma = 0.3$, $Tr = 0.8$, $a = 1.0/49$

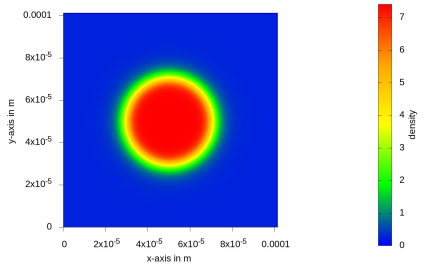



Figure 4.17: 3D thermodynamic consistency result, slice in the z-axis halfway through domain, $\sigma = 0.3$, $Tr = 0.8$, $a = 0.5/49$

During the initial few thousand timesteps the droplet oscillates due to imperfect interface initialization. This can be seen in fig. 4.18 and fig. 4.19 in the maximum density, where the plots are generated by the simulation and little control is given for the layout. They are more pronounced for 2D simulations than 3D simulations, since increasing the surrounding vapour density has a dampening effect on the oscillations. A more technical analysis on droplet oscillations was done by Xu et al. [148].

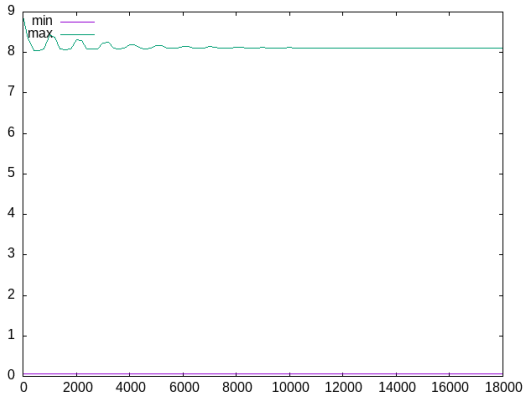



Figure 4.18: Plot contains non-dimensional density (y-axis) vs timesteps (x-axis): Showing droplet oscillations for 2D $\sigma = 0.35$ and $a = 0.5/49$ at a $Tr = 0.7$

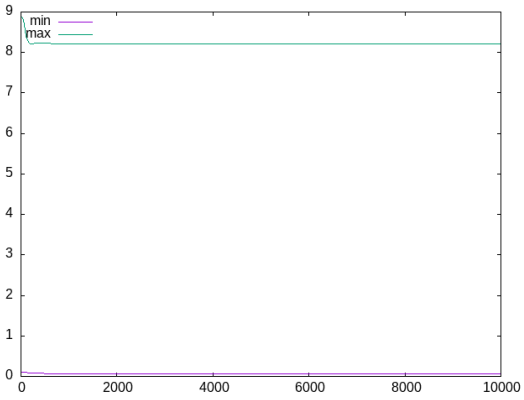



Figure 4.19: Plot contains non-dimensional density (y-axis) vs timesteps (x-axis): Showing droplet oscillations for 3D $\sigma = 0.31$ and $a = 0.5/49$ at a $Tr = 0.7$

## Impact of variables

As Li et al. [77] indicated, the correct value of $\sigma$ needs to be determined numerically. But it is found that $a$ has an impact on the correct value of $\sigma$. To study the impact of $a$ and $\sigma$ the droplet was simulated with $a$ ranging from $2.0/49$ to $0.25/49$ in decreasing steps of $0.25/49$, at various $\sigma$ values. Of interest are only two values, $\sigma$

being equal to 0.3 and 0.31, as will be explained soon.

The first series figs. 4.20 to 4.25 shows the impact of $a$ on the achieved densities and stability of the simulation. At the minimum required temperature ratio of 0.6, which corresponds to 115°$C$ for water, the $a$ value should be 0.5/49. This value is close to the one used by Xu et al. of 1/100 [148].

However, such a low value for $a$ has a drastic effect on the simulation densities, strongly deviating from the expected simulations, especially for the vapour densities. The liquid densities are best shown in figures figs. 4.26 to 4.27, which has some impact. The difference can be best seen in the density ratio, shown in fig. 4.28. To remedy the lower temperature, the parameter $\sigma$ is increased from 0.3 to 0.31. The effect of this change is best seen in the density ratio, shown in fig. 4.29. The impact on the exact vapour and liquid densities is shown in figs. 4.30 to 4.31.

The merger of the two lines for $a = 0.25/49$ in fig. 4.25 indicates that the phase interface became so large that the phase separation collapsed, becoming one mixed fluid. This happens at higher temperature ratios, since then the density ratio is lower and as such there is a smaller force separating the phases.

The results are presented using three different graphs. The traditional P-V plot is given, showing the pressure as a function of specific volume. The vapour curve (right curve) consistency is especially visible, however the liquid curve (left curve) is not. Therefore the same data is shown on the P-$\rho$ plot, showing the inverse of the x axis. This flips the curve location and the liquid curve (right) can be seen better. Lastly, the density ratio plot is used, which provides a different perspective to the accuracy of the thermodynamic consistency.

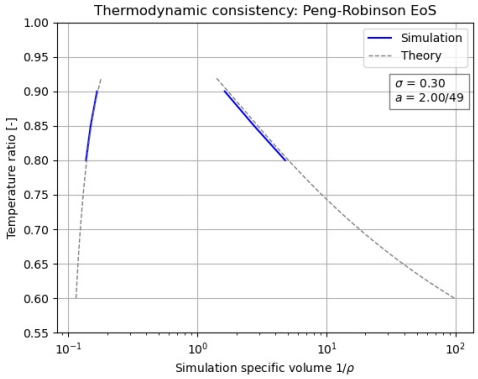



Figure 4.20: P-V plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{2}{49}$

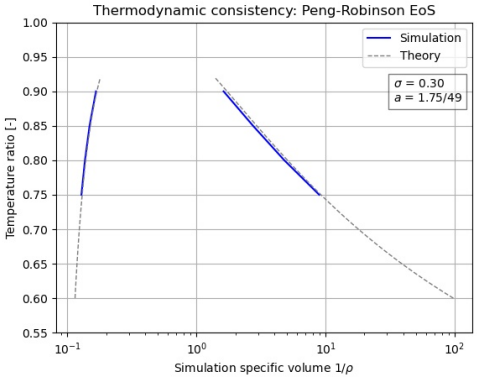



Figure 4.21: P-V plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{1.75}{49}$

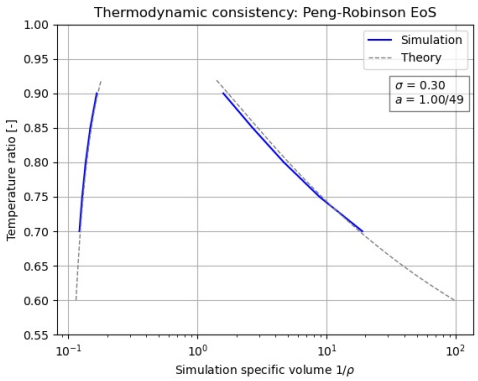



Figure 4.22: P-V plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{1}{49}$

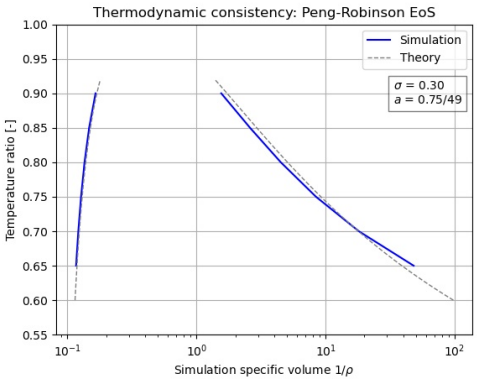



Figure 4.23: P-V plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{0.75}{49}$

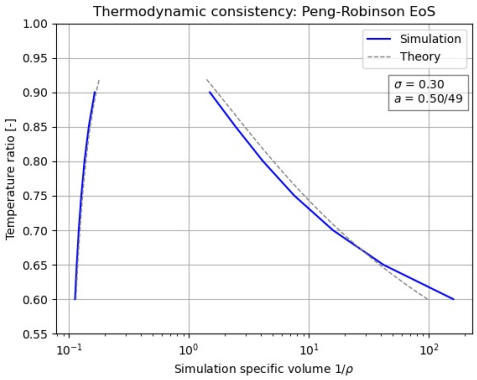



Figure 4.24: P-V plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{0.5}{49}$

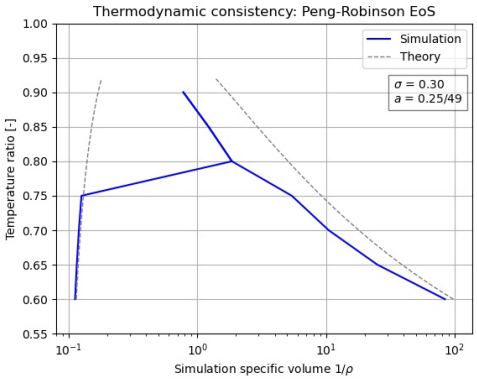



Figure 4.25: P-V plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{0.25}{49}$

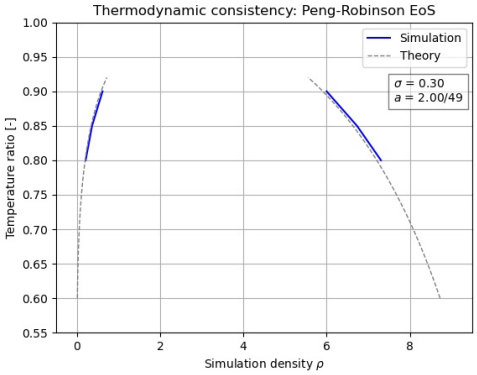



Figure 4.26: P-$\rho$ plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{2}{49}$

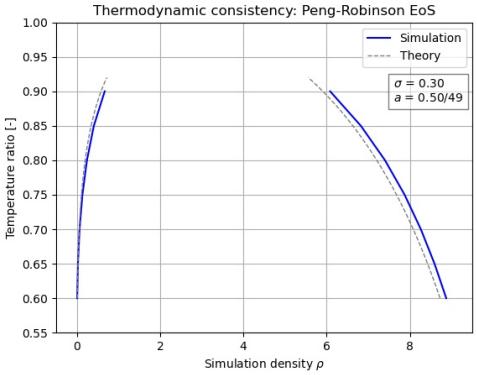



Figure 4.27: P-$\rho$ plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{0.5}{49}$

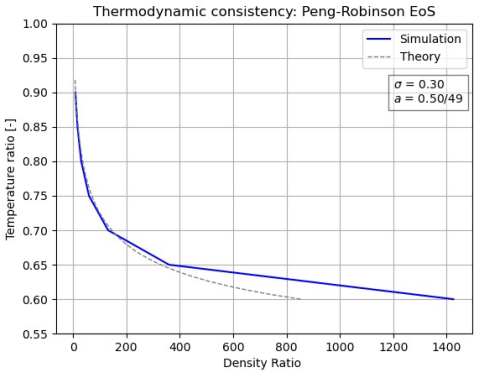



Figure 4.28: Density ratio plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{0.5}{49}$

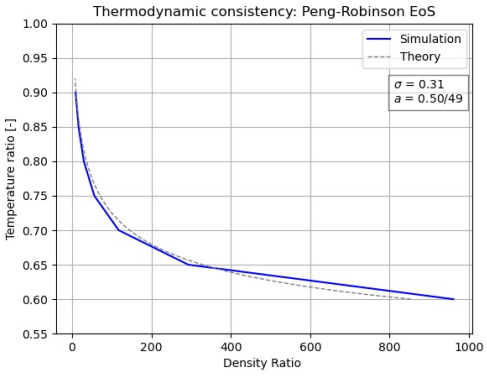



Figure 4.29: Density ratio plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.31$ and $a = \frac{0.5}{49}$

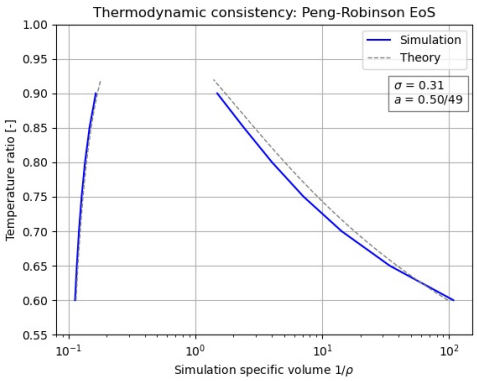



Figure 4.30: P-V plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.31$ and $a = \frac{0.5}{49}$

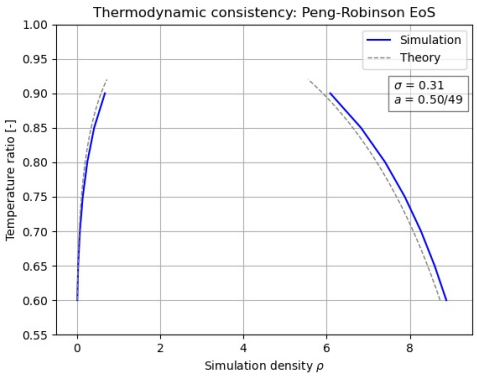



Figure 4.31: P-$\rho$ plot of results from 3D thermodynamic consistency using PR EoS with $\sigma = 0.31$ and $a = \frac{0.5}{49}$

### 4.2.1. Thermodynamic consistency results

After concluding the 3D simulations, the same study was performed for 2D simulations. The figures showing the thermodynamic consistent cases are shown in appendix C. The resultant thermodynamic combinations of $\sigma$& $a$ for both 2D and 3D simulations are summarized in table 4.2, note than the lowest tested temperature ratio is 0.6.

Table 4.2: Thermodynamic consistency valid $\sigma$ and $a$ combinations

| Lattice | $\sigma$ | $a$ | Valid up to Tr |
|---|---|---|---|
| D3Q19 | 0.30 | $\frac{2}{49}$ | 0.8 |
| D3Q19 | 0.30 | $\frac{1}{49}$ | 0.7 |
| D3Q19 | 0.31 | $\frac{1}{49}$ | 0.7 |
| D3Q19 | 0.31 | $\frac{0.5}{49}$ | 0.6 |
| D2Q9 | 0.30 | $\frac{3.0}{49}$ | 0.7 |
| D2Q9 | 0.31 | $\frac{2.5}{49}$ | 0.7 |
| D2Q9 | 0.32 | $\frac{2.0}{49}$ | 0.65 |
| D2Q9 | 0.33 | $\frac{1.5}{49}$ | 0.65 |
| D2Q9 | 0.34 | $\frac{1.0}{49}$ | 0.65 |
| D2Q9 | 0.35 | $\frac{0.5}{49}$ | 0.6 |

Overall, it is possible to achieve thermodynamic consistency using the modified Guo forcing scheme. The values taken for the simulations depend on multiple factors, such as the diffuse phase thickness and lowest expected temperature ratio. There are some inconsistencies with over-predicting the liquid densities, but that may be due to the fact that the liquid is still compressible and the Laplace pressure, see section 4.3, artificially increases the liquid density, as explained in section 4.2.3.

### 4.2.2. Spurious currents around a droplet

One numerical phenomena which is well understood are the spurious currents at the interface [25]. Spurious currents are, as the name implies, erroneous currents with a false origin. They arise due to simulating a curved surface on a square lattice and are not LBM specific.

The final simulation state of the $\sigma = 0.32$ and $a = 2/49$ 2D thermodynamic consistency at a $Tr = 0.7$ simulation is shown in fig. 4.32, where the arrows show the direction and magnitude of the velocity field. The velocity field is better shown in fig. 4.33, where the arrows are no longer sized according to the magnitude, and thus better show the field. The droplet is simulated does not experience any external forces, and is simulated till it reached an equilibrium. From a physics point of view, there should be zero velocities, thus any currents present are spurious.

Several factors determine the magnitude of the spurious currents. The forcing method used, whether BGK, TRT, or MRT dynamics are used, the size of the interface, the relaxation time, and the density ratio all play a factor into this [20, 42, 78, 80, 99, 113, 134, 150]. The simplest method to decrease the spurious currents is to increase the interface width by lowering $a$. A compromise between simulation accuracy and spurious currents is adjusting the relaxation time. The most effective is to use an MRT dynamics, which greatly decreases the currents magnitude increasing simulation accuracy, and only has a light impact in computational cost. The downside is the complexity of the MRT collision operator.

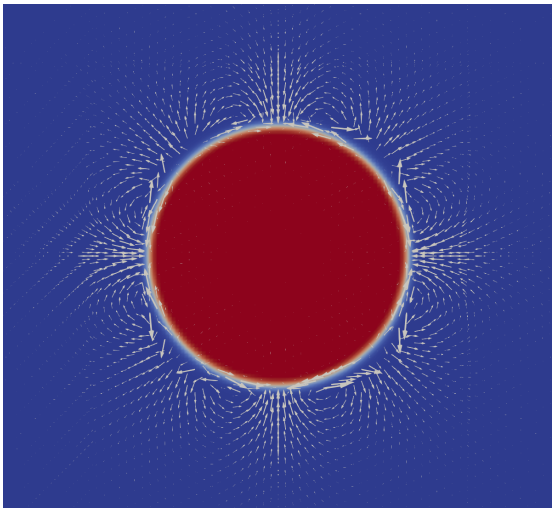


Figure 4.32: Final simulation state showcasing the spurious currents showing the density field. The size of the arrow represents the direction and magnitude of the velocity. Droplet shown is from the 2D thermodynamic consistency case $\sigma = 0.32$ and $a = 2/49$ at $Tr = 0.7$.

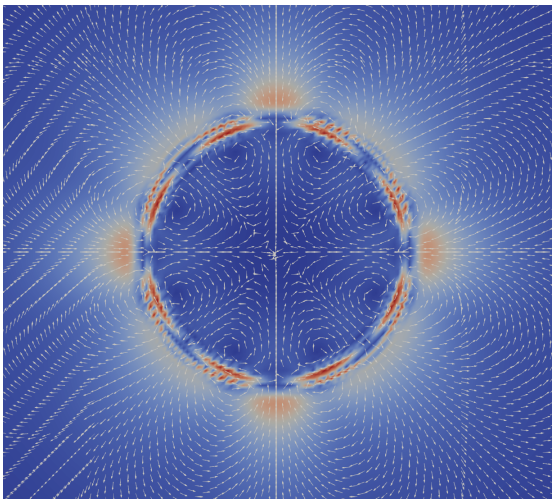


Figure 4.33: Final simulation state showcasing the spurious currents showing the velocity field. The size of the arrow represents the direction of the velocity. Droplet shown is from the 2D thermodynamic consistency case $\sigma = 0.32$ and $a = 2/49$ at $Tr = 0.7$.

### 4.2.3. Expanding crashing

During the initial steps, the droplet expands and contracts due to a non-equilibrium initialization. The expansion of the droplet creates a small low density point in the centre fig. 4.34. In the following contraction the density spikes to well above the thermodynamic consistent density fig. 4.35. An equilibrium initialization of the droplet is difficult to perform due to the internal droplet density gradient, shown in fig. 4.36. The droplet internal density is slightly higher than the thermodynamic consistency due to the Laplace pressure, shown in fig. 4.37. The Laplace pressure verification test is given in section 4.3.

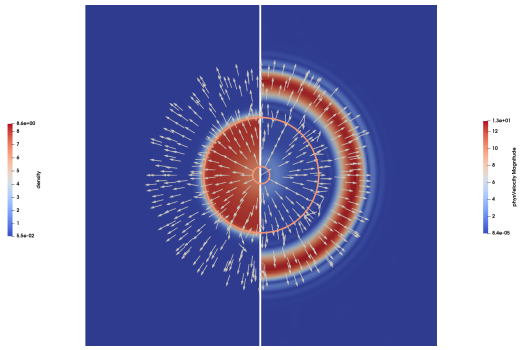


Figure 4.34: Droplet expansion. The droplet forms a low density point in its centre. $\Delta t = 100$

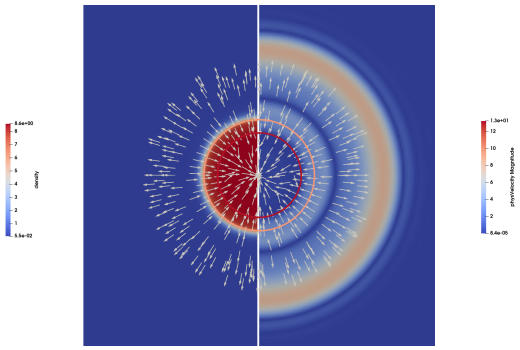


Figure 4.35: Droplet contraction. The density is well above the thermodynamic consistent density of $\rho = 8.08$. This density spike can result in crashes. $\Delta t = 225$

## 4.3. Young-Laplace law

An important part of the simulations is the correct simulation of surface tension. A well known case to verify this is the Young-Laplace law, which gives a relation between the pressure difference across the interface, the interface curvature and the surface tension. For 2D the Young-Laplace equation is shown in eq. (4.1) [13, 43, 113]. Correct simulation of the surface tension would include following the Young-Laplace law as well as recovering the correct surface tension.

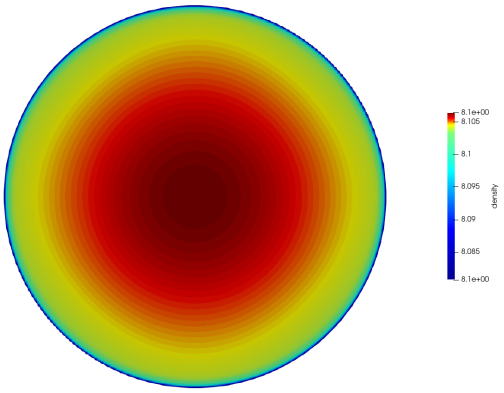


Figure 4.36: High contrast liquid droplet internal density field after simulation convergence. Density below the thermodynamic consistent liquid density is hidden. Colour scheme adjusted such that density gradient is better visible.

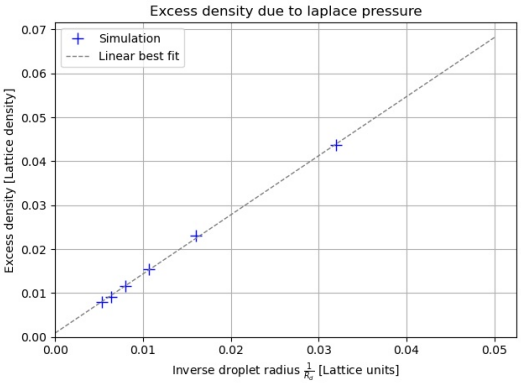



Figure 4.37: Internal droplet excess density compared to thermodynamic equilibrium density for 2D simulations. The excess density scales with $\frac{1}{r}$ indicating a relation to Laplace pressure.

$$\Delta p = \gamma \frac{1}{R} \tag{4.1}$$

A fully periodic domain as in the thermodynamic consistency is used for this test. The domain is enlarged to 5 times the characteristic length for both x and y. The resolution is 125, resulting in a domain size 625x625. The resultant geometry is shown in fig. 4.38, and a zoomed in section in fig. 4.39.

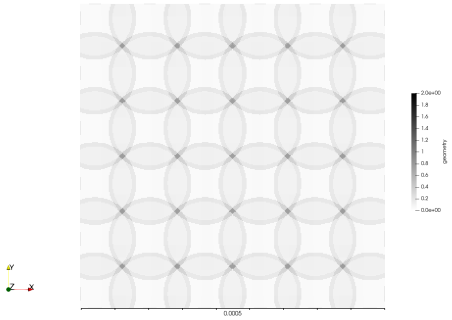


Figure 4.38: Laplace test geometry 2D, complete domain. The circular patterns visible are visual interference due to the high amount of circles needing to be rendered with limited amount of pixels

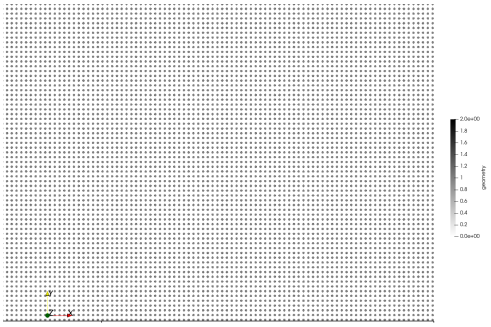


Figure 4.39: Laplace test geometry 2D, bottom right corner, one full tick is visible to indicate how many nodes are within each tick.

A series of simulations are performed of various droplet sizes, ranging from initial diameters equal to 0.1 to 0.6 times the simulation domain. The simulation is stopped if it either reached the maximum timestep or it converged. Convergence is measured using the average kinetic energy and using OpenLB's `ValueTracer` utility. The simulation is said to be converged if the standard deviation of the average kinetic energy over an interval of 150 steps is less than $10^{-5}$. All values used are presented in table 4.3.

The standard pressure calculations in OpenLB is assuming single phase flow ($p = c_s^2 \rho$), but the pseudopotential modifies this pressure term ($p = c_s^2 \rho + 0.5 c_s^2 G \psi^2$), where $G = -1$ is the pseudopotential strength and $\psi$ the local pseudopotential. Thus the pseudopotential is manually calculated and saved during the simulation. How to do that in OpenLB, see appendix E.2.

Table 4.3: Young-Laplace simulation setup parameters

| Parameter | Symbol | Value 2D | Unit |
|---|---|---|---|
| Spatial resolution | N | 125 | - |
| Relaxation time | $\tau$ | 0.8 | - |
| Characteristic length | L | $100 \cdot 10^{-6}$ | $m$ |
| Domain size X | | $5 \cdot L$ | $m$ |
| Domain size Y | | $5 \cdot L$ | $m$ |
| Characteristic speed | U | 15 | $m \cdot s^{-1}$ |
| Physical kinematic viscosity | $\nu$ | $1 \cdot 10^{-5}$ | $m^2 \cdot s^{-1}$ |
| Vapour density multiplier | | 1.1 | - |
| Temperature ratio | $Tr$ | 0.7 | - |
| EoS parameter a | $a$ | $\frac{0.5}{49}$ | - |
| EoS parameter b | $b$ | $\frac{2}{21}$ | - |
| EoS acentric factor | $\omega_{EoS}$ | 0.3443 | - |
| Maximum timesteps | | $1000 \cdot 10^3$ | $\Delta t$ |
| Convergence interval | | 150 | |
| Convergence std deviation | | $1 \cdot 10^{-5}$ | |

## Results

Examples of the final droplet states are given for droplet radius 0.1 and 0.2 times the simulation domain in figs. 4.40 to 4.41. The pressure cross-section is given in fig. 4.42.

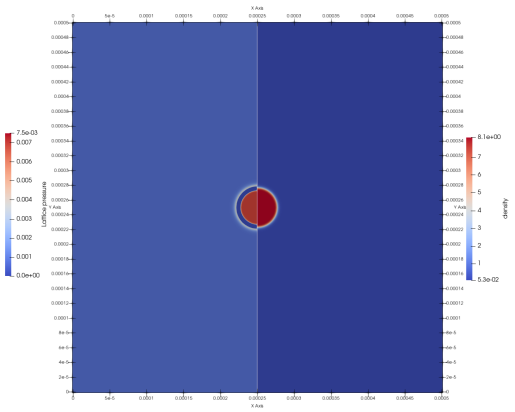


Figure 4.40: Final state of the droplet $R = 0.1$ domain size. Convergence was reached at timestep 95454.

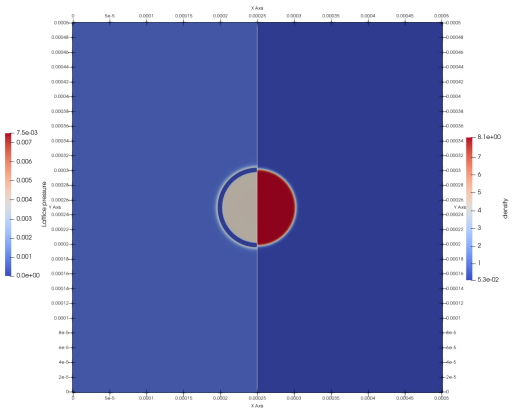


Figure 4.41: Final state of the droplet $R = 0.2$ domain size. Convergence was reached at timestep 73412.

To evaluate the successfully finished simulations, the pressure difference between the edge of the simulation domain and the centre of the droplet needs to be calculated. Since the interface is diffuse, the choice where the droplet begins is a bit arbitrary. For the calculations, $\rho = 4$ is used, which is in the middle of the interface.

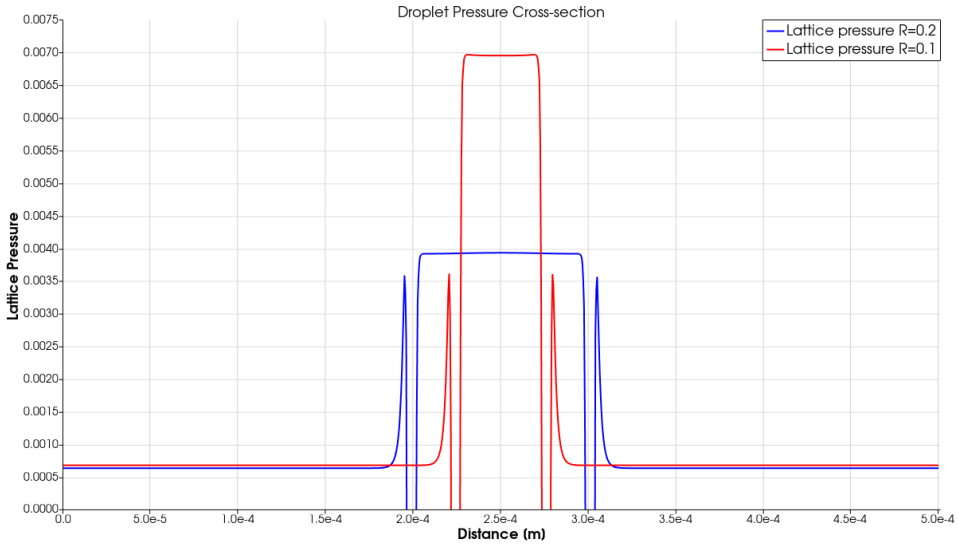



Figure 4.42: Pressure plot along the line y=250$e-6$.

Table 4.4: Results 2D Young-Laplace simulation dimensionless

| **Droplet fraction** | **Droplet size** l.u. | **Liquid pressure** $\cdot 10^3$ | **Vapour pressure** $\cdot 10^3$ | **Pressure difference** $\cdot 10^3$ |
|---|---|---|---|---|
| 0.1 | 34.17 | 6.95537 | 0.681442 | 6.273928 |
| 0.2 | 65.75 | 3.93553 | 0.638102 | 3.297428 |
| 0.3 | 97.45 | 2.85009 | 0.623126 | 2.226964 |
| 0.4 | 129.35 | 2.30233 | 0.615578 | 1.686752 |
| 0.5 | 166.86 | 1.93717 | 0.611031 | 1.326139 |
| 0.6 | 195.44 | 1.76017 | 0.608056 | 1.152114 |

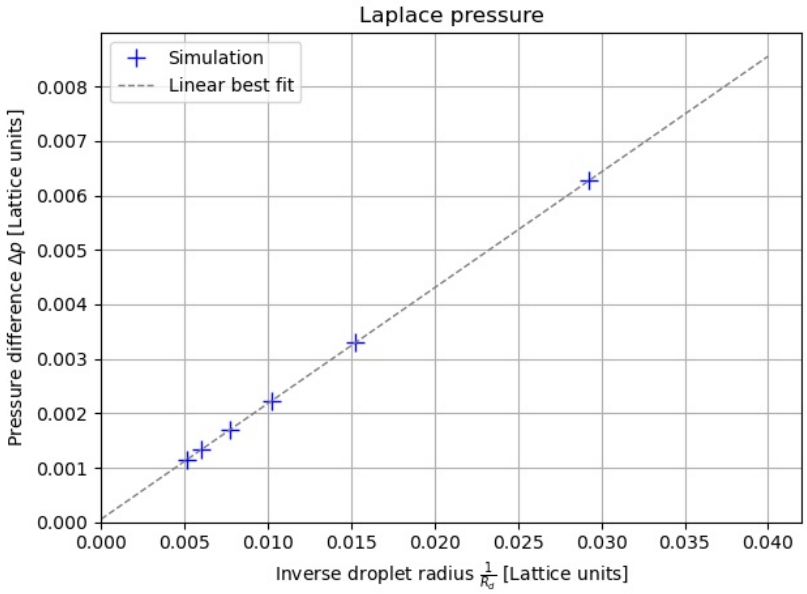



Figure 4.43: Plotting of the pressure difference between edge and centre of droplet vs the inverse of the droplet radius. The Young-Laplace stipulates that pressure difference and radius are inversely proportional.

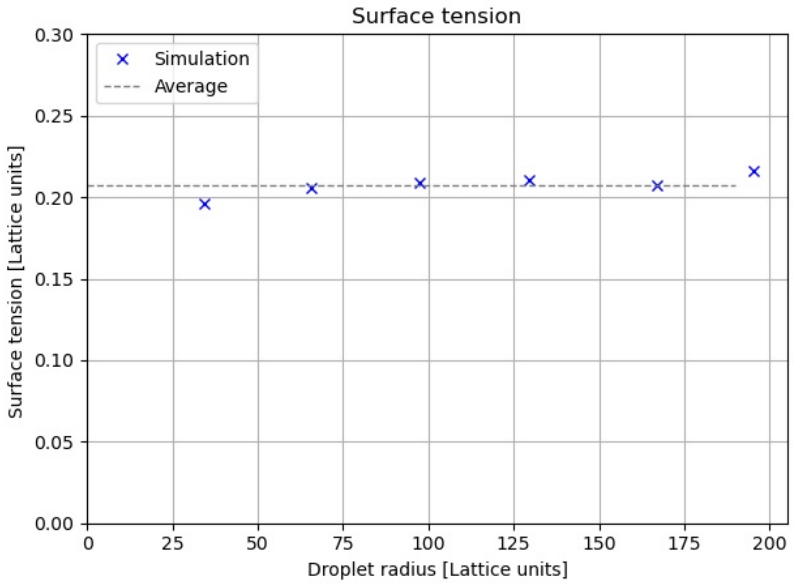



Figure 4.44: Plotting of the surface tension, i.e. pressure difference between edge and centre of droplet times the radius, vs droplet radius. The Young-Laplace law stipulates that the surface tension should remain constant.

The pressure difference plot fig. 4.43 clearly shows that the simulation follows the Laplace law. However, the linear best fit does not pass through the origin point. This may be due to how the droplet radius is taken.

The resultant surface tension of the simulation shown in fig. 4.44 is an order of magnitude higher than it should be. This is because the pseudopotential method recovered macroscopic surface tension force, see section 2.3, is dependent on the pseudopotential, and not density. Picking the pseudopotential equal to the density would solve the surface tension exactly, but the simulation is no longer thermodynamically consistent. The EoS parameter $a$ also affects the surface tension, but due to the limitation of the current collision operator the simulation model, adjustments to that parameter is limited. It may be possible that using a MRT collision operator allows for adjusting the free relaxation time to help in recovering the correct surface tension.

Thus in conclusion, the simulation can correctly capture the behaviour of the surface tension, but fails to recover the correct value. This deficiency can be accounted for by changing the equation of state parameters such that the surface tension value is similar to the wanted value. However, the range is limited by the collision operator used.

## 4.4. Wall wettability

This section details the test to show the functioning of the wall wettability. The definition of the wall wettability is defined in section 2.3.5 and is determined by the contact angle a droplet of water makes with the wall. Correct wall wettability behaviour is required for the simulations. The criteria for success is to be able to set the wall wettability to an arbitrary value.

### Simulation setup

The setup is based on the similar test performed in section 4.2. A simulation domain of size 150x150x75 nodes is set up for 3D, where the x and y directions are periodic. The surfaces are walls with a no slip boundary condition. No body forces are applied in this simulation. For the 2D simulations, a bigger simulation domain is applied with 1500x150 nodes.

The geometry of the 2D simulation is shown in fig. 4.45, with a close up on the top wall fig. 4.46 and bottom wall fig. 4.46. The ghost node (geometry=0) are shown beyond the wall nodes (geometry = 2 for bottom, = 3 for top) and in between are the fluid nodes (geometry=1).

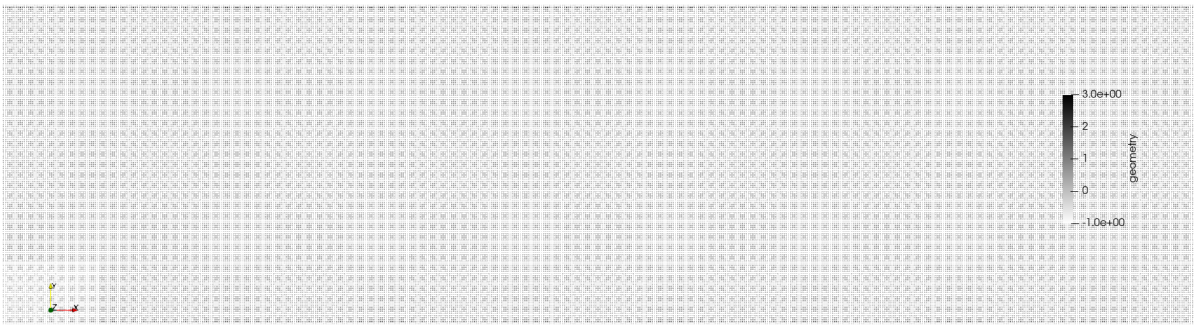



Figure 4.45: Wall wettability geometry 2D, partial zoom

The droplet is initiated using the OpenLB smooth sphere indicator. The sphere radius is one sixth of the simulation domain height. The initial interface width is a quarter of the sphere radius. The origin of the sphere is in the middle of the xy domain, and just above the bottom wall, i.e. a droplet radius plus the interface width.

The simulation stops after 1e6 steps or when convergence is reached. Convergence is based on the `Val-`

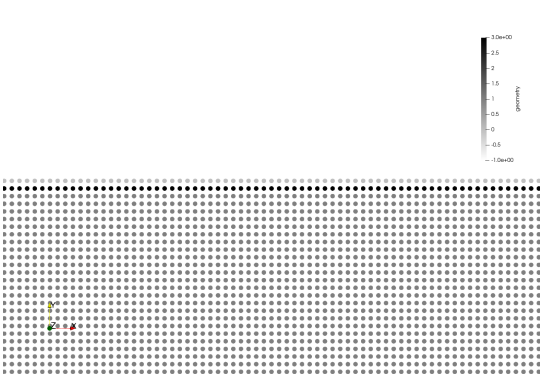


Figure 4.46: Wall wettability geometry 2D, top wall

Figure 4.47: Wall wettability geometry 2D, bottom wall

`ueTracer` class. This checks if a tracked value has a standard deviation less than the defined residual over a time interval. The simulation's average energy is taken as the tracked value, where the time interval is 50 steps, and the residual is 1e-7.

For the multiphase flow, the results from section 4.2 are used: The Peng Robinson equation of state uses $a = 0.5/49$, $b = 2/21$, $R = 1$, $G = -1$. For the forcing scheme, the value for $\sigma$ is taken as 0.31 for 3D and 0.35 for 2D.

The initial liquid and vapour density is taken from the Maxwell construction. The sphere internal density is initialized with 0.99 times the theoretical liquid density. The wall wettability is changed by changing the wall fixed density. The wall density is initially set to the vapour density and ramps up to the tested density over 5e3 timesteps, as is explained in section 3.10

## Results

After the simulations have concluded, the contact angle can be measured and thus the wall wettability determined. The contact angle is then calculated from the wetting length, $l_d$, and the droplet height $h_d$, given by the equation in eq. (4.2) [31]. The contour plot density chosen for measurement is equal to the wall density. This simplifies the measurements needed to be done and it does not vary significantly from taken a contour of constant density. This is explained in more detail in appendix D

$$\theta = \arctan\left(\frac{l_d}{2|R_d - h_d|}\right) \tag{4.2}$$

$$R_d = \frac{4h_d^2 + l_d^2}{8h_d} \tag{4.3}$$

The 2D simulation resulting contour plots are shown in figs. 4.48 to 4.62, and plotted in fig. 4.69, where the measurements and angles are shown in table 4.6. The 3D simulations resulting contour plots are shown in figs. 4.63 to 4.68, and plotted in fig. 4.70, where the measurements and angles are shown in table 4.7.

The attempts for simulating the 3D wall wettability at 0.5 and 7.5 wall density failed. For the 0.5 case, the droplet was initialized a bit too close, resulting in the droplet being pushed away from the wall during the initial timesteps. As such, the droplet did not adhere to the wall. For the 7.5 case, the simulation domain was not sufficiently large enough, and thus the droplet interacted with itself through the periodic boundary, resulting in complete wetting of the surface with no distinct droplet appearing.

Table 4.5: Wall wettability simulation setup parameters

| **Parameter** | **Symbol** | **Value 3D** | **Value 2D** | **Unit** |
|---|---|---|---|---|
| Spatial resolution | N | 75 | 150 | - |
| Temporal resolution | $N_t$ | $7N$ | $7N$ | - |
| Relaxation time | $\tau$ | 0.8 | 0.8 | - |
| Characteristic length | L | $100 \cdot 10^{-6}$ | $100 \cdot 10^{-6}$ | $m$ |
| Domain size X | | $2 \cdot L$ | $10 \cdot L$ | $m$ |
| Domain size Y | | $2 \cdot L$ | $1 \cdot L$ | $m$ |
| Domain size Z | | $1 \cdot L$ | - | $m$ |
| Characteristic speed | U | 50 | 40 | $m \cdot s^{-1}$ |
| Physical kinematic viscosity | $\nu$ | $1 \cdot 10^{-5}$ | $1 \cdot 10^{-5}$ | $m^2 \cdot s^{-1}$ |
| Physical density | $\rho$ | 1 | 1 | $kg \cdot m^{-3}$ |
| Vapour density multiplier | | 1 | 1 | - |
| Liquid density multiplier | | 0.99 | 0.99 | - |
| Maximum timesteps | | $100 \cdot 10^3$ | $100 \cdot 10^3$ | $\Delta t$ |
| EoS parameter a | $a$ | $\frac{0.5}{49}$ | $\frac{0.5}{49}$ | - |
| EoS parameter b | $b$ | $\frac{2}{21}$ | $\frac{2}{21}$ | - |
| EoS acentric factor | $\omega_{EoS}$ | 0.3443 | 0.3443 | - |
| Correction parameter $\sigma$ | $\sigma$ | 0.31 | 0.35 | - |

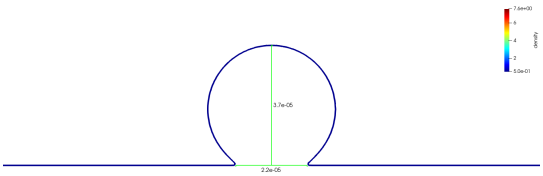


Figure 4.48: Wall wettability 2D simulation with $\rho_w = 0.5$ and contour at $\rho = \rho_w$

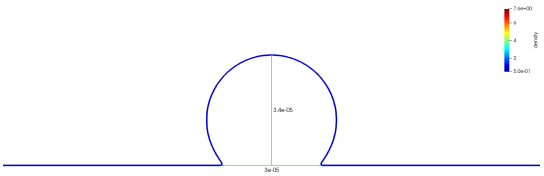


Figure 4.49: Wall wettability 2D simulation with $\rho_w = 1.0$ and contour at $\rho = \rho_w$

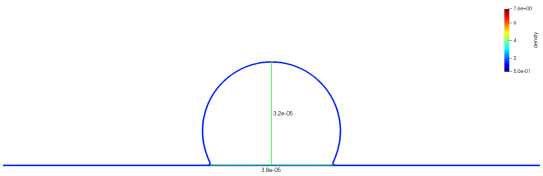


Figure 4.50: Wall wettability 2D simulation with $\rho_w = 1.5$ and contour at $\rho = \rho_w$

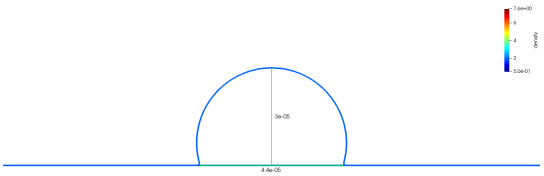


Figure 4.51: Wall wettability 2D simulation with $\rho_w = 2.0$ and contour at $\rho = \rho_w$

Table 4.6: Wall wettability 2D droplet measurements and contact angle

| Wall Density | Droplet Height [m] | Wetting Length [m] | Contact Angle [deg] |
|---|---|---|---|
| 0.5 | $3.7 \cdot 10^{-5}$ | $2.2 \cdot 10^{-5}$ | 146.9 |
| 1 | $3.4 \cdot 10^{-5}$ | $3.0 \cdot 10^{-5}$ | 132.4 |
| 1.5 | $3.2 \cdot 10^{-5}$ | $3.8 \cdot 10^{-5}$ | 118.6 |
| 2 | $3.0 \cdot 10^{-5}$ | $4.4 \cdot 10^{-5}$ | 107.5 |
| 2.5 | $2.83 \cdot 10^{-5}$ | $5.1 \cdot 10^{-5}$ | 96.0 |
| 3 | $2.67 \cdot 10^{-5}$ | $5.7 \cdot 10^{-5}$ | 86.3 |
| 3.5 | $2.51 \cdot 10^{-5}$ | $6.4 \cdot 10^{-5}$ | 76.2 |
| 4 | $2.34 \cdot 10^{-5}$ | $7.1 \cdot 10^{-5}$ | 66.8 |
| 4.5 | $2.16 \cdot 10^{-5}$ | $7.7 \cdot 10^{-5}$ | 58.6 |
| 5 | $1.97 \cdot 10^{-5}$ | $8.56 \cdot 10^{-5}$ | 49.4 |
| 5.5 | $1.76 \cdot 10^{-5}$ | $9.48 \cdot 10^{-5}$ | 40.7 |
| 6 | $1.52 \cdot 10^{-5}$ | $10.5 \cdot 10^{-5}$ | 32.3 |
| 6.5 | $1.27 \cdot 10^{-5}$ | $11.8 \cdot 10^{-5}$ | 24.3 |
| 7 | $0.96 \cdot 10^{-5}$ | $13.4 \cdot 10^{-5}$ | 16.3 |
| 7.5 | $0.63 \cdot 10^{-5}$ | $15.0 \cdot 10^{-5}$ | 9.6 |

Table 4.7: Wall wettability 3D droplet measurements and contact angle

| Wall Density | Droplet Height [m] | Wetting Length [m] | Contact Angle [deg] |
|---|---|---|---|
| 1.5 | $2.95 \cdot 10^{-5}$ | $3.8 \cdot 10^{-5}$ | 114.4 |
| 2.5 | $2.68 \cdot 10^{-5}$ | $5.1 \cdot 10^{-5}$ | 92.8 |
| 3.5 | $2.32 \cdot 10^{-5}$ | $6.2 \cdot 10^{-5}$ | 73.6 |
| 4.5 | $1.91 \cdot 10^{-5}$ | $7.0 \cdot 10^{-5}$ | 57.2 |
| 5.5 | $1.43 \cdot 10^{-5}$ | $7.8 \cdot 10^{-5}$ | 40.2 |
| 6.5 | $0.85 \cdot 10^{-5}$ | $8.4 \cdot 10^{-5}$ | 22.9 |

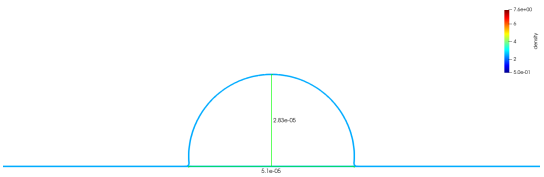


Figure 4.52: Wall wettability 2D simulation with $\rho_w = 2.5$ and contour at $\rho = \rho_w$

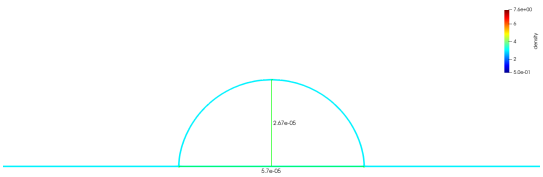


Figure 4.53: Wall wettability 2D simulation with $\rho_w = 3.0$ and contour at $\rho = \rho_w$

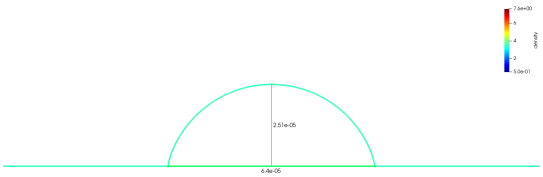


Figure 4.54: Wall wettability 2D simulation with $\rho_w = 3.5$ and contour at $\rho = \rho_w$

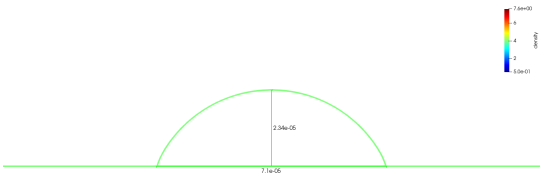


Figure 4.55: Wall wettability 2D simulation with $\rho_w = 4.0$ and contour at $\rho = \rho_w$

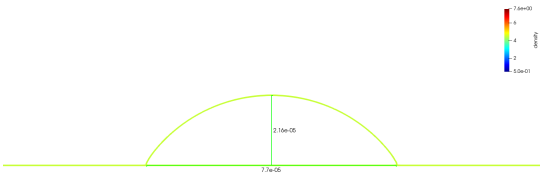


Figure 4.56: Wall wettability 2D simulation with $\rho_w = 4.5$ and contour at $\rho = \rho_w$

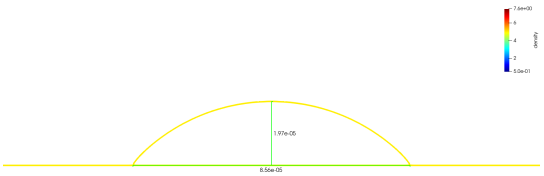


Figure 4.57: Wall wettability 2D simulation with $\rho_w = 5.0$ and contour at $\rho = \rho_w$

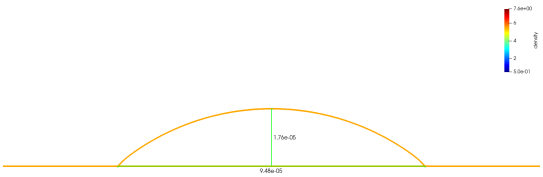


Figure 4.58: Wall wettability 2D simulation with $\rho_w = 5.5$ and contour at $\rho = \rho_w$

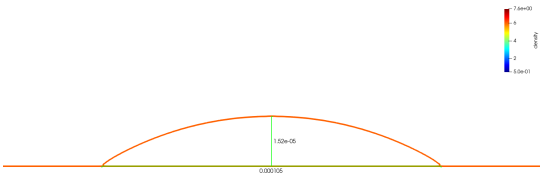


Figure 4.59: Wall wettability 2D simulation with $\rho_w = 6.0$ and contour at $\rho = \rho_w$

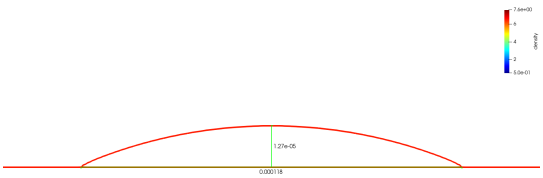


Figure 4.60: Wall wettability 2D simulation with $\rho_w = 6.5$ and contour at $\rho = \rho_w$

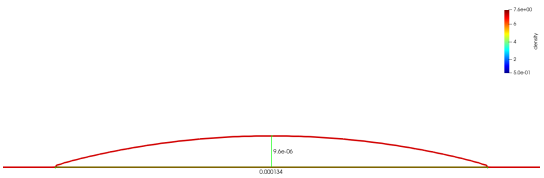


Figure 4.61: Wall wettability 2D simulation with $\rho_w = 7.0$ and contour at $\rho = \rho_w$

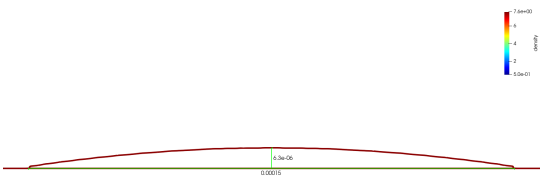


Figure 4.62: Wall wettability 2D simulation with $\rho_w = 7.5$ and contour at $\rho = \rho_w$

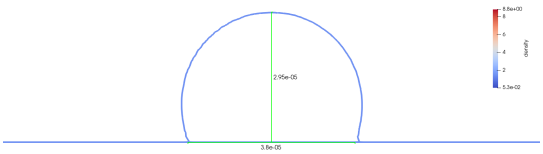


Figure 4.63: Wall wettability 3D simulation with $\rho_w = 1.5$ and contour at $\rho = \rho_w$

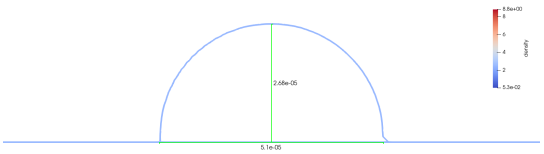


Figure 4.64: Wall wettability 3D simulation with $\rho_w = 2.5$ and contour at $\rho = \rho_w$

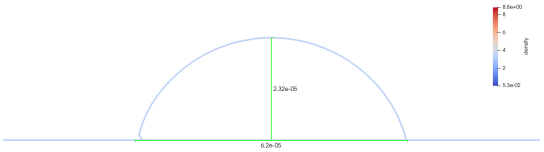


Figure 4.65: Wall wettability 3D simulation with $\rho_w = 3.5$ and contour at $\rho = \rho_w$

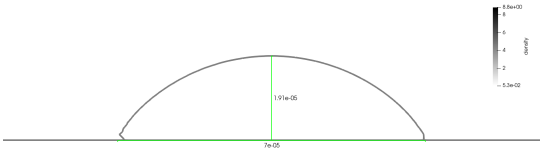


Figure 4.66: Wall wettability 3D simulation with $\rho_w = 4.5$ and contour at $\rho = \rho_w$

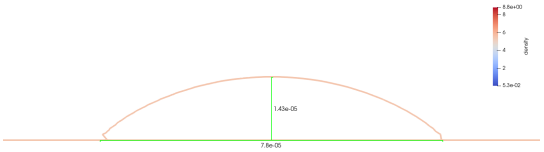


Figure 4.67: Wall wettability 3D simulation with $\rho_w = 5.5$ and contour at $\rho = \rho_w$

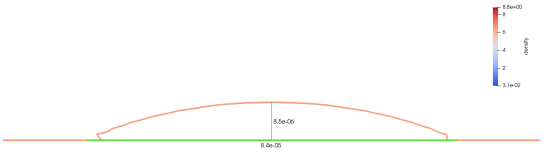


Figure 4.68: Wall wettability 3D simulation with $\rho_w = 6.5$ and contour at $\rho = \rho_w$

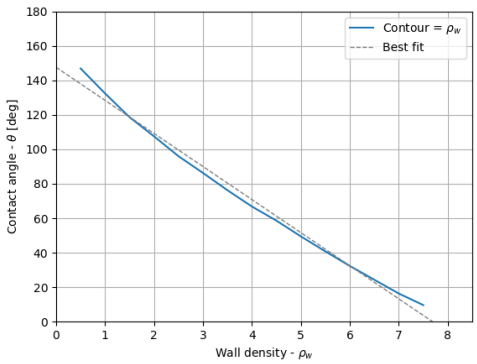



Figure 4.69: Wall wettability plot showing contact angle as function of wall density for 2D droplets

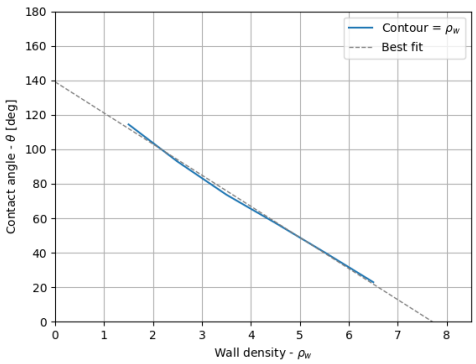



Figure 4.70: Wall wettability plot showing contact angle as function of wall density for 3D droplets

The plots shown in fig. 4.69 and fig. 4.70 for both 2D and 3D shows that the contact angle is a near linear function of the wall density. The curve that is shown in fig. 4.69 and fig. 4.70 is expected as it also is present in fig. 4.71 [130] who used the same wall wettability method.

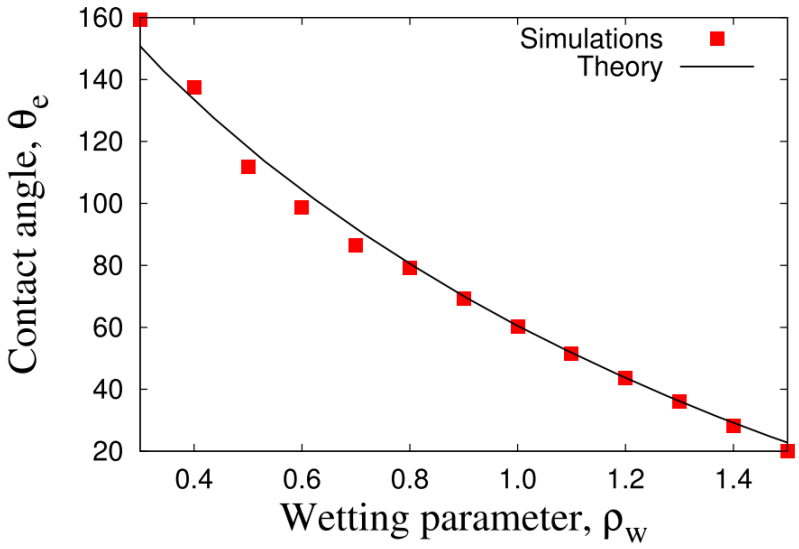



Figure 4.71: Contact angle as simulated by Srivastava [130] using the same wall wettability method

Overall, the wall wettability functions as expected, and allows for setting the exact contact angle. This allows for simulating the chamber with different wall wettabilities to determine if that impacts the boiling behaviour. An estimate of the wall density can be quickly calculated with decent accuracy near the neutral wettability, due to the near linearity present.

# 5 Verification Thermal LBM

This chapter details the verification of the simulations, focusing on the thermal aspects. The thermal phase change verification case detailed in this chapter is the $D^2$ law. An implementation mistake was found in the verification process, which was not covered by any verification test. The mistake is analysed and corrected, each getting a discussion.

## 5.1. Thermal MRT implementation

The thermal MRT implementation, see section 3.5.1, is verified by comparing the simulation results of a droplet evaporation using BGK AD dynamics with the MRT AD dynamics. The simulation thermal results are shown in fig. 5.1. The temperature across the domain, from the bottom-left corner to the top-right corner, is additionally plotted, shown in fig. 5.2 and fig. 5.3. A diagonal line is used such that the temperature shown crosses the cold spots inside the droplet.

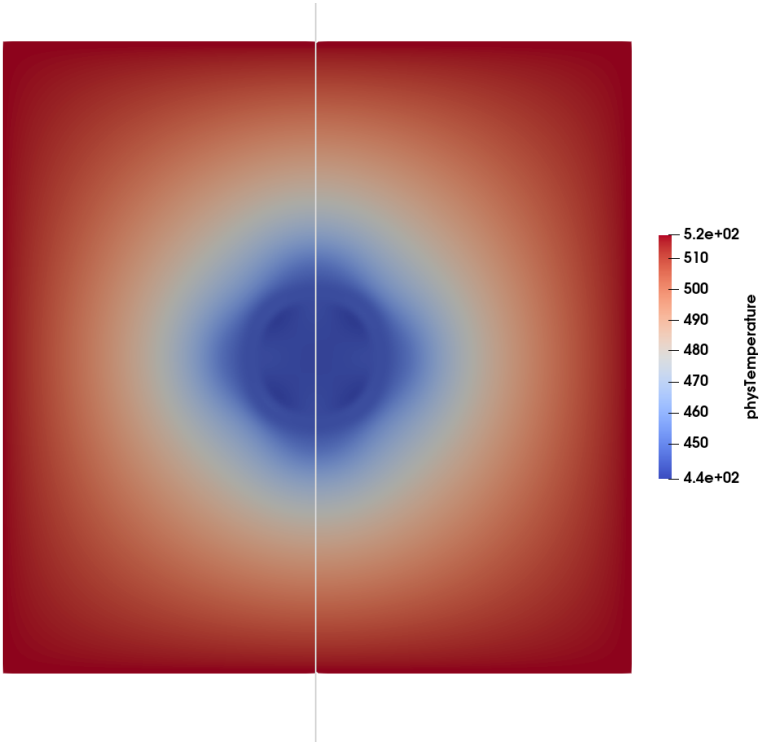



Figure 5.1: Thermal advection diffusion Dynamics BGK (left) vs MRT (right) comparison

The maximum deviation in the temperature happens inside the droplet, where the MRT simulation predicts 0.05K lower temperature than the BGK simulation. With such small difference, it is shown that the MRT dynamics is implemented correctly.

## 5.2. Droplet evaporation

The next verification target is the evaporative model. A commonly used benchmark is the $D^2$-law of droplet evaporation [31]. This law states that the droplet radius changes linearly in time [72], since both the rate of

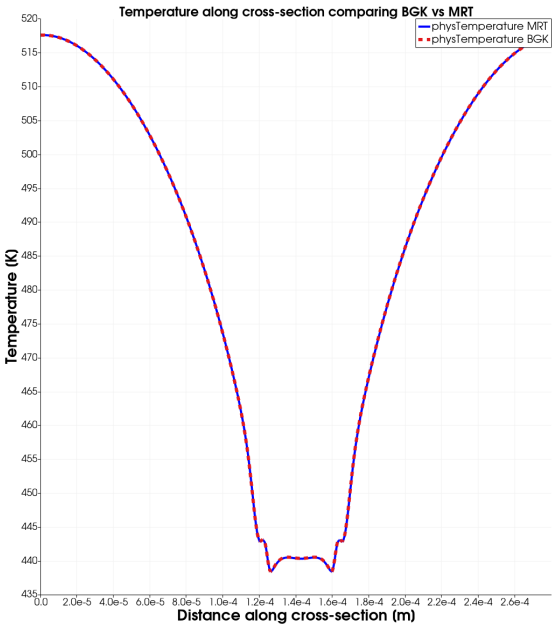



Figure 5.2: Temperature across the simulation domain (bottom-left to top-right)

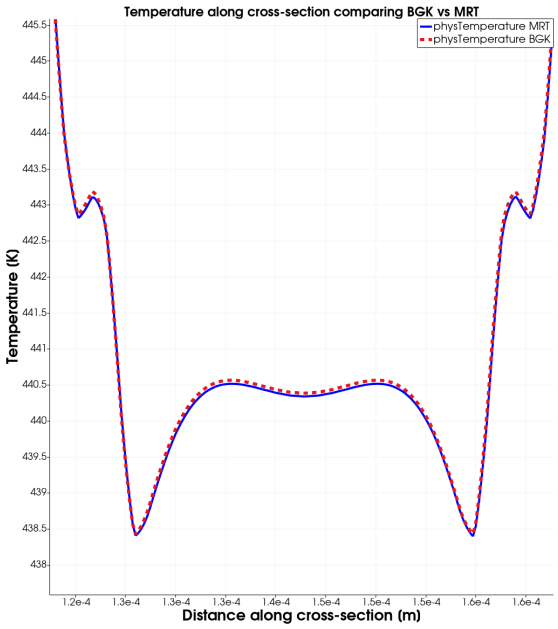



Figure 5.3: Temperature across the simulation domain (bottom-left to top-right) zoomed on droplet

evaporation as well as rate of heating is dependent on the droplet surface area. It also shows that the rate of evaporation is related to thermal conductivity. The $D^2$-law assumes no forced nor natural convection. The transport properties for the both gas and liquid are the same.

In this unit test a larger bug was found specific to how the phase change is implemented into OpenLB. This is explained in more detail in section 5.3 and how to properly implement it in section 5.3.2. However, the $D^2$-law unit test did not show any abnormalities. In section 5.2.1 it is shown how even with an erroneous implementation, correct looking results can be achieved. This is still included in this thesis as a warning, but less attention is spent on that section.

An interesting phenomena occurs due to the droplet not being at equilibrium at simulation start, shown in fig. 5.4. The expansion and contraction pushes heat to the outside, making the internals of the droplet colder and the outside warmer than the initial temperature. The additional heat is then cooled by the constant temperature wall. This results in a droplet which has a colder temperature than expected. To avoid this, the temperature field is held at a constant temperature till the droplet has reached equilibrium and heating begins.

### 5.2.1. Erroneous simulation setup

A simulation domain of size 150x150x150 with periodic boundaries is set up. A droplet of radius $L/5$ is initialized in the centre using the OpenLB's smooth sphere indicator. The viscosity used in the simulation is 1e-5 $m^2 s^{-1}$.

The thermal lattice boundary is set to a constant temperature. The thermal heat capacity for both vapour and liquid in lattice units is 6.6, following closely to the values given by Li et al. and Gong et al. [45, 79].

After 2.5e3 steps, the droplet has come to rest. Then the thermal lattice boundary temperature is increased by 32K. At the same time the droplet radius is recorded. For tracking the droplet radius, a density contour of 4 is used as the cut-off value, the contour length, i.e. the circumference, is used to calculate the radius. The simulation ends after 25e3 steps.

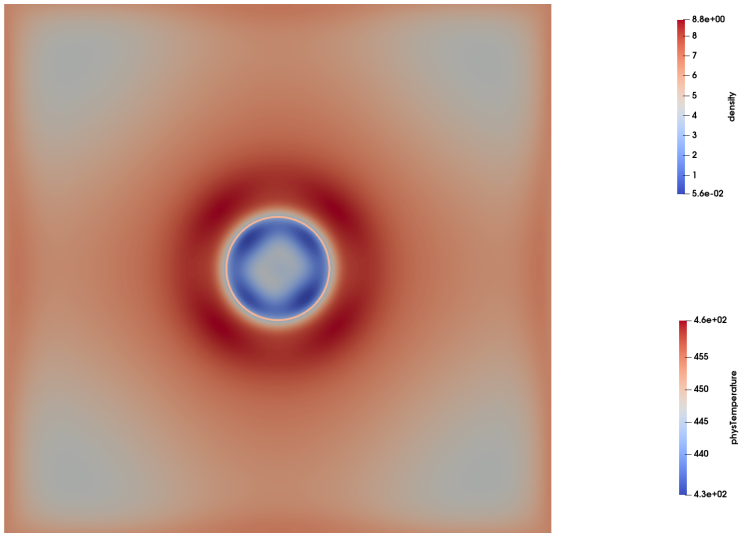



Figure 5.4: Thermal behaviour during the initial timesteps. Initial temperature is $T = 453$K. Droplet cools down due to expansion and contraction till an equilibrium is reached. Contour at $\rho = 6$ shows the droplet location

## Erroneous Results

A series of snapshots at the start of heating, during and at the end are given in figs. 5.6 to 5.9. The results of the simulations are shown in fig. 5.5. As can be seen, after the initial period, the change in radius is constant, which is in line with the $D^2$-Law. The gradients should be a multiple of each other, and the ratio compared to $\lambda = 0.1$ are: 2.17 for $\lambda = 0.2$ and 3.00 for $\lambda = 3.00$.

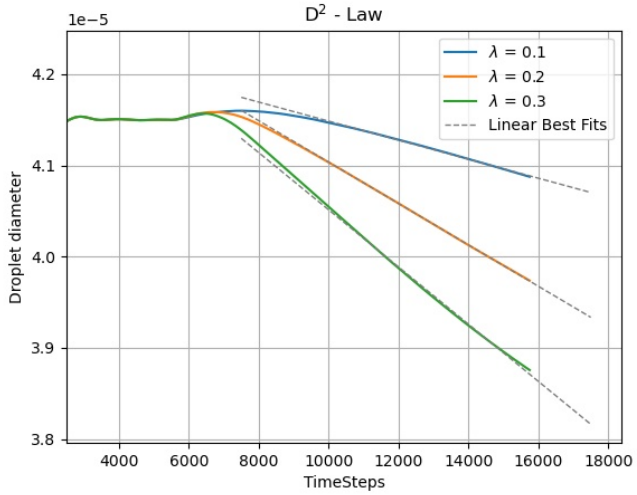



Figure 5.5: Change in droplet diameter due to evaporation using erroneous thermal phase change implementation

The effect of the erroneous implementation is visible in the thermal lattices in figs. 5.6 to 5.9. The latent heat is much lower than expected, which results in a faster heating of the droplet. There is almost no noticeable difference in temperature between the droplet and the surrounding vapour. Due to the quick heating, the droplet does not have enough time to evaporate and thus first expands. This expansion is seen in the initial 'hump' at $\Delta t = 6500$ in fig. 5.5.

The cross shape that is present in the temperature field of the simulations in fig. 5.7 and fig. 5.8 is due to the spurious currents. This is explained in more detail in section 6.1. Even in this erroneous setup, the simulation follows the $D^2$ law, as can be seen by linear evaporation of the droplet.

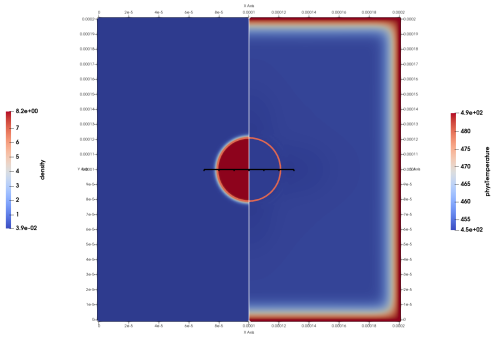


Figure 5.6: Erroneous results showing density (left) and temperature (right) at the start of the heating $\Delta t = 5000$

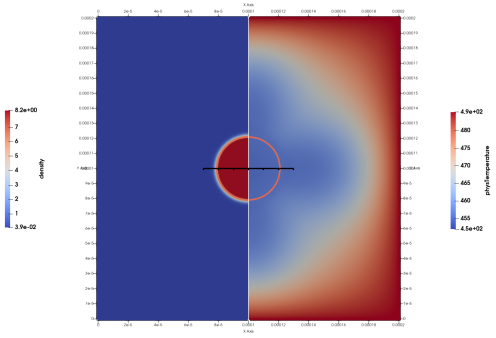


Figure 5.7: Erroneous results showing density (left) and temperature (right) during the heating $\Delta t = 8000$

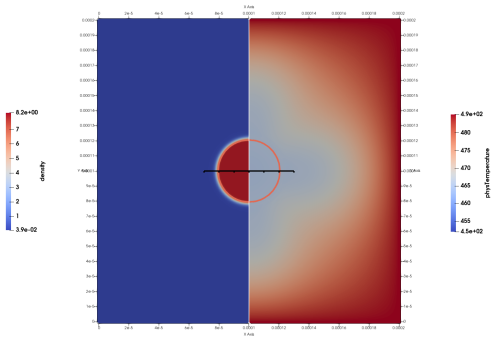


Figure 5.8: Erroneous results showing density (left) and temperature (right) at the start of the heating $\Delta t = 10000$

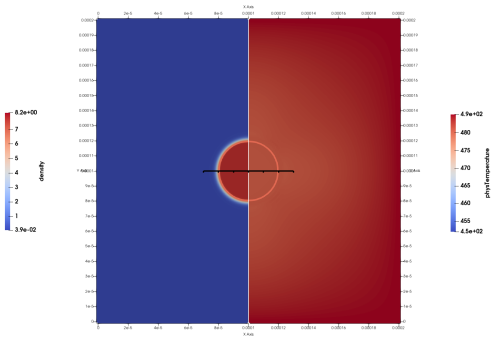


Figure 5.9: Erroneous results showing density (left) and temperature (right) during the heating $\Delta t = 15750$

### 5.2.2. Proper simulation setup

The problem identified in section 5.2.1 is fixed and the proper simulations to verify the $D^2$ Law are run. The problem and solution are explained in depth in section 5.3.1 and section 5.3.2. The setup parameters of this simulation series are shown in table 5.1. The geometry is given in fig. 5.10.

For the simulation, three different thermal conductivities are simulated. The physical thermal conductivities are $\lambda = 0.1, 0.3, 0.5$. In lattice thermal conductivities simulated are $0.0217 \cdot m$, with m = 1,3,5. Combined with the lattice specific heat, this results in the diffusivity of $D = 0.286691\rho \cdot m$.

The characteristic length and speed, with the spacial and temporal resolution determines the values to nondimensionalize the simulation. The characteristic length is kept constant for all simulations, and the characteristics speed is adapted such that it is below the maximum expected velocity.

To increase the simulation size, while keeping the same characteristic length, the domain size is set to twice the characteristic length. This is to ensure that the hot wall is sufficiently far away from the droplet.

The physical viscosity is kept constant across all simulations such that they are comparable. The relaxation time is calculated from the physical viscosity and the resolutions. A value for the physical viscosity is taken such that the resulting relaxation time is above 0.65 and thus ensuring the stability of the simulations. A lower viscosity would result in unstable simulations. In combination with higher resolutions to ensure stability, the simulations would take too long.

Table 5.1: $D^2$Law simulation setup parameters

| **Parameter** | **Symbol** | **Value 2D** | **Unit** |
|---|---|---|---|
| Spatial resolution | N | 100 | - |
| Temporal resolution | $N_t$ | 7 | - |
| Relaxation time | $\tau$ | $\frac{1}{1.4}$ | - |
| Characteristic length | L | $100 \cdot 10^{-6}$ | $m$ |
| Domain size X | | $2 \cdot L$ | |
| Domain size Y | | $2 \cdot L$ | |
| Characteristic speed | U | 20 | $m \cdot s^{-1}$ |
| Physical kinematic viscosity | $\nu$ | $1 \cdot 10^{-5}$ | $m^2 \cdot s^{-1}$ |
| Vapour density multiplier | | 1.0 | - |
| Liquid density multiplier | | 0.99 | - |
| Initial temperature ratio | $Tr$ | 0.7 | - |
| Heated temperature ratio | $Tr$ | 0.8 | - |
| EoS parameter a | $a$ | $\frac{0.5}{49}$ | - |
| EoS parameter b | $b$ | $\frac{2}{21}$ | - |
| EoS parameter R | $R$ | 15.23 | - |
| EoS acentric factor | $\omega_{EoS}$ | 0.3443 | - |
| Initial timesteps | | $7 \cdot 10^3$ | $\Delta t$ |
| Maximum timesteps | | $1407 \cdot 10^3$ | $\Delta t$ |
| Thermal relaxation time | $\tau_g$ | 0.7 | |
| Physical specific heat | $c_v$ | 4000 | $\frac{J}{kgK}$ |
| Physical thermal conductivity | $\lambda$ | 0.1, 0.3, 0.5 | $\frac{W}{mK}$ |

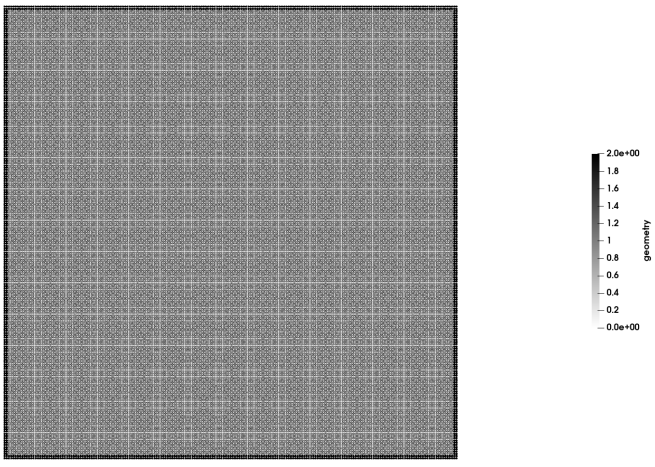



Figure 5.10: Geometry used in the $D^2$ Law simulations. For the fluid lattice, nodes with geometry equal to 1 and 2 are normal fluid nodes, periodic boundaries are used. For the thermal lattice, nodes with geometry equal to 1 are normal thermal nodes and geometry = 2 are thermal wall nodes.

### 5.2.3. Proper results

The initial state of the simulation just after the thermal wall temperature has increased is shown in fig. 5.11. The final conditions are shown in figs. 5.12 to 5.13 for the three different thermal conductivities simulated. The resultant evaporation shown in the change of droplet radius is shown in fig. 5.15 with a droplet cutoff value of $\rho = 4$ and for $\rho = 7$ in fig. 5.16.

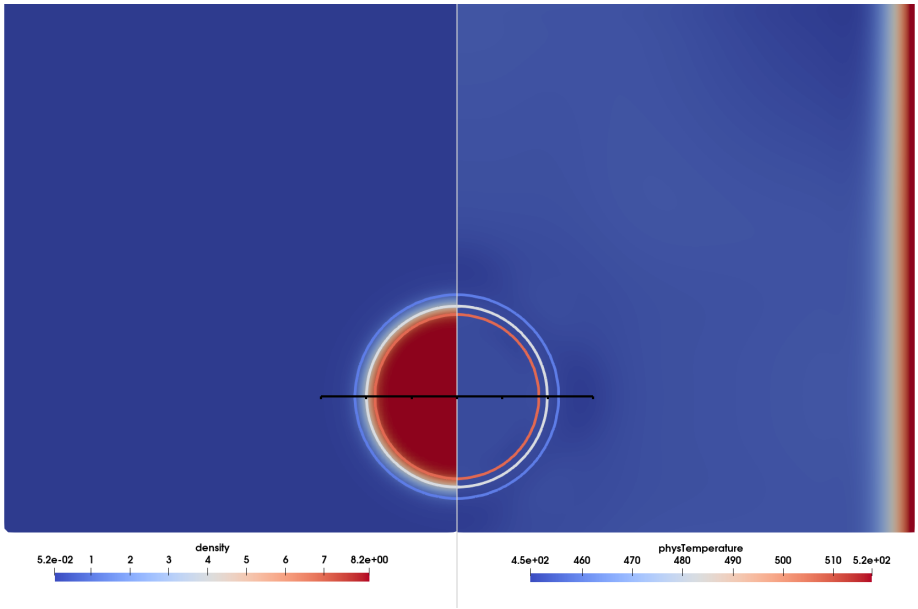



Figure 5.11: Droplet at equilibrium just after the domain edge temperature was increased. Showing density (left) and temperature (right) with contours for $\rho = 1, 4, 7$. $\Delta t = 7500$

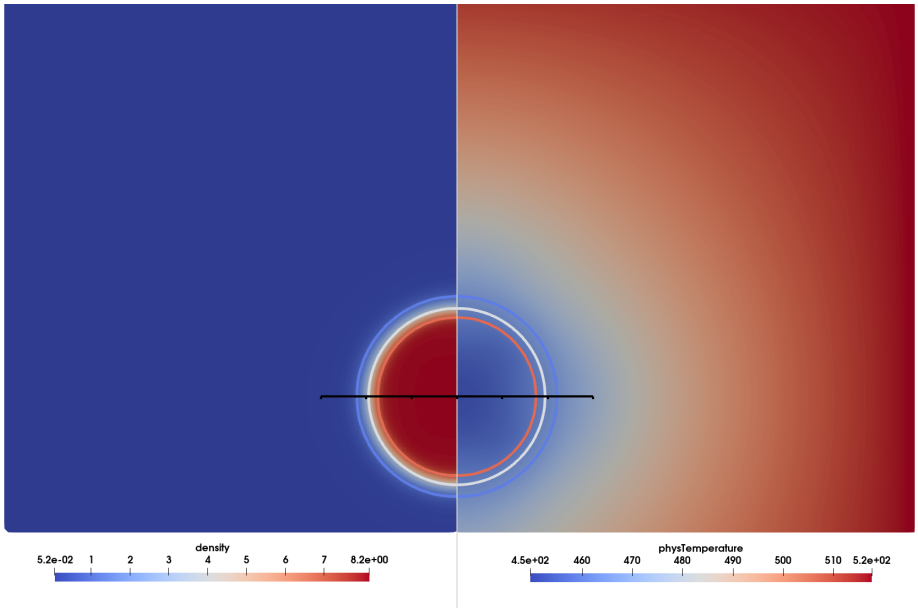



Figure 5.12: Final simulation state of droplet with physical thermal conductivity of $\lambda = 0.1\ \frac{W}{mK}$. Showing density (left) and temperature (right) with contours for $\rho = 1, 4, 7$. $\Delta t = 1.405 \cdot 10^6$

In each of the $D^2$ simulation plots, a black ruler was added to help show the evaporation. Due to the proper latent heat, the droplet temperature is lower than the temperature experienced in fig. 5.9. For $\lambda = 0.1$, the droplet temperature was almost unchanged compared to the initial state.

The change in droplet radius where the same droplet cut-off value was used as in the erroneous $D^2$ simulation

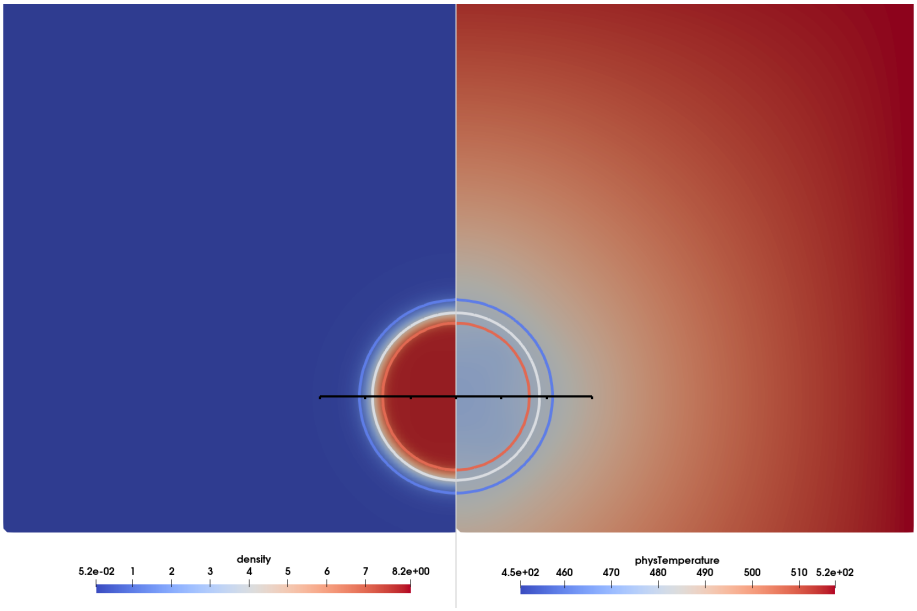



Figure 5.13: Final simulation state of droplet with physical thermal conductivity of $\lambda = 0.3\ \frac{W}{mK}$. Showing density (left) and temperature (right) with contours for $\rho = 1, 4, 7$. $\Delta t = 1.405 \cdot 10^6$

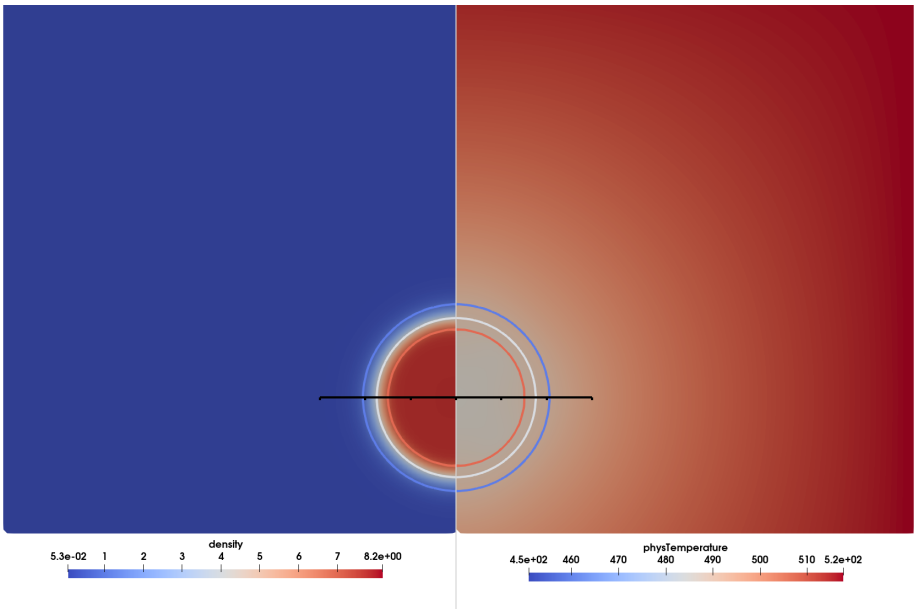



Figure 5.14: Final simulation state of droplet with physical thermal conductivity of $\lambda = 0.5\ \frac{W}{mK}$. Showing density (left) and temperature (right) with contours for $\rho = 1, 4, 7$. $\Delta t = 1.405 \cdot 10^6$

is shown in fig. 5.15. The plot shows that the change is linear as expected. The measurement for the gradient is tabulated in table 5.2. The difference in radius for $\lambda = 0.1$ is very small, which is to be expected since the droplet internal temperature barely changed as well.

The same plot of fig. 5.15 but with a different density cut-off value $\rho = 7$ is used. There is a slight difference in the calculated gradient, but more interesting is that the droplet radius expansion at the start ($\Delta t \approx 20000$) of the simulation is no longer present. This is especially visible for the $\lambda = 0.1$ simulation.

The gradient calculations for cut-off values $\rho = 4$ and $\rho = 7$ are given in table 5.2 and table 5.3 respectively. The gradients are converted to physical units to get a better feel for the magnitude. In this simulation, one timestep $\Delta t$ is equal to $\frac{1}{1.4} \cdot 10^{-8}$s. In both cases the gradient ratio compared to $\lambda = 0.1$ is slightly too high.

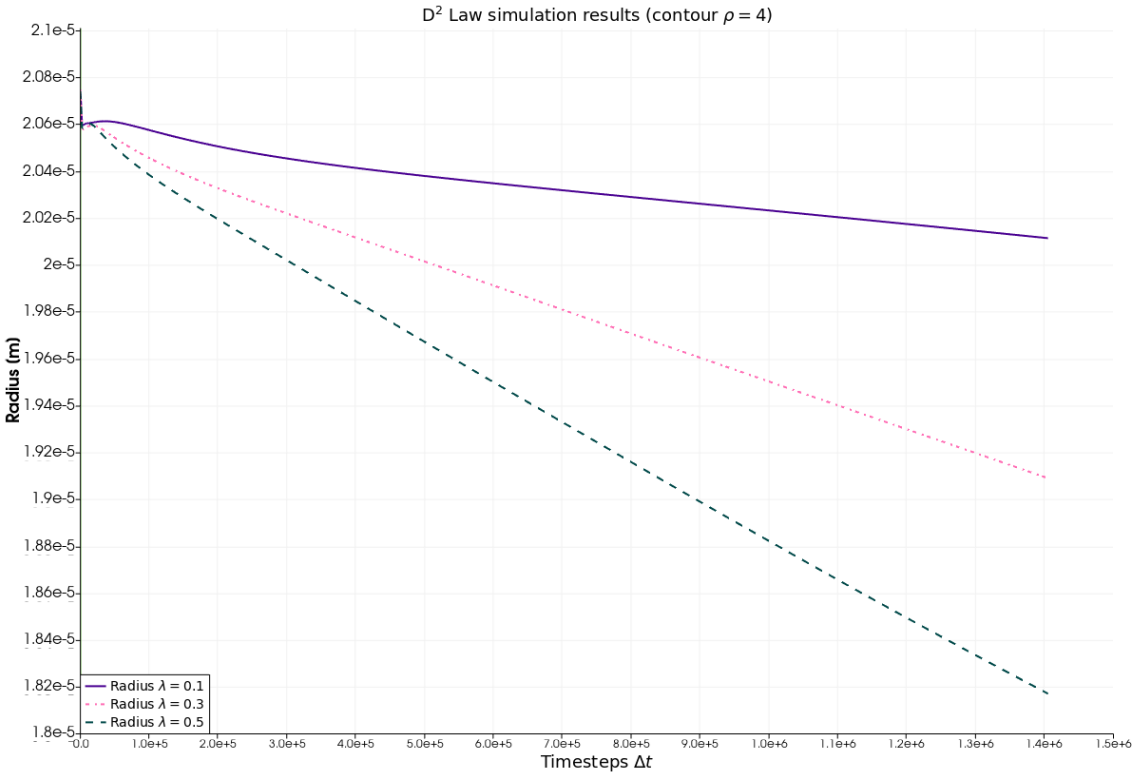



Figure 5.15: Change in droplet diameter due to evaporation using corrected thermal phase change implementation measuring droplet size with a contour of $\rho = 4$

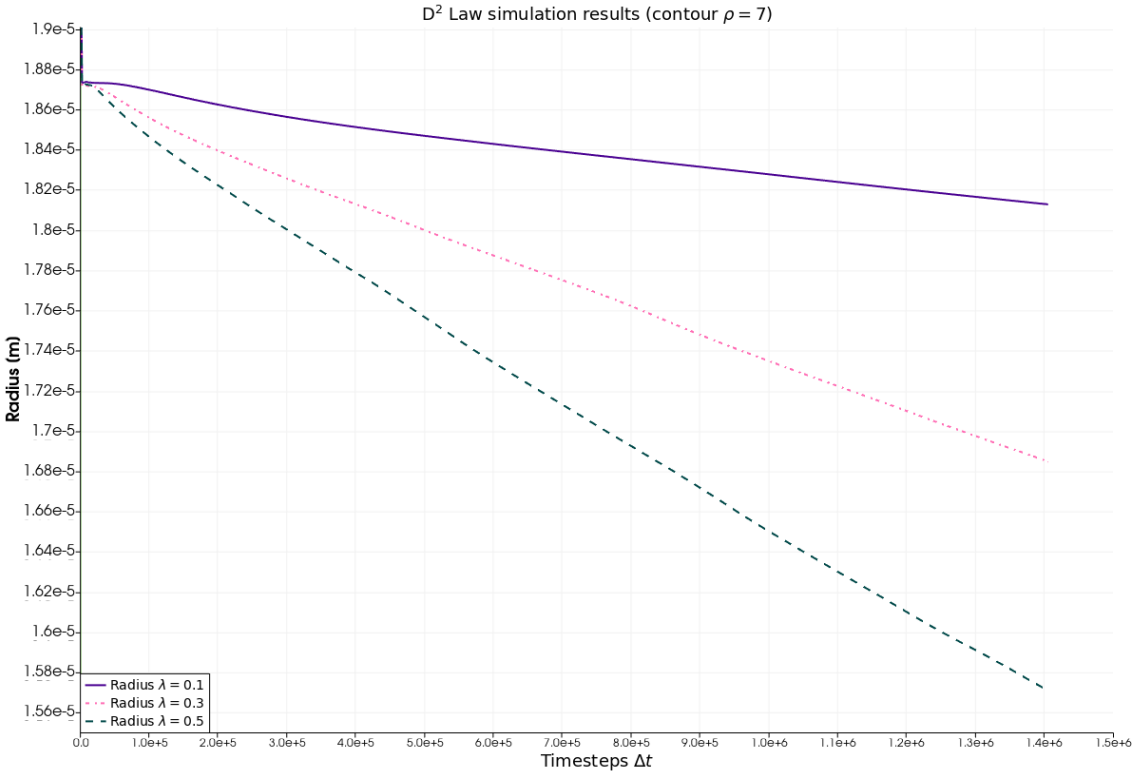



Figure 5.16: Change in droplet diameter due to evaporation using corrected thermal phase change implementation measuring droplet size with a contour of $\rho = 7$

However, the ratio between the 0.3 and 0.5, which should be $0.5/0.3 = 1.\dot{6}$ is 1.63 for both $\rho = 4$ and $\rho = 7$ showing good consistency across those contours. The gradient for $\rho = 4$ is lower than the ones in $\rho = 7$ indicating that the interface is expanding slightly as the droplet evaporates.

Table 5.2: Gradient calculations for the $D^2$law simulations with a contour at $\rho = 4$

| Thermal conductivity $\lambda$ [$\frac{W}{mK}$] | Radius [$\mu$ m] at $1.4 \cdot 10^6 \Delta t$ | Radius [$\mu$ m] at $0.4 \cdot 10^6 \Delta t$ | Gradient [$\mu$ m/s] | Gradient ratio compared to $\lambda = 0.1$ |
|---|---|---|---|---|
| 0.1 | 20.1140 | 20.4136 | -41.7 | 1 |
| 0.3 | 19.0902 | 20.1165 | -142.6 | 3.43 |
| 0.5 | 18.1710 | 19.8452 | -233.2 | 5.59 |

Table 5.3: Gradient calculations for the $D^2$law simulations with a contour at $\rho = 7$

| Thermal conductivity $\lambda$ [$\frac{W}{mK}$] | Radius [$\mu$ m] at $1.4 \cdot 10^6 \Delta t$ | Radius [$\mu$ m] at $0.4 \cdot 10^6 \Delta t$ | Gradient [$\mu$ m/s] | Gradient ratio compared to $\lambda = 0.1$ |
|---|---|---|---|---|
| 0.1 | 18.1283 | 18.5128 | -53.6 | 1 |
| 0.3 | 16.8517 | 18.1293 | -178.0 | 3.32 |
| 0.5 | 15.7099 | 17.7884 | -289.5 | 5.41 |

## 5.3. Thermodynamic latent heat

As was preluded in section 5.2.1, there was a flaw in the implementation of the phase change term into OpenLB. This flaw causes the latent heat to not be recovered correctly. This erroneous recovery from a straightforward implementation is shown in section 5.3.1. Then the correct way to implement the phase change term is discussed in section 5.3.2.

### 5.3.1. Link to latent heat

The used phase change term $\left(-\frac{T}{\rho c_v}\left(\frac{\mathrm{d}P}{\mathrm{d}T}\right)_\rho \nabla \cdot u\right)\Delta t$ can also be expressed in terms of latent heat of vaporization. The starting point of this derivation is eq. (5.1), where $v_{fg}$ is the specific volume difference between point $f$ and $g$, $h_{fg}$ the latent heat [158].

$$\left(\frac{\mathrm{d}p}{\mathrm{d}T}\right)_v = \frac{h_{fg}}{T v_{fg}} \tag{5.1}$$

The first step is to express $v_{fg}$ for a small change in volume.

$$v_{fg} = \Delta v = v_2 - v_1 = (v_1 + \mathrm{d}v) - v_1 = \mathrm{d}v \tag{5.2}$$

The next step is to convert specific volume to density: $v = \frac{1}{\rho}$

$$v_{fg} = \frac{\mathrm{d}v}{\mathrm{d}\rho}\mathrm{d}\rho = \frac{\mathrm{d}\frac{1}{\rho}}{\mathrm{d}\rho}\mathrm{d}\rho = -\frac{1}{\rho^2}\mathrm{d}\rho \tag{5.3}$$

Inserting eq. (5.3) into eq. (5.1), and rearranging for $h_{fg}$ results in eq. (5.4).

$$h_{fg} = T\frac{1}{\rho^2}\mathrm{d}\rho\left(\frac{\mathrm{d}p}{\mathrm{d}T}\right)_\rho \tag{5.4}$$

Recalling that in the derivation of the used phase change term, the conservation of mass was used $-\frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}t} = \nabla\cdot u$ [83]. And as such, the used phase change term can be written as eq. (5.5)

$$\left(\frac{T}{\rho^2 c_v}\left(\frac{\mathrm{d}P}{\mathrm{d}T}\right)_\rho \frac{\mathrm{d}\rho}{\mathrm{d}t}\right)\Delta t = \frac{T}{\rho^2 c_v}\left(\frac{\mathrm{d}P}{\mathrm{d}T}\right)_\rho \mathrm{d}\rho \tag{5.5}$$

The only difference between eq. (5.4) and eq. (5.5) is the term $c_v$. This means that the phase change term used in the simulation is equal to $\frac{h_{fg}}{c_v}$. The equation of state used thus also determines the latent heat of vaporization.

To calculate the latent heat predicted by the equation of state, it is important to realize that the simulation uses a different latent heat depending on the local density. The $\left(\frac{\mathrm{d}P}{\mathrm{d}T}\right)_\rho$ is a function of density, and thus only valid for a specific density. Therefore to calculate the EoS latent heat, eq. (5.4) needs to be evaluated via integration, eq. (5.6).

$$h_{lv} = T\int_{\rho_v}^{\rho_l} \frac{1}{\rho^2}\left(\frac{\mathrm{d}P}{\mathrm{d}T}\right)_\rho \mathrm{d}\rho \tag{5.6}$$

To validate that eq. (5.6) does indeed produce the latent heat, the dimensionalized Peng-Robinson equation of state is used. Either the values $a = 599.40$, $b = 0.01895$, $\omega_{EoS} = 0.344$ and $R = 8314/18.015$ from Table A.7 [157] are used, or those parameters are calculated from the critical point of water ($T_c = 647.4$K, $P_c = 22.1$MPa, $\omega_{EoS} = 0.344$ and $R = 8314/18.015$) [106, 157].

The calculation and evaluation of the PR EoS latent heat is performed using the python code with the relevant code presented in appendix F. The latent heat is then printed, which resulted in $2334.7\cdot10^3$ J/kg. The temperature ratio used is 0.577, which with $T_c = 647.4$K results in 100.4°C. According to NIST [106], the difference in enthalpy between saturated water vapour and liquid at 373K is $2675.3 - 418.5 = 2256.8\cdot10^3$ J/kg. Comparing the calculated latent heat to the latent heat of vaporization gives a percentage error of 3.45%. This can be attributed to inaccuracy of the EoS. Given the general inaccuracies of defining a fluid by an equation of state, the predicted latent heat has a very good agreement with the latent heat given by NIST.

A similar approach was performed in the appendix of [43], and while they reached a similar yet different expression in eq. (5.6), their resultant latent heat from evaluating the PR EoS at 100.4°C is $2337.4\cdot10^3$. The 0.12% difference between their method and one mentioned in eq. (5.6) can be attributed to their lack of mentioning which critical point values they took when calculating the latent heat.

Using the non-dimensionalized Peng-Robinson EoS and evaluating eq. (5.6) at 100°C (Tr = 0.577) for different values of $a$, while keeping $b = 2/21$, $R = 1$ and $\omega_{EoS} = 0.3443$ results in the following latent heats table 5.4. The lattice unit to physical conversion is done with a factor $\frac{\Delta x}{\Delta t} = 210$.

Table 5.4: Latent heat of vaporization derived from equation of state with improper implementation

| Parameter $a$ | Lattice $h_{lv}$ | Physical $h_{lv}$ [$\frac{\mathrm{kJ}}{\mathrm{kg}}$] |
|---|---|---|
| 0.5/49 | 39.7 | 777.4 |
| 1.0/49 | 79.3 | 3498 |
| 2.0/49 | 158.6 | 6996 |
| 3.0/49 | 238.0 | 10494 |

It can be seen that the parameter $a$ is very impactful on the resultant latent heat. From this table, it would seem as if to match the latent heat of vaporization for water a value for $a$ in between $a = 0.5$ and $a = 1.0$ should be taken. However eq. (5.6) is independent of the simulation environment, meaning that if $\frac{\Delta x}{\Delta t}$ changes, for example due to a different temporal resolution being used, the physical latent heat changes as well. This indicates that the straightforward implementation into OpenLB was not correctly non-dimensionalized.

### 5.3.2. OpenLB non-dimensionalized equation of state implementation

It is useful to think about three scales for the fluid properties. The physical scale, the lattice scale and the equation of state scale. The calculations performed in OpenLB should all be lattice units. However, when setting up the equation of state with $b = 2/21$, $R = 1$, one introduces a new set of units.

The Peng Robinson equation of state predicted density is dominated by its critical density $\rho_c$, which is purely a function of $b$ as shown in eq. (5.7). In the pseudopotential method, only the density is used, and a density ratio of 1:1 between lattice density and EoS density is chosen. Setting $R$=1 is then a matter of simplicity and the parameter $a$ is free to choose to adjust the interface thickness.

$$\begin{aligned} \rho_c &= \frac{p_c}{0.3074RT_c} \\ p_c &= \frac{0.0778RT_c}{b} \\ \rho_c &= \frac{0.0778RT_c}{0.3074bRT_c} \\ &= \frac{0.0778}{0.3074b} \end{aligned} \tag{5.7}$$

However, the phase change term uses more than just the density from equation of state: $\left(-\frac{T}{\rho c_v}\left(\frac{\mathrm{d}P}{\mathrm{d}T}\right)_\rho \nabla\cdot u\right)\Delta t$. In that case, it is wrong to assume that the lattice units and equation of state units have a 1 to 1 ratio, as is evident by the latent heat dependence on the grid detailed in table 5.4. This is especially true for handling the temperature since OpenLB temperature scale is offset by 0.5. For clarification, in OpenLB, the lowest non-dimensional temperature is represented by 0.5 and the highest by 1.5.

To correctly implement the latent heat calculation, two approaches are presented. The first relies on using conversion factors to convert from the EoS unit scale to the Lattice unit scale. The other is to use a non-dimensional $R$ value in the EoS to pre-emptively perform the required conversions. Hand on examples for both methods are shown in appendix F.

The conversion factor method requires the multiplication of the derivative term with the following $\frac{p_{c,real}}{p_{c,eos}}\frac{\rho_{c,eos}}{\rho_{c,real}}$ and $T$ is in EoS units $T = Tr\cdot T_{c,eos}$. That results in the overall equation shown in eq. (5.8), where $Tr(T_{lb})$ is a function to convert the lattice temperature into the temperature ratio.

$$Tr(T_{lb})T_{c,eos}\left(\frac{\mathrm{d}p_{eos}}{\mathrm{d}T}\right)_\rho \frac{p_{c,real}}{p_{c,eos}}\frac{\rho_{c,eos}}{\rho_{c,real}} \tag{5.8}$$

The non-dimensionalized R method uses a different value for $R$, as defined by: $R = \frac{\mathbb{R}}{M} * T_{c,real}\left(\frac{\Delta x}{\Delta t}\right)^{-2}$, where $\mathbb{R}$ is the universal gas constant and $M$ the molar mass. With this value of $R$, the correct latent heat can be recovered from the equation of state without a need for the conversion factors, only requiring the usage of the $Tr(T_{lb})$ function, as shown in eq. (5.9).

$$Tr(T_{lb})\left(\frac{\mathrm{d}p_{eos}}{\mathrm{d}T}\right)_\rho \tag{5.9}$$

The complete source term to implement into the lattice is often written as eq. (3.22), repeated below for convenience.

$$T\left[1-\frac{1}{\rho c_v}\left(\frac{\mathrm{d}p_{eos}}{\mathrm{d}T}\right)_\rho\right]\nabla\cdot\mathbf{u}$$

It is important to note that for the OpenLB implementation the T refers to two different temperature scales. The term $T\nabla\cdot\mathbf{u}$ uses the $T_{LB}$ and the term containing $\left(\frac{\mathrm{d}P}{\mathrm{d}T}\right)_\rho$ uses the $T_{eos}$.

This follows from the origin of the two $T$. The $T\nabla\cdot\mathbf{u}$ is needed due to the rewriting of $\mathbf{u}\cdot\nabla T$ of the macroscopic temperature equation into $\nabla\cdot(T\mathbf{u}) - T\nabla\cdot\mathbf{u}$, since LBM solves for $\nabla\cdot(T\mathbf{u})$. Thus, that $T$ is in the lattice unit

scale. The $T$ of the second term appears from the derivation of the local entropy balance to encompass the energy loss due to phase change, which is expressed using the EoS, and thus that $T$ is in the EoS unit scale.

To show that the value of $R$ does not impact the previous simulations, the 2D thermodynamic consistency test for $\sigma = 0.35$ and $a = 0.5/49$ are rerun, but with $R = 7.47$. They are compared to the ones with $R = 1$ ones in figs. 5.17 to 5.20.

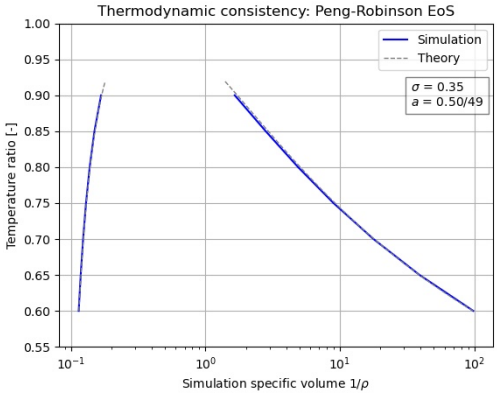



Figure 5.17: P-V plot of results from 2D thermodynamic consistency using PR EoS with $R = 1$, $\sigma = 0.35$ and $a = \frac{0.5}{49}$

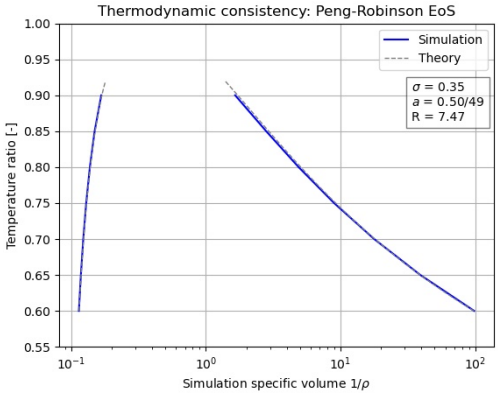



Figure 5.18: P-V plot of results from 2D thermodynamic consistency using PR EoS with $R = 7.47$, $\sigma = 0.35$ and $a = \frac{0.5}{49}$

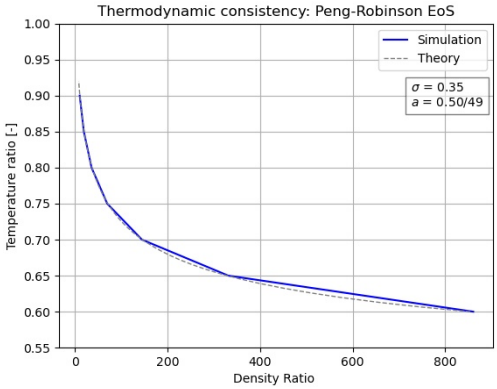



Figure 5.19: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $R = 1$, $\sigma = 0.35$ and $a = \frac{0.5}{49}$

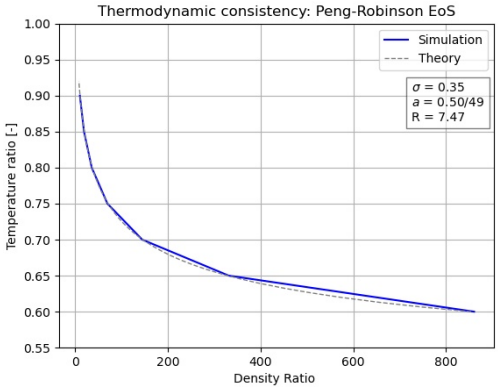



Figure 5.20: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $R = 7.47$, $\sigma = 0.35$ and $a = \frac{0.5}{49}$

# 6 Simulation Tool Discussion

In this chapter, further aspects of the simulation tool beyond verifications are discussed. The tool is used to simulate nucleation in pool boiling. In addition, the impact of spurious currents on the thermal lattice are investigated. A plan for the full validation of the simulation tool is presented.

## 6.1. Impact of spurious currents

In addition to the $D^2$-law, section 5.2 showed a few more aphysical behaviour. These aphysical behaviour are found to occur due to spurious currents, discussed in section 4.2.2. The spurious current interaction with the thermal lattice is investigated in more detail, this is not covered in literature. Three phenomena are shown, the thermal cross, the spurious condensation and evaporation and numerical explosive boiling.

The explosive phenomena occurred in simulations where the $a$ parameter of the EoS was taken to be 3/49, following the values used in literature. Through numerical trials, the phenomena vanished when using a value of $a = 0.5/49$.

### 6.1.1. Thermal cross

The first impact the spurious currents have on the thermal lattice results in a cross shape in the temperature field, shown in fig. 6.1. Since the temperature is advected along with the velocity, the spurious currents transports cold fluid out. The fluid is then heated near the wall, and subsequentially carried back in. This means that at locations where the spurious currents are flowing out, the vapour temperature is colder, while where the currents flow in, it is warmer.

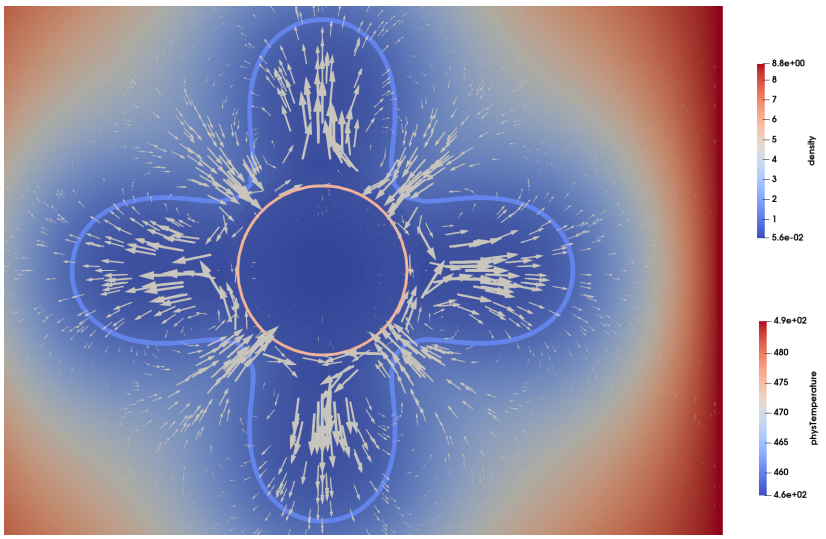



Figure 6.1: Spurious currents carrying cold vapour away and warm vapour in, resulting in a thermal cross

### 6.1.2. Spurious condensation and evaporation

For the phase change term, the change in density is represented by the term $\nabla \cdot \mathbf{u}$. This means that the spurious currents also produce spurious condensation and evaporation. The velocity field and magnitude as shown in

fig. 6.2, where the arrows purely detail direction and the background the magnitude. The 4 concentric circles are the density contours at $\rho = 0.2$, $\rho = 1$, $\rho = 4$ and $\rho = 7.5$. The contours are chosen such that they overlap with the maximum velocity locations.

Given a phase interface thick enough, multiple eddy currents can arise within the interface, as in fig. 6.2. The maximum velocity locations within the diffuse interface are at the location where two eddies flow alongside each other. Thus the contours segment the eddies. In other words, between each contour exists an eddy.

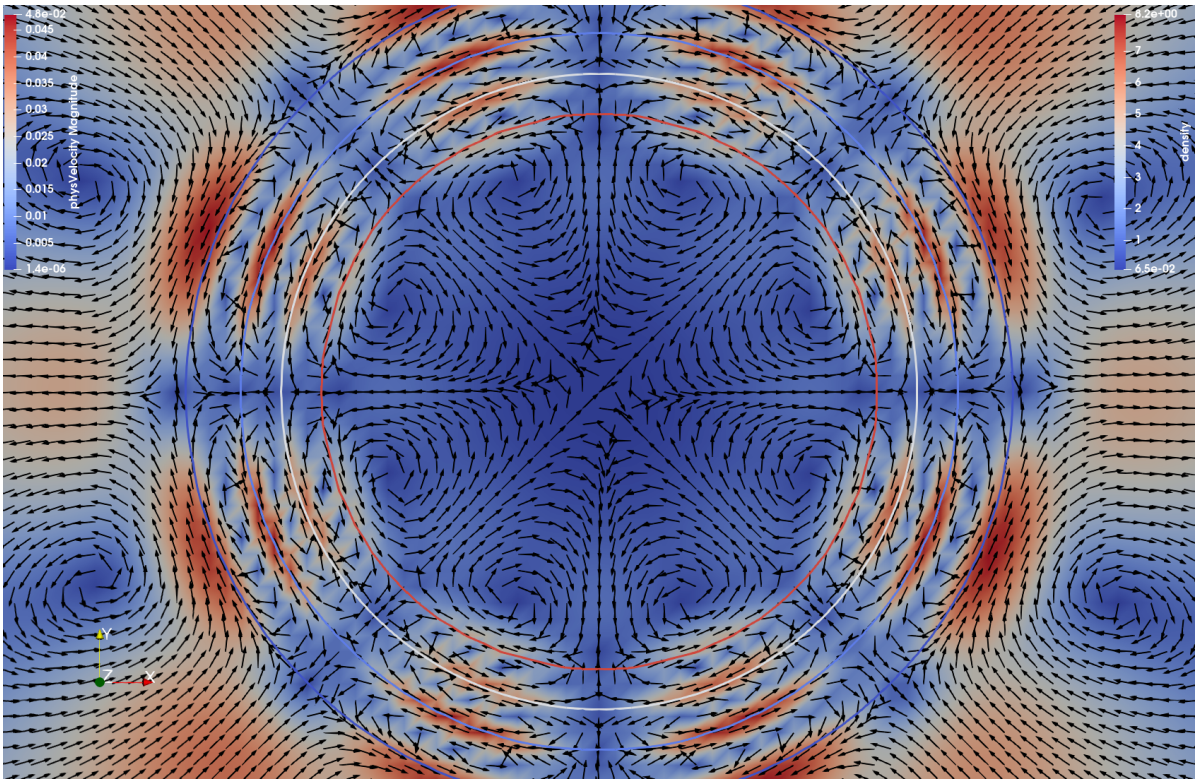



Figure 6.2: Detailed velocity field of the spurious currents inside the droplet. Contours are shown for $\rho = 0.2, 1, 4, 7.5$

The spurious effects on the temperature field is seen fig. 6.3. In the regions where $\nabla \cdot \mathbf{u}$ is positive due to spurious currents, i.e. more is flowing out than in, spurious evaporation occurs. This results in lower than expected temperature. The reverse is true when $\nabla \cdot \mathbf{u}$ is negative, resulting in spurious condensation.

Spurious evaporation and the effect on the thermal lattice is very much visible at the diagonals inside the $\rho = 7$ contour. In short, this is due to it being a local node where there is more fluid outflow than inflow. This is occurring at any location where multiple eddies meet and forming a local outflow, shown in fig. 6.4. When the eddies form a local inflow node, the temperature is slightly elevated.

There is an nuanced answer to why the diagonals inside the $\rho = 7$ contour the droplet is colder than average. To see this, one has to follow the streamline from the centre of the droplet out.

At the centre, the streamlines point along the axis. Following left along the negative x axis, the streamline comes to a stop at $\rho = 7$ as it meets an inflowing eddy. Thus the streamlines turns either up or down along the y axis. Following the streamline, one reaches the diagonal cold spot. At this cold spot, the streamline either follows the internal eddy back to the droplet centre, but some flows out to join the eddy between $\rho = 4, 7$. This outflow is the evaporation.

A theoretical package of fluid would need to follow four eddies to flow out of the droplet. At each eddy there is a point of evaporation where the density changes. However, the latent heat needed/given to expand/contract between each of these densities differ. The eq. (5.6), repeated below for convenience, can be used to calculate the latent heat, shown in table 6.1. The contour densities are used for the density ranges.

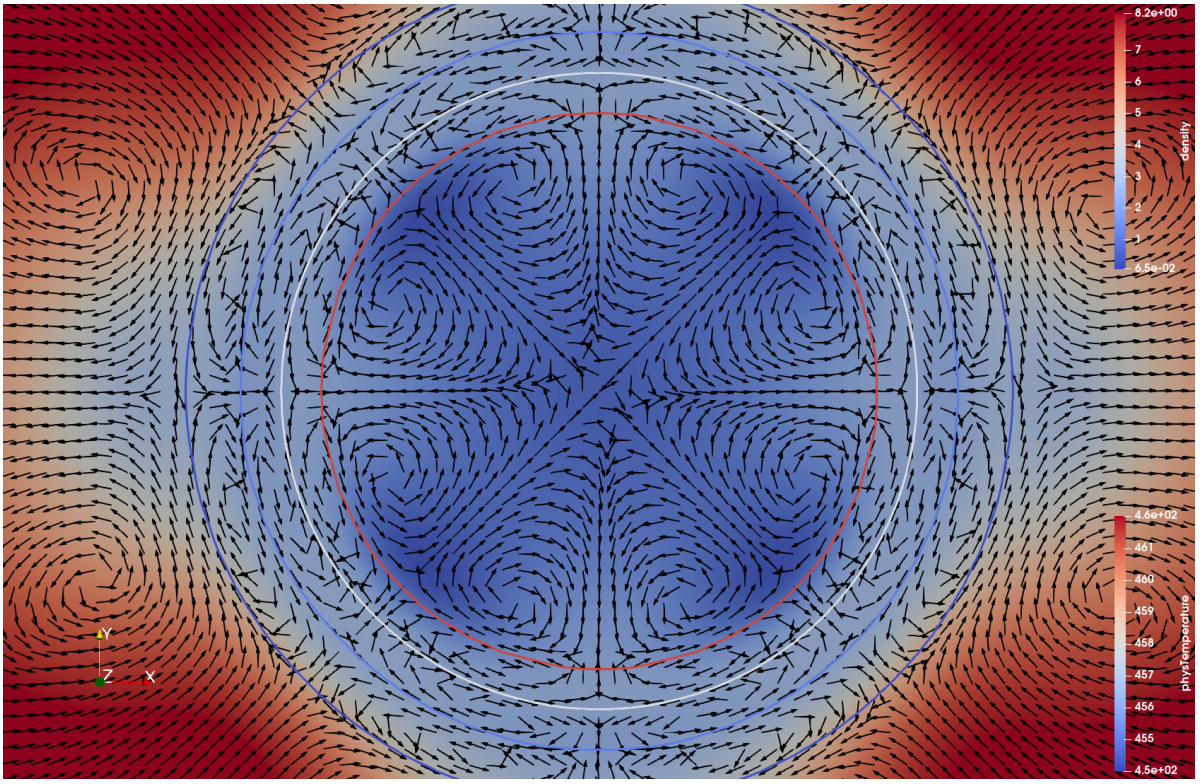



Figure 6.3: Temperature field of droplet, showing the impact of spurious currents on thermal lattice. Contours are shown for $\rho = 0.2, 1, 4, 7.5$

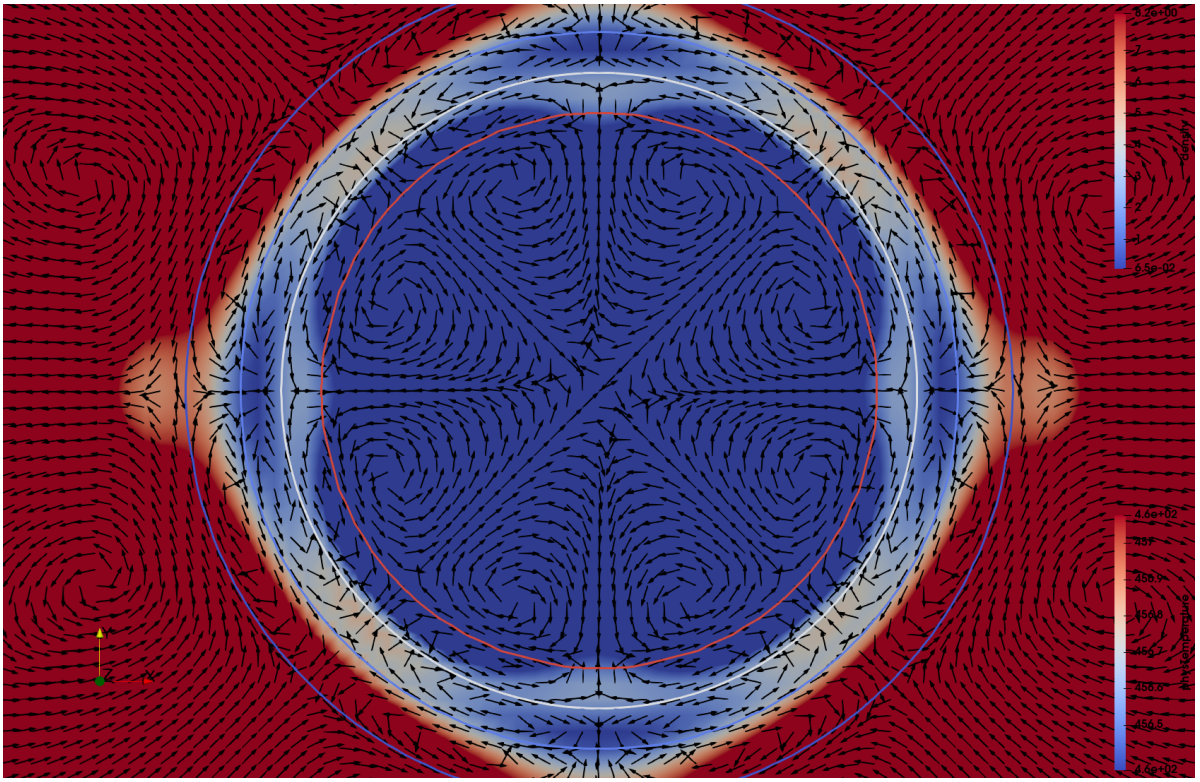


Figure 6.4: Temperature field of droplet detailing the phase interface, showing the impact of spurious currents on thermal lattice. Contours are shown for $\rho = 0.2, 1, 4, 7.5$

$$h_{lv} = T \int_{\rho_v}^{\rho_l} \frac{1}{\rho^2} \left( \frac{\mathrm{d}P}{\mathrm{d}T} \right)_{\rho} \mathrm{d}\rho$$

Table 6.1: Stepwise latent heat of vaporization for water using PR at $Tr = 0.7$, $b = 2/21$, $M = 18.0152$

| Density range | Physical $h_{lv}$ $[\frac{\mathrm{kJ}}{\mathrm{kg}}]$ |
|---|---|
| 8.08 to 7.5 | 102.5 |
| 7.5 to 4 | 567.4 |
| 4 to 1 | 668.2 |
| 1 to 0.2 | 454.2 |
| Sum above | 1792.3 |
| 8.08-0.2 | 1792.3 |
| 8.08-0.056 | 2082.5 |

Each time an eddy transports a fluid package from one contour to the next, they release / take an amount of energy shown in table 6.1. For example, from contour $\rho = 4$, fluid travels to contour $\rho = 7.5$, releasing 567.4 kJ/kg, and travelling to $\rho = 1$, takes 668.2 kJ/kg. This can be seen in fig. 6.4 where there is an increase in temperature between contours $\rho = 4, 7.5$ and a decrease between $\rho = 4, 1$.

Inside the interface, the contours are close together, and an eddy where evaporation occurs is directly opposite to an eddy where it condenses. Between the four contours, the latent heat are roughly equal. As such diffusion can make it seem as if the temperature inside the interface is uniform, while it is not. This also answers the question why there are 4 diagonal cold spots. It is a node of where energy loss due to evaporation is not overcome by the energy gain due to condensation.

### 6.1.3. Numerical explosive boiling

The second phenomena is what could be called numerical explosive boiling. During the tests of the $D^2$ law, once the temperature was raised just slightly, the droplet starts to move. This movement is sporadic, and increases with simulation time. This is not a phenomena specific to only droplets, but also channel flow.

The stationary droplet is shown in fig. 6.5, which shows the expected spurious currents surrounding the droplet. Only slightly later, the pattern changes, with the spurious currents exiting the droplet to the right and bottom nearly vanishing fig. 6.6, this gives the droplet a momentum and the droplet starts to travel slightly into that direction, as shown in fig. 6.7.

The velocity field soon after flips, and another interesting change occurs. The spurious current of 4 outgoing and 4 incoming streams changes to predominantly 2 outflowing and 3 incoming.

The current pattern keeps on changing till it is no longer confined to the field near the droplet. This is shown in fig. 6.9, where the outflow to the top of the droplet loops over the periodic boundary and flows to the two inflows at the bottom. Since the boundary for the thermal lattice has a fixed temperature, the inflowing temperature is increased, thus bringing more energy to the system. This increases the movement energy available to the droplet. Thus the velocity currents become increasingly sporatic and more energetic as shown in figs. 6.10 to 6.14.

In fig. 6.14 the droplet crosses over the periodic boundary, which instantly increases the core temperature. At that point the evaporation becomes very quick and the outflow velocity of the droplet increases drastically. This increase can cause crashes in the simulation, since a comparatively too large timestep is taken for the occurring velocities.

This numerical explosive behaviour is not unique to the test case and also occurred in all other simulations.

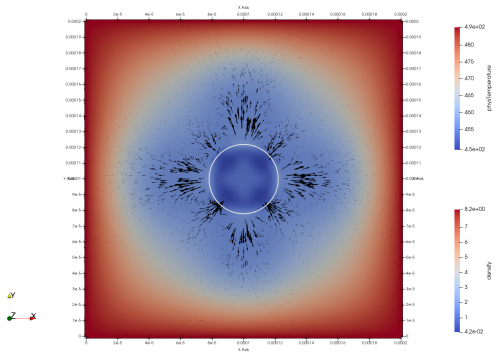


Figure 6.5: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 10e3$

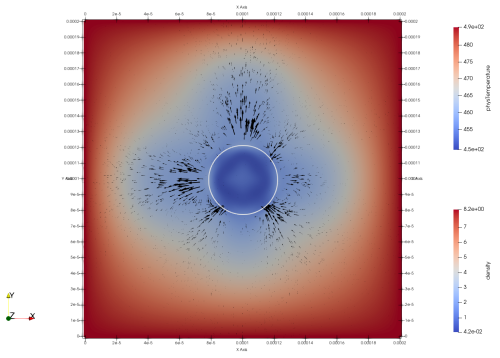


Figure 6.6: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 14e3$

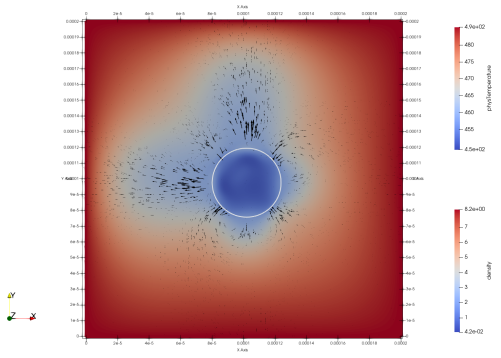


Figure 6.7: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 20e3$

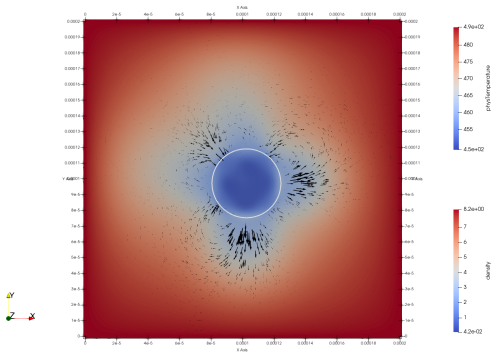


Figure 6.8: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 25e3$

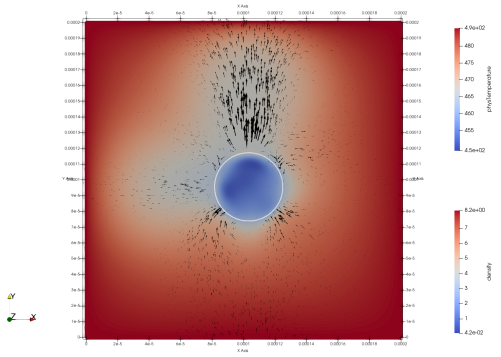


Figure 6.9: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 40e3$

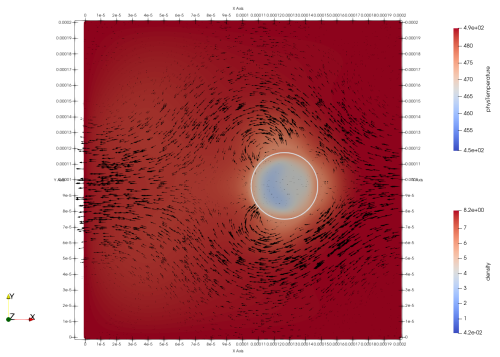


Figure 6.10: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 100e3$

An example is shown for a multi-channel simulation. Two snapshots are taken, one before the liquid flows into the channel, shown in figs. 6.15 to 6.17, and one with liquid in the channels, shown in figs. 6.18 to 6.22. The inflow is a manual set density and velocity profile. The outlet has the same velocity profile and speed, except scaled to the density ratio. This can lead to densities below the equilibrium density near the outlet, but it should not cause this phenomena, since it occurs in other simulations as well.

The velocity magnitude and direction of figs. 6.15 to 6.22 are shown with the black arrows. They are all at the same scale. It can be seen that in fig. 6.16, the most open explosive numerical boiling, generated a dispropor-

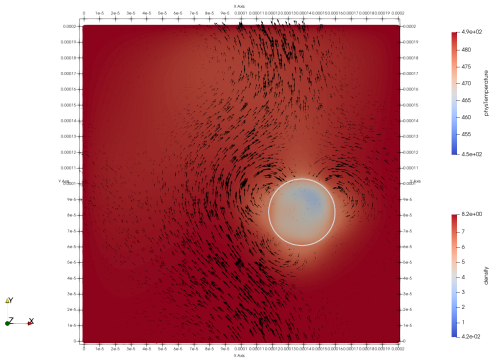


Figure 6.11: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 110e3$

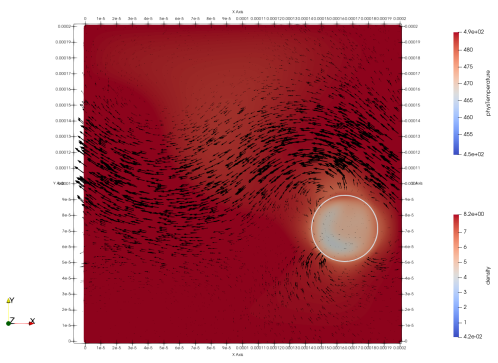


Figure 6.12: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 120e3$

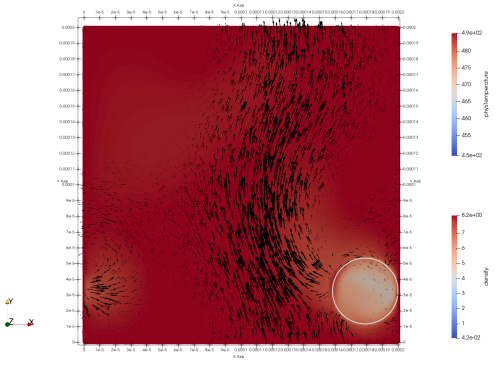


Figure 6.13: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 130e3$

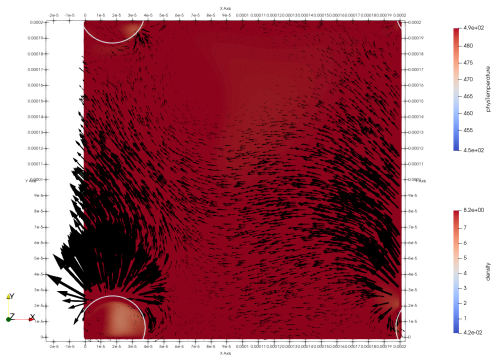


Figure 6.14: D2-law simulation showcasing numerical explosive boiling simulation with $R = 15.2$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 140e3$

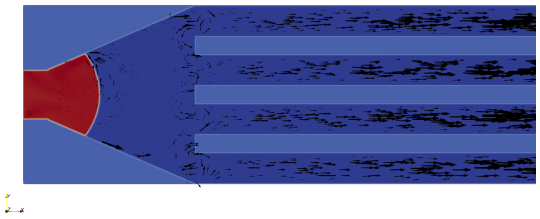


Figure 6.15: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 120e3$

Figure 6.16: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 130e3$

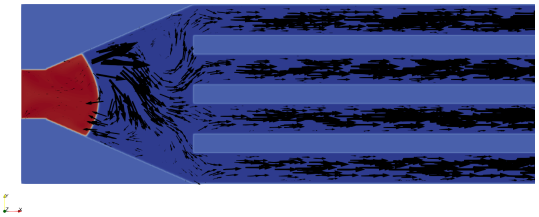


Figure 6.17: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 140e3$

tionally huge amount of velocity. The liquid front, which was a single curve becomes distorted. In fig. 6.16, it can be seen that the location of this numerical explosive boiling happened more at the top of the liquid front. The location of where the numerical explosive boiling occurs oscillates from the top to the bottom, aggravating the liquid front to sway with it.

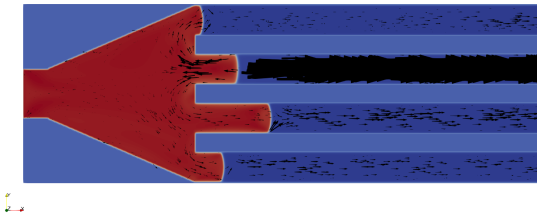


Figure 6.18: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 1000e3$

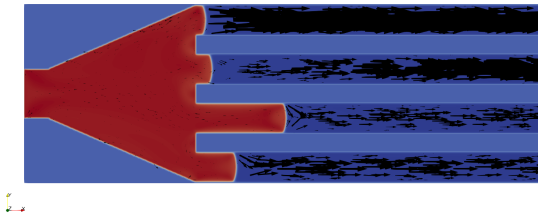


Figure 6.19: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 1010e3$

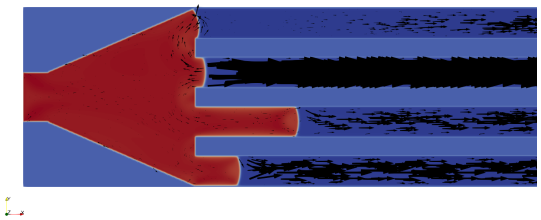


Figure 6.20: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 1020e3$

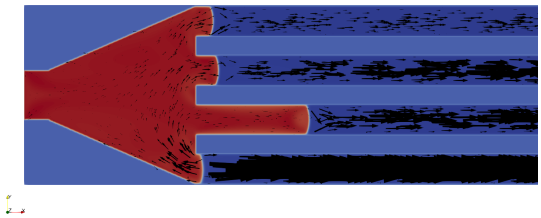


Figure 6.21: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 1030e3$

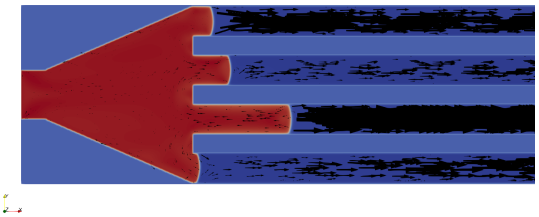


Figure 6.22: Multi channel simulation showcasing numerical explosive boiling simulation with $R = 1$, $\sigma = 0.30$ and $a = \frac{3}{49}$ at $\Delta t = 1040e3$

In figs. 6.18 to 6.22, the same phenomena occurs, however it is contained inside the channels. There is nothing new to say regarding this phenomena from this series, however, it is shown that multichannel interactions does occur, with liquid distributing into other channels due to backflow.

## 6.2. Pool Boiling

A very useful benchmark that is often used to showcase phase change is pool boiling. In pool boiling, a partially filled container is heated from the bottom. The expected behaviour is that the liquid starts to boil, with nucleation sites forming, bubble growth, and bubble release.

### 6.2.1. Simulation layout

The geometry for pool boiling is given in figs. 6.23 to 6.25. The geometry nodes that are 0, given by the white circles, are ghost nodes and have no dynamics. The nodes with geometry equal to 1 are fluid nodes. The

nodes with geometry equal to 2 (bottom) and 3 (top) are walls using bounce back scheme, where the bottom wall is neutral, making a contact angle of 90° and the top wall hydrophobic. Thermally, the bottom wall (geometry = 2) is a constant temperature wall which is increased after the liquid has come to rest. The top wall is a constant temperature wall which stays constant throughout the simulation. The sides are periodic.

The exact length and width of the simulation changes, however the method how the boundaries are set up does not change. The temperature lattice is held at a constant temperature while the fluid finds its equilibrium.

In addition, gravity is taken into account. The method for including gravity is explained in section 2.3.6, with the implementation detailed in section 3.7. For these simulations, the average density is used as a reference density. This method adds no net momentum to the simulation.

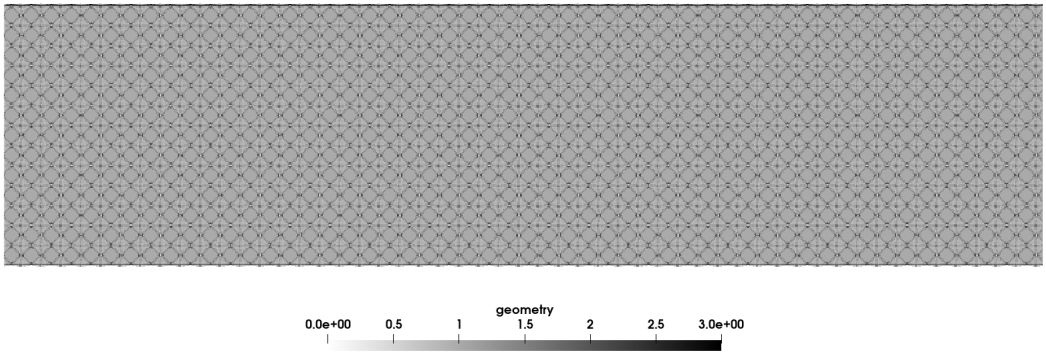



Figure 6.23: Pool boiling geometry showing full simulation domain

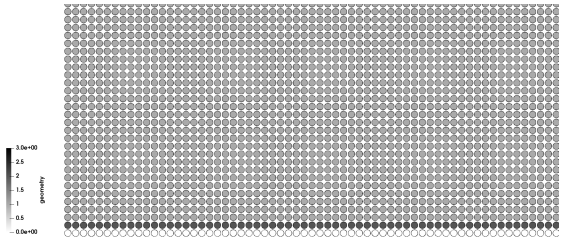


Figure 6.24: Pool boiling geometry showing bottom left corner

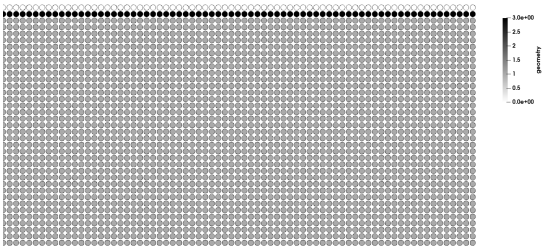


Figure 6.25: Pool boiling geometry showing top right corner

### 6.2.2. Heat transfer at wall

The variable wall wettability is achieved by giving the wall a fictitious density. This however has an unwanted effect on the wall heat transfer. The relevant aspects of the thermal diffusion, see section 3.5.2, is repeated below for convenience.

$$\frac{\nabla(\lambda\nabla T)}{\rho c_v} = \frac{\nabla\lambda\nabla T}{\rho c_v} + \frac{\lambda\nabla^2 T}{\rho c_v}$$

$$\frac{\nabla\lambda\nabla T}{\rho c_v} = \frac{\frac{\partial\lambda}{\partial\rho}\nabla\rho\nabla T}{\rho c_v}$$

The first term on the right hand side corrects for a changing thermal conductivity, which occurs in the phase interface. Since thermal conductivity for the vapour and liquid are defined by the local density, the term can be rewritten in term of changing density. Since the thermal conductivity scales linearly and for water the liquid conductivity is higher than for vapour, the term $\frac{\partial\lambda}{\partial\rho}$ is a positive constant.

By giving the wall a density to define its wettability, one also changes the wall heat transfer to the fluid. For example, if liquid water is in contact with the wall, the $\nabla\rho$ is positive. A wall which is hotter than the fluid $\nabla T$ is negative. Thus in total the sign is negative, indicating a decreased heat transfer from the wall to the fluid.

### 6.2.3. The nucleation problem

One issue that was uncovered by the pool boiling simulation attempts is the nucleation - or the lack thereof - problem. First a common pool boiling simulation is given which shows that while enough heating exists, no nucleation occurs. Then an explanation for the lack of nucleation is given, providing both theory and explanation.

#### Lack of nucleation

An issue that occurred is that the pool boiling simulations showed no nucleation. For example, fig. 6.26 and fig. 6.27 shows the simulation state after 4.3 ms real time for a 0.25 mm times 0.8 mm big channel. The initial temperature is 485 K with a hot wall temperature of 550 K, a difference of 65K. Gravity is pointing downwards with 9.81 m $s^{-2}$. This should be sufficient for nucleation to occur, but only convection can be seen.

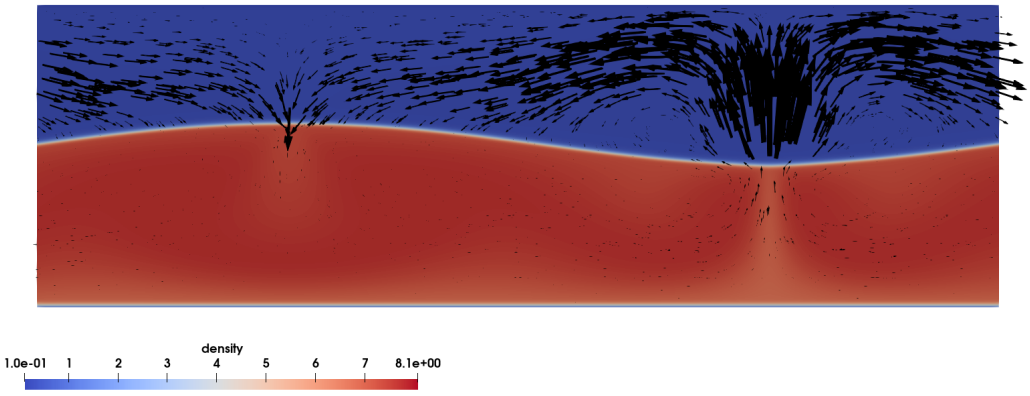



Figure 6.26: Pool boiling without nucleation: density lattice with velocity field shown by arrows scaled by velocity magnitude

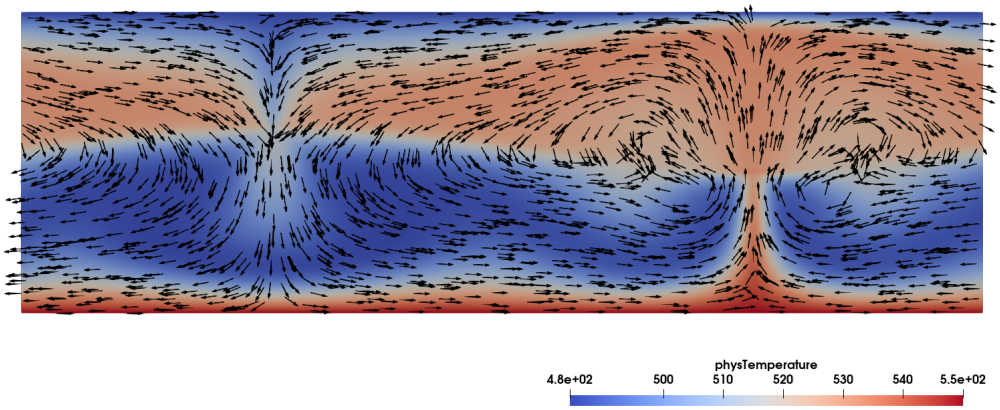



Figure 6.27: Pool boiling without nucleation: temperature lattice with velocity field shown by arrows not scaled

### Metastable Region

In a recent paper (published online September 2020) *How does boiling occur in lattice Boltzmann simulations?* by Li et al.[84], they presented their knowledge gained from their previous work along with theoretical background to support it. They also provided their insight into the conditions for when nucleation occurs.

The problem of no nucleation is due to the simulation state being in the metastable region, which is shown in fig. 6.28 [84]. The "[...] spinodal name [is] given to the two curves that border the instability domain." [5]. And while Li et al. draw good conclusions regarding how this impacts LBM simulations, the physical relevance can be explained better.

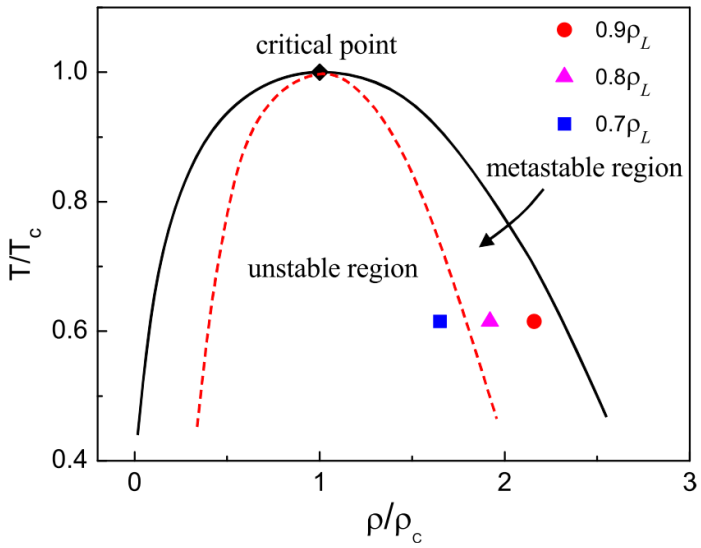



Figure 6.28: "Sketch of a phase diagram with the coexistence curve (solid line) and spinodal line (dashed line)." [84]

"Among the practical issues that motivate the study of metastable states — and, in particular, the pinpointing of the spinodal lines — is the question of how much a liquid can be heated above its saturation temperature before it has no choice but to boil." [5]

"If the liquid is free of impurities that usually act as sites for the formation of the first vapor bubbles (“nucleation” sites), the temperature of the liquid can increase above the corresponding saturation temperature, $T_{\text{sat}}(P_f)$, while the liquid remains single phase. We know from the stability considerations centered on [fig. 6.29] that this process cannot continue to temperatures that might exceed the liquid spinodal temperature corresponding to $P_f$: namely, $T_{\text{sp}}(P_f)$ in [fig. 6.29]." [5] The figure referenced in the quote is shown in fig. 6.29 and the quote was adapted accordingly.

A similar metastable concept that is more commonly known is the supercooling of a liquid, for example water. Given no impurities, water can be cooled below its freezing point while remaining in its liquid form. While in such a supercooled state, any disturbance results in the water quickly freezing.

Only one other set of authors, Hazi and Markus, from all the literature read for this thesis mentioned the metastability. They said: "the thermodynamic fluctuations need to be high enough to drive the system locally into the metastable region. Then phase transition can occur. Note, however, that the metastable state is between the spinodals, so the fluctuations must high enough to achieve this region. Therefore, in order to model boiling in a real fluid, like water by this method we need a specific equation of state (modified between the binodals) and we need a corresponding potential given in analytical form. The development of such potential is in progress and results obtained for water will be published in the future. Here we limit the scope of our discussion to a model fluid defined by the equation of state" [53]

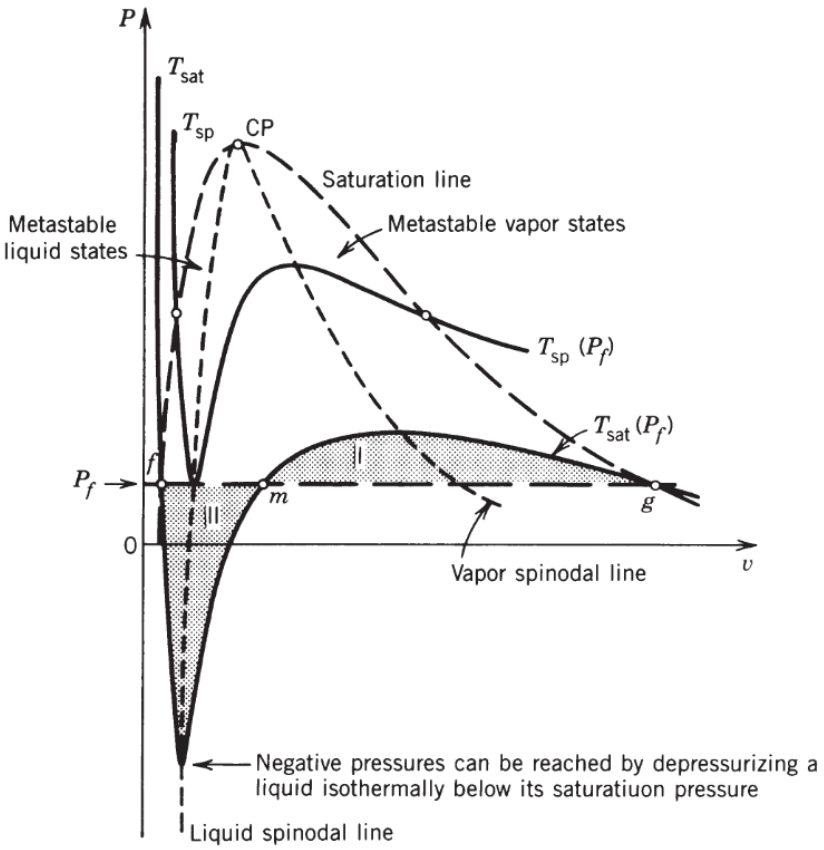



Figure 6.29: Maximum temperature ($T_{sp}$) that can be reached during the isobaric ($P_f$) heating of a saturated liquid.[5]

### Nucleation in LBM

Li et al. conclude that "a considerable decrease of the fluid density near the heating surface may lead to the liquid–vapor phase transition"[84]. And "a faster decrease of the fluid density near the heating surface results in an earlier liquid–vapor phase transition" [84].

This explains why nucleation was so difficult to achieve. To limit the spurious current impact, a thicker interface width was taken, by using a lower $a$ in the EoS. This thicker interface width resulted in the density gradient near the wall to be shallow.

Also unhelpful is the limited impact gravity has in microchannels. For the simulation above, the non-dimensional gravity has an order of magnitude in range $10^{-9}$ to $10^{-10}$, depending on the exact time step taken. However, the non-dimensional gravity used for pool boiling in literature is order of $10^{-5}$ [41, 43, 45, 53, 79, 82, 84]. The increased gravity effectively squishes the interface, thus increasing the likelihood of nucleation to happen.

Gong and Cheng [44] managed to simulate flow boiling in microchannels neglecting gravitational forces, however they are using the standard DDF for thermal diffusion, which limits the range of the thermal transport. Thus their thermal transport is comparative to the one used in the pool boiling examples much higher.

In the literature covered, the most common boundary used was the Zuo-He non-equilibrium BB method with a true fictitious density that only impacts the pseudopotential force. The no-slip BB method currently used sets a fixed density at the wall, thus at the wall there is a gradient from the wall density to the density away from the wall. Using the Zou-He boundary with the truly fictitious wetting implementation avoids this density gradient, and thus also help in the nucleation.

A wide phase interface, low gravitational force, and low thermal diffusion allows for the liquid near the wall to gently expand and reach the equilibrium density of the new temperature and thus stay within the metastable

region. This results in no nucleation.

### 6.2.4. Temperature fluctuation discussion

"To enhance bubble formation, small temperature fluctuations are added to the equation of state in the first grid point layer near the bottom solid plate." [154] This was carried over by Li et al. "small temperature fluctuations are initially added to the equation of state in the first grid layer near the bottom solid wall so as to enhance the bubble formation" [79]. However, neither specified how large these temperature fluctuations are. In another paper by Li et al. [82], they specified a nucleation site by setting three nodes to $Tr = 1.25$.

As such, it was investigated whether setting random temperature fluctuations to the wall temperature can aid in pool boiling. These simulations are purely qualitative to get a better understanding of pool boiling in LBM. The non-dimensionalized gravity used is 2e-7, which is higher than earth's gravity for microchannels but closer to the value used in literature.

### 6.2.5. Impact on heat transfer

For pure thermal transfer without phase change, only the wall average temperature matters. For example the simulation state of a hot wall made of pure fluctuations $T_w = 2(T_h - T_0) \cdot \text{RAND}$ is compared to the exact same simulation with a wall temperature $T_w = T_h$. fig. 6.30 shows a comparison of two simulations, where the figure is a composite from both simulations. The thermal temperature increase into the fluid are near equal. A contour of the liquid at $\rho = 7$ is shown, which shows that a similar amount of fluid has expanded.

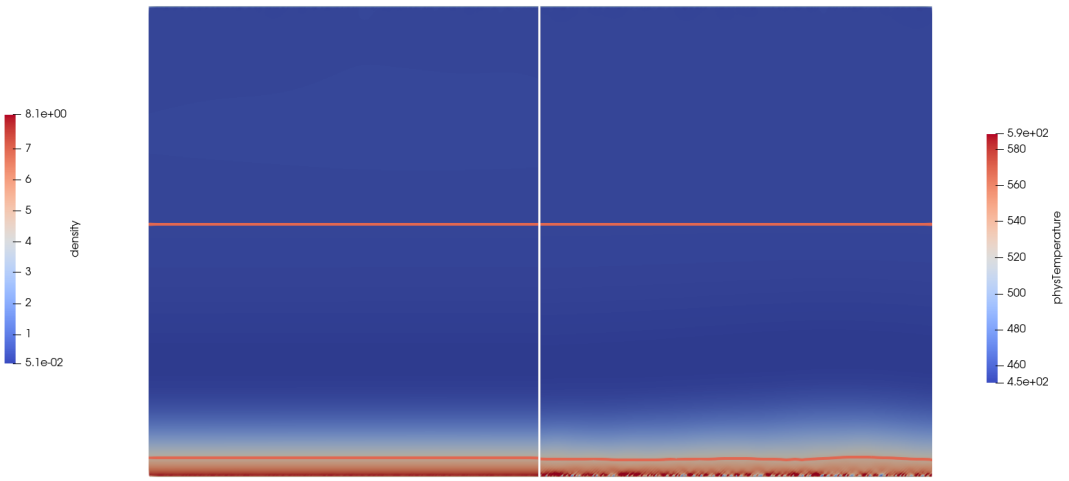



Figure 6.30: Comparison of two simulations with different hot wall implementation: constant temperature (left) vs pure fluctuations (right). Both walls have the same average temperature. Density contour at $\rho = 7$ is shown.

### 6.2.6. High temperature ratio

Most literature showcase pool boiling where the temperature ratio is already very high, $Tr > 0.8$. A simulation was performed with initial $Tr = 0.8$, and a hot wall temperature of $Tr = 0.9$ containing added temperature fluctuations up to $Tr = 1.2$. While the simulation is not in the targeted temperature range, it does show how pool boiling can emerge. The results are shown in the series figs. 6.31 to 6.39

Notable is that the bubble nucleation always occurs at the same locations. Most likely is that several nodes after each other had a high random value, thus creating bigger incentive for bubbles to form. It might be interesting to have temperature fluctuations that change each time step.

Two nucleation sites at 0.4e-3 and 0.5e-3, shown in fig. 6.35 merged together to form one bubble at 0.45e-3. The bubble at 0.5e-3 was larger when the merger happened and it caused a shift in all other vapour bubbles.

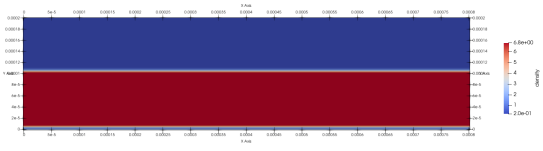


Figure 6.31: High temperature ratio pool boiling: Density initial condition

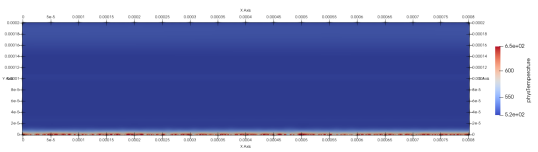


Figure 6.32: High temperature ratio pool boiling: Temperature initial condition

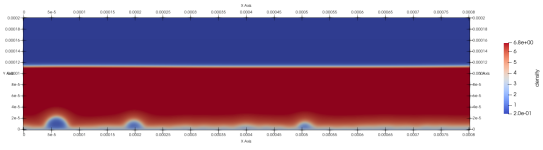


Figure 6.33: High temperature ratio pool boiling: Density $\Delta t = 200k$

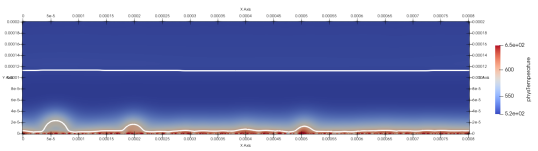


Figure 6.34: High temperature ratio pool boiling: Temperature $\Delta t = 200k$

Each bubble however shifted only so much such that the original nucleation site is still within the vapour bubble.

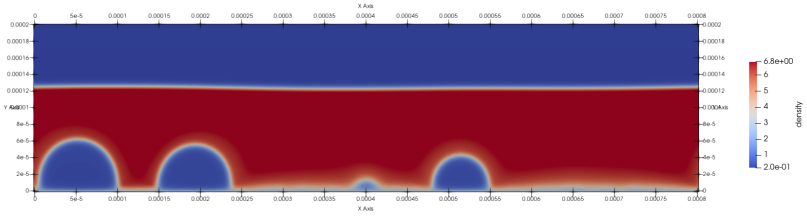


Figure 6.35: High temperature ratio pool boiling: $\Delta t = 240k$

In fig. 6.36 and fig. 6.37 the bubble breaches through to the top vapour layer. The other bubbles soon follow, and the simulation returns to a simulation state similar to the initial condition, with an albeit warmer top vapour layer and liquid.

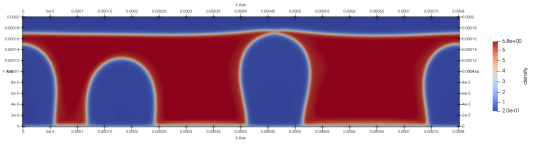


Figure 6.36: High temperature ratio pool boiling: $\Delta t = 380k$

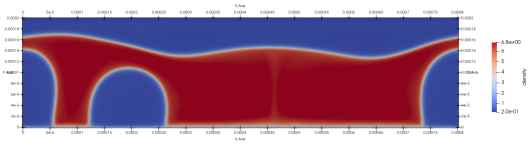


Figure 6.37: High temperature ratio pool boiling: $\Delta t = 400k$

The bubble nucleation, departure, and bursting follows a cyclic behaviour. The nucleation starts at $\Delta t = 200k$, $\Delta t = 740k$, and $\Delta t = 1400k$. The frequency decreases, which can be attributed to the lower density difference at the later time steps. In fig. 6.39, the average liquid density dropped from 6.8 to around 5, while the vapour density increased from 0.2 to 1.2. The frequency is given by the relation eq. (6.1), where $D_b$ is the bubble departure diameter eq. (6.2) [53].

$$\text{frequency} \propto D_b^{-1} \left[ \frac{\gamma g(\rho_l - \rho_v)}{\rho_l^2} \right]^{1/4} \tag{6.1}$$

$$D_b \propto \sqrt{\frac{\gamma}{g(\rho_l - \rho_g)}} \tag{6.2}$$

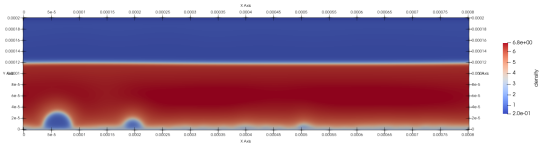


Figure 6.38: High temperature ratio pool boiling: $\Delta t = 740k$

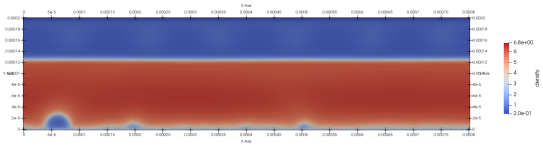


Figure 6.39: High temperature ratio pool boiling: $\Delta t = 1400k$

### Fluctuation with correct wall average

However, just adding fluctuations is not the answer, and can be detrimental as well. A showcase is shown with high temperature fluctuations reaching up to $1.2T_c$, but on average the wall temperature is $0.8T_c$. The initial temperature is $0.7T_c$. In this case, the bool boiling initiates a bubble, but the bubble is not released from the wall. A quasi-steady state is achieved, shown in fig. 6.40, with the corresponding temperature field shown in fig. 6.42. A more detailed zoom onto the bubble is shown in fig. 6.43 and fig. 6.44 for both density and temperature.

Figure 6.40: Quasi steady pool boiling with stuck bubble: density field

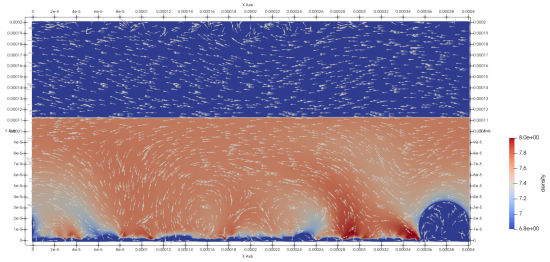


Figure 6.41: Quasi steady pool boiling with stuck bubble: liquid density variations with velocity field

The non-phase change interactions between hot and cold spots can be seen in fig. 6.42. The expected phenomena that warmer fluid with a lower density rises is shown, best seen at the hot spot around 7e-5, called hotspot 1. However, at the next hotspot around 13e-5, called hotspot 2, the opposite is true. This is due the overall interactions occurring in the liquid.

The rising liquid at hotspot 1 travels to the left, and liquid has to recirculate towards it from the right. A strong cold spot between hotspot 1 and 2 has a lower density and attracts more liquid from the surrounding. The recirculation and cold spot creates a stream downwards that is stronger than the rising currents of the hotspot 2. The warm liquid from hotspot 2 is thus pushed to the right, where there is a cold spot, cooling the liquid down, effectively neutralizing the hotspot.

A third hotspot is located at 24e-5. This showcases a self contained recirculation. The warm liquid rises and drags cold liquid from the coldspot to the right up. The two streams merge and flow to the left along with the general fluid flow. The cold liquid cools down the warm liquid, thus decreasing the buoyancy. The liquid then flows down to the same hotspot, creating a circular flow.

The zoomed in section of the bubble provides some interesting insight regarding how the fluctuations impacts bubble departure. The circular bubble velocity inside the bubble is also observed by others [31] during

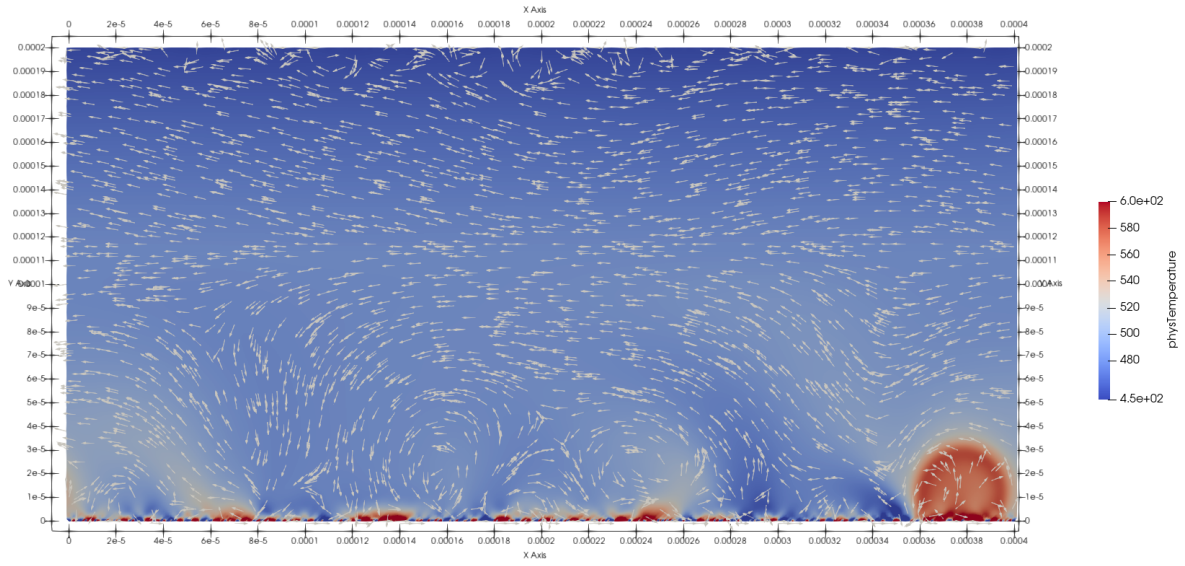



Figure 6.42: Quasi steady pool boiling with stuck bubble: temperature field

bubble evaporation on a heated surface. The bubble itself is formed on a large hotspot, which is bounded by two cold spots, and the bubble does not grow beyond those. The contact angle of the bubble shows the expected contact angle defined by the setup. With a bounded hot spot, and a given contact angle, the bubble has a defined size. This size should then be below the bubble departure diameter [53, 137].

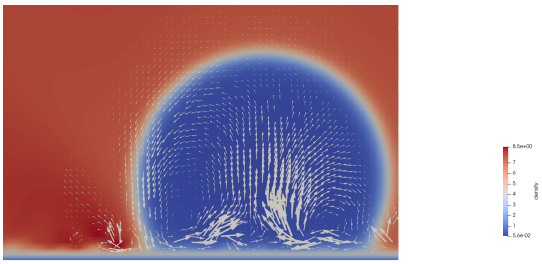


Figure 6.43: Quasi steady pool boiling with stuck bubble: density field

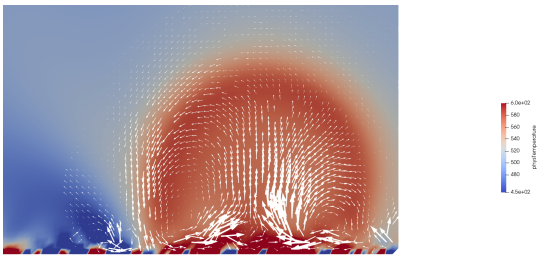


Figure 6.44: Quasi steady pool boiling with stuck bubble: temperature field

### 6.2.7. Bubble burst droplet generation

Using the above lessons, a new pool boiling simulation was set up. This simulation showed how droplets can form from bubble burst. The initial bulk temperature ratio is 0.7. The physical viscosity is $1e-5 m^2/s$, with a characteristic length $L$ of $100e-6m$, and a domain size of $8L$ long and $3L$ high. The non-dimensional gravity used was $-6.66e-6$.

The initial height of the liquid was lowered from 0.5 to 0.4. In previous simulations, the nucleation bubbles displaced enough liquid such that no more vapour remained in the top half. Lowering the ratio from 0.5 to 0.4 helped avoiding that issue.

The lower wall temperature ratio was raised to $0.9 \pm 2.5$ after the initial rest period. The random distribution was squared, such that lower random values are given less importance. The average wall temperature ratio was 0.895. A fictitious wall density of 5 was assigned to the lower wall.

Since the burst process happened very quickly, the detailed domain save missed the moment. However, the higher frequency figure save captured it quite well, but the colour scale used is different. The droplet

formation is shown in figs. 6.45 to 6.50.

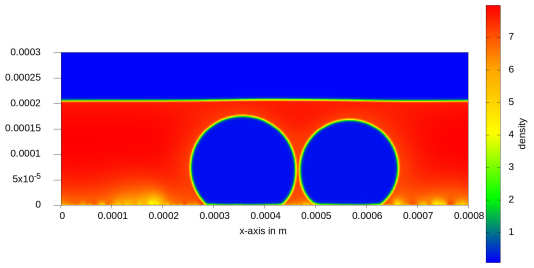



Figure 6.45: Two nucleation bubbles are growing next to each other and are about to merge. $\Delta t = 95e3$

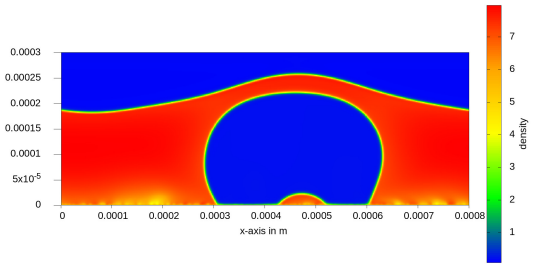



Figure 6.46: The nucleation bubble after merging together. $\Delta t = 105e3$

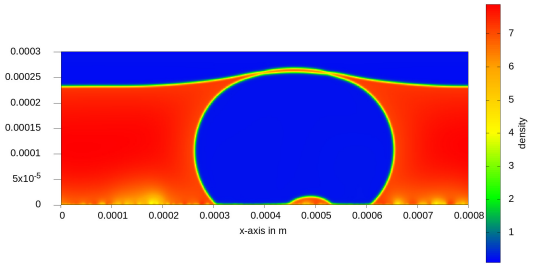



Figure 6.47: The nucleation bubble grows and is about to burst. $\Delta t = 140e3$

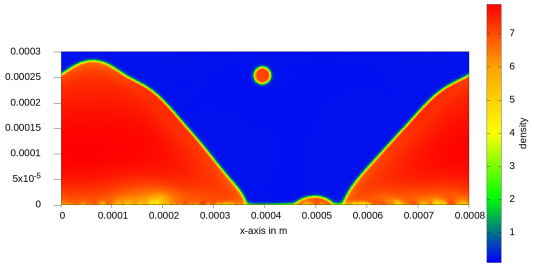



Figure 6.48: The bubble bursts allowing for the liquid to fill the void. The liquid bridge formed a droplet. $\Delta t = 145e3$

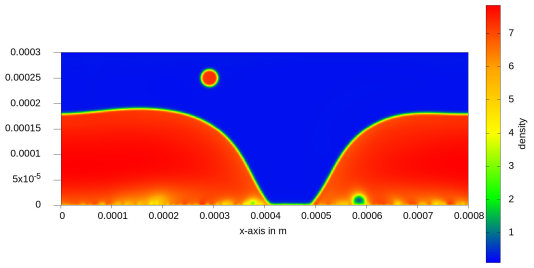



Figure 6.49: The droplet is carried away with the vapour. $\Delta t = 155e3$

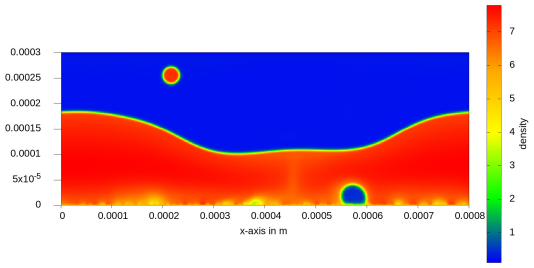



Figure 6.50: The droplet is carried away with the vapour. A new nucleation bubble is already forming. $\Delta t = 165e3$

For a reference of simulation speed, the time taken between step $\delta t = 90e3$ and $\delta t = 170e3$ is 6.8 minutes for a domain containing 135800 nodes. The simulation was run on an system with a 8-core processor (Ryzen 3700X) running at a clock of 4.0GHz.

## 6.3. Multichannel geometry generation

To study the impact of various heating chamber layouts, a parametric geometry generation function was created. The exact code is shown in appendix E.6. This allows for various configurations of straight channel designs to be tested. A few example generation to showcase the capabilities are shown in figs. 6.51 to 6.53.

The simulation geometry is generated from four segments: the inlet, the distribution, the channels and the convergent. For each segment, all parts are parametrically defined, for example the channels, the relevant parameters are: number of channels, channel length, channel width, and channel spacing. The inlet, outlet,

cold and hotwall are automatically differentiated and are assigned a different geometry number, such that the appropriate dynamics can be added.

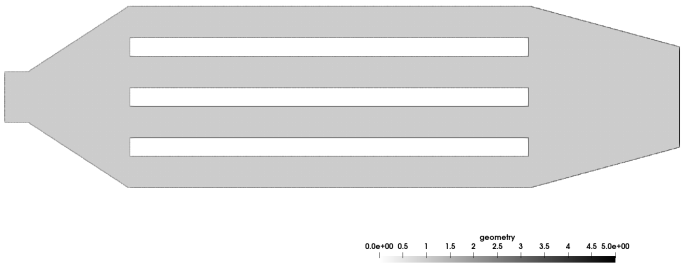



Figure 6.51: Example baseline geometry generated using the parametric multichannel geometry generator

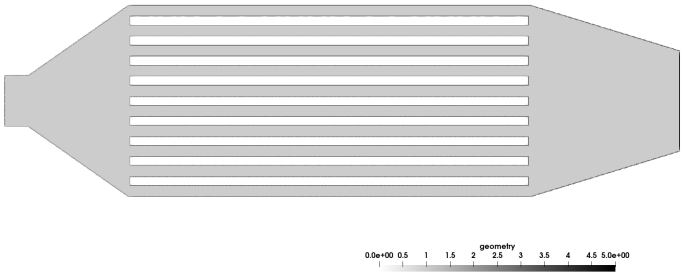



Figure 6.52: Multichannel geometry where the number and size of channels were changed compared to example baseline geometry.

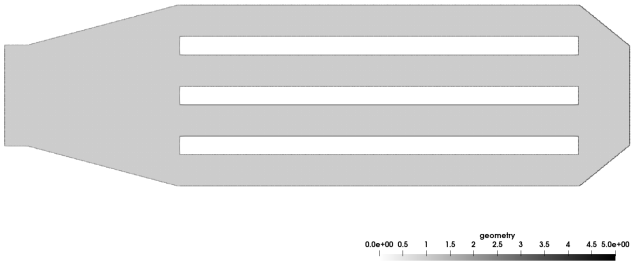



Figure 6.53: Multichannel geometry where the inlet and outlet parameters were changed compared to example baseline geometry.

## 6.4. Validation plan

The plan for the validation was to use two methods, one specific to boiling in multichannels via comparison to literature and a novel way to estimate multiphase flow from thruster performance. This allows for validation of the simulation tool not specific to thrusters and a validation whether it predicts the correct multiphase flow.

### 6.4.1. Comparison to literature

Flow boiling in microchannels has been analysed in literature and a lot of data is available regarding that. However, they do not have a chocked flow, and thus are not exactly representative of the flow of the VLM heating chamber. The literature is however very interesting for validation purposes.

A good experimental test series was done by Kuang et al. for a 4 channel parallel minichannel [65]. Their analysis resulted in a pressure fluctuation relation. The actual experiment data and the provided relation can be used to see if the simulations performed by the tool developed and experiments match up quantitative and qualitative.

In addition, Hetsroni et al. [54, 55], who have done extensive research into microchannel boiling instabilities, also found what the boiling frequency is proportional to. This allows for validation with their given data and a provides and additional check when analysing results at different scales.

### 6.4.2. Multiphase flow type estimation

This approach details a method to estimate the type of multiphase flow present in the thruster from the performance impact due to increased massflow. A few assumptions need to be taken, but this method shows good agreement with experimental data.

The first assumption is that the thruster can vapourize a certain massflow of liquid, $\dot{m}_g$. An increase in massflow beyond that point is an increase in unvaporized liquid, $\dot{m}_l$. The sum of those two is the total massflow, $\dot{m}$, shown in eq. (6.3). The unvaporized massflow fraction, $\phi$, is defined in eq. (6.4).

The second assumption is that the unvaporized flow can be categorized into one of four categories: Isothermal non-acceleration, thermal non-accelerating, isothermal accelerating, thermal accelerating.

Isothermal means that the unvapourized flow keeps the same temperature throughout the nozzle, and thus does not contribute to the expansion of the vapour in the nozzle. The thermal category assumes that the unvaporized flow has the same temperature as the surrounding vapour, thus contribution to the expansion of the vapour.

Non-accelerating means that the unvaporized flow has a much lower speed than the expanding vapour, and also takes no additional energy away that could be used to expand. Accelerating assumes that the unvaporized flow has the same speed as the vapour.

An example of an thermal accelerating flow are fine droplets. While a large annular flow is an example of an isothermal non-accelerating flow.

With these assumptions, it allows for taking the derivative of the specific impulse of the ideal rocket theory with respect to an increased massflow rate. The derivative is different for each category. Comparing the derivatives with the experimental data allows for an estimation of the type of flow. The derivations are shown in appendix B, and the resultant specific impulse derivatives are given in eqs. (6.5) to (6.8), where $c_l$ and $c_g$ are the specific heat capacities for liquid and gas phase respectively. It is assumed that the change in heat capacity ratio is negligible, which only applies to the thermal categories.

$$\dot{m} = \dot{m}_g + \dot{m}_l \tag{6.3}$$

$$\phi = \frac{\dot{m}_l}{\dot{m}_g + \dot{m}_l} \quad \Rightarrow \quad 1 - \phi = \frac{\dot{m}_g}{\dot{m}_g + \dot{m}_l} \tag{6.4}$$

$$\frac{\partial I_{\text{sp,therm,accel}}}{\partial \dot{m}} = -\frac{I_{sp}}{2\dot{m}} \cdot \frac{(c_l - c_g)}{\left(c_g + c_l \frac{\phi}{1-\phi}\right)} \tag{6.5}$$

$$\frac{\partial I_{\text{sp,iso,accel}}}{\partial \dot{m}} = -\frac{I_{sp}}{2\dot{m}} \tag{6.6}$$

$$\frac{\partial I_{\text{sp,therm,no-a}}}{\partial \dot{m}} = -\frac{I_{sp}}{2\dot{m}} \frac{c_l \left(\frac{\phi}{(1-\phi)} + 1\right)}{\left(c_g + c_l \frac{\phi}{1-\phi}\right)} \tag{6.7}$$

$$\frac{\partial I_{\text{sp,iso,no-a}}}{\partial \dot{m}} = -\frac{I_{sp}}{\dot{m}} \tag{6.8}$$

An example analysis is given using the data of Cen and Xu [14], shown in fig. 6.54 in combination with fig. 6.55.

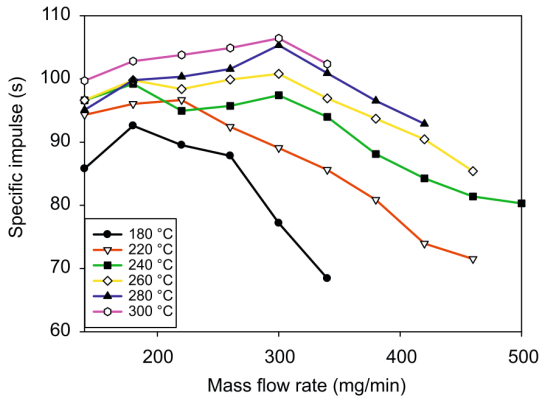



Figure 6.54: Variation of specific impulse with mass flow rate [14]

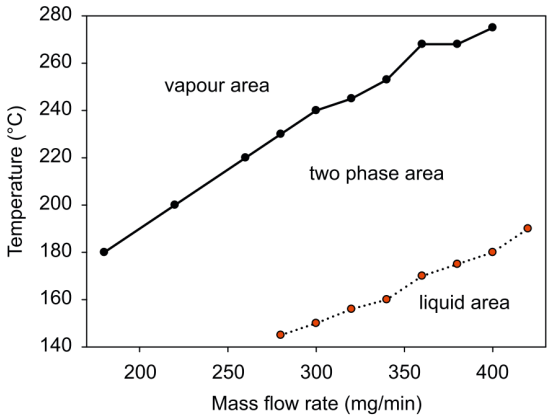



Figure 6.55: Fluid state at the outlet of the nozzle [14]

The data point analyzed is for the curve 180°C at the massflow rate 260 mg/min. The massflow rate step, $\Delta\dot{m}$, is 40 mg/min. For the specific heat, the values used are: $c_l = 4200$ and $c_g = 1900$. Using fig. 6.55, the value for vaporized massflow $\dot{m}_g = 180$ mg/min is determined. The web plot digitizer tool https://automeris.io/WebPlotDigitizer/ is used to determine the exact value for the two datapoints. At 260 mg/min, the specific impulse is 88.0, and at 300 mg/min the specific impulse is 77.3 s.

If the multiphase flow is isothermal and accelerating, the expected specific impulse would be 81.2 s.

$$\begin{aligned}
\frac{\partial I_{\text{sp,iso,accel}}}{\partial \dot{m}} &= -\frac{I_{sp}}{2\dot{m}} \\
&= -\frac{88}{2 \cdot 260} = -0.169 \text{s}^2\text{kg}^{-1} \\
I_{\text{sp,iso,accel,1}} &= I_{\text{sp,iso,accel,0}} + \frac{\partial I_{\text{sp,iso,accel}}}{\partial \dot{m}} \Delta\dot{m} \\
&= 88 - 0.169 \cdot 40 = 81.2 \text{s}
\end{aligned}$$

If the multiphase flow is thermal and not accelerating, the expected specific impulse would be 77.1 s.

$$\frac{\partial I_{\text{sp,therm,no-a}}}{\partial \dot{m}} = -\frac{I_{sp}}{2\dot{m}} \frac{c_l \left(\frac{\phi}{(1-\phi)} + 1\right)}{\left(c_g + c_l \frac{\phi}{1-\phi}\right)}$$

$$\begin{aligned}
\frac{\phi}{1-\phi} &= \frac{\dot{m}}{\dot{m}_g} - 1 = \frac{260}{180} - 1 = 0.444 \\
&= -\frac{88}{2 \cdot 260} \frac{4200 \cdot (0.444 + 1)}{(1900 + 4200 \cdot 0.444)} \\
&= -\frac{88}{2 \cdot 260} \frac{6066.7}{3766.7} = -\frac{88}{2 \cdot 260} \cdot 1.61 = 0.272 \text{s}^2\text{kg}^{-1} \\
I_{\text{sp,therm,no-a,1}} &= I_{\text{sp,therm,no-a,0}} + \frac{\partial I_{\text{sp,therm,no-a}}}{\partial \dot{m}} \Delta \dot{m} \\
&= 88 - 0.272 \cdot 40 = 77.1 \text{s}
\end{aligned}$$

If the multiphase flow is isothermal and not accelerating, the expected specific impulse would be 74.5 s.

$$\begin{aligned}
\frac{\partial I_{\text{sp,iso,no-a}}}{\partial \dot{m}} &= -\frac{I_{sp}}{\dot{m}} \\
&= -\frac{88}{260} = -0.338 \text{s}^2\text{kg}^{-1} \\
I_{\text{sp,iso,no-a,1}} &= I_{\text{sp,iso,no-a,0}} + \frac{\partial I_{\text{sp,iso,no-a}}}{\partial \dot{m}} \Delta \dot{m} \\
&= 88 - 0.338 \cdot 40 = 74.5 \text{s}
\end{aligned}$$

From the above three relations, the closest one is the thermal, non-accelerating flow. The exact multiphase flow type cannot be determined, but it does provides more insight into the multiphase flow occurring. A further analytical study may be required to extract more information regarding the multiphase flow that is occurring.

# 7 Conclusions and Recommendations

In the introduction, the need for a simulation tool capable of simulating complex fluid flows for micro thrusters was given. The following chapters introduces and extends an existing simulation base to be capable of handling the needs identified. In this chapter the conclusions with a reflection to the needs identified will be given. Then any recommendations are presented based on any need that remains unmet.

## 7.1. Conclusions

The thesis research objective that was introduced in the introduction is:

> *to provide quantitative data regarding the unvaporized flow characteristics exiting the heating chamber of the VLM and investigate how design parameters influence the unvaporized flow aiding further research into performance impact assessment and development mitigation methods.*

The research objective was then subdivided into research questions items RQ-1 to RQ-3 with success criteria. The success rate of each research question is discussed below.

The planned final simulations for single and multi channel flow did not occur due to the simulation's fluid metastability region being too large, thus preventing nucleation, see section 6.2.3. Therefore, the overarching success rate for the research question is insufficient. However, since enough information is available to solve the metastability problem, a secondary opinion is given with the assumption that the nucleation process is functioning.

### 7.1.1. Conclusions research question 1: Influence of design parameters

In item RQ-1, the question is about the influence of design parameters on the un-vaporized flow characteristics and flow instabilities for straight channel heating chambers. Overall, this was not answered due to incomplete simulations, however the simulation tool has shown to be able to answer this question almost fully.

The requirements of item RQ-1 was to vary the design parameters by ±10% and ±20%, where the parameters to be varied in question are: channel length, size and amount of channels for the heating chamber and surface tension, wettability, viscosity, specific heat capacity for the fluid.

For the operational point, the parameters to be varied are the inflow massflow rate, wall temperature and impact of gravity. The setting of the massflow in the simulation is achieved by setting both the density and velocity at the inlet, as was demonstrated. Additionally, mitigation techniques are given to deal with the wall interaction impacts. The wall temperature is possible to be set to a specific temperature and gravity can be added to the simulations.

It was shown that all the parameters of the geometry are possible to be varied. The geometry in question was simple to generate programmatically, allowing a large flexibility for choosing the parameters freely. It is also possible to import externally generated geometries, thus allowing for the implementation of more complex geometries.

For the fluid, the parameters that have been shown to be variable are: wettability, viscosity, specific heat, thermal conductivity. The equation of state allows for varying of the fluid used, and with it the latent heat and densities, which were not a requirement. A flaw due to the implementation of the multiphase method is the dependence of the surface tension on the pseudopotential. Thus, while it is possible to vary the surface

tension, it is not as straight forward as with other parameters, such as the specific heat.

Overall, it was shown that the tool is capable of varying the design parameters mentioned easily, with the exception of surface tension.

### 7.1.2. Conclusions research question 2: Exiting flow characteristics

This research question is about the simulated flow exiting the heating chamber and the requirements of the simulation tool.

The *first* requirement for the simulation is to be applicable to the VLM. This requires the simulation to encompass thermal, phase change and multiphase flow, all of which was achieved. However, due to the metastability issue, incorrect boiling behaviour is recovered. The LBM is suitable to simulating the VLM, for this and future iterations which may include more complex geometry.

The *second* requirement for the flow to be 3D can be achieved as well. The 3D simulations are however, computationally much more expensive than 2D, and while some verification cases were performed for both 3D and 2D, full 3D simulations were not performed.

The *third* requirement for the flow to encompass compressible fluid flow is achieved as well, since the LBM is inherently weakly compressible. The pseudopotential method allows for pressure waves in the vapour as well as the liquid. However, the pressure also impacts the density of the fluid, which is aphysical. This was seen by the Laplace pressure influencing the density inside a droplet, see section 4.2.3. The benefit of this is that the pressure oscillations can be recorded for the *seventh* requirement.

The *fourth* regarding the minimum spatial resolution can be achieved. However, it does impact time step and increase simulation time due to viscosity linkage. At the current interface width, this requirement results in an unreasonable fine grid. Using a thinner interface and mitigate the spurious currents differently would allow for this requirement to be met.

The *fifth* and *sixth* requirement regarding the simulation duration is purely a function of time available and processing power. Since the implementation of the simulation model to develop the tool took the majority of thesis time, not enough time was available to fulfil this requirement. However, the 2D multichannel simulations time requirement was met with overnight simulations.

The *eight* and *ninth* requirement for the simulation walls and save interval were fulfilled. The bounce back method gives a no-slip wall and a constant temperature wall. The simulation can be saved as a full domain save, specific data such as pressure at the outlet or maximum density, or directly as a picture. This gives many ways to save the simulation state at high frequency with negligible impact on simulation speed.

The simulation did not predict any multiphase flow pattern, in the microchannels, and therefore requirement *ten* and *eleven* are a fail. However, the problems were identified and solutions to fix them have been presented.

### 7.1.3. Conclusions research question 3: Simulation accuracy

The final research question is about the simulation accuracy. Due to the lack of nucleation, there is a lack of multiphase flow patterns, leaving the first requirement unanswered. However, the simulation did follow the Young-Laplace law and the $D^2$-law, two very important benchmarks for multiphase systems. Especially the $D^2$-law indicates that evaporation is possible, successfully answering the third requirement.

The focus of this thesis was on correctly implementing the model, which was successfully done. However, since there was no nucleation, investigating that was prioritised after the implementation. Therefore, simulations to assess the accuracy of droplet dynamics and film breakage, to answer the second and fourth requirement, were not performed. While there was no explicit film breakage simulated, film breakage inside the channels was simulated.

Once the nucleation problem is successfully addressed, the unanswered research question and requirements can all be addressed. For that, a validation plan is presented as well, focusing on microchannel boiling and instabilities. A novel method to estimate the multiphase flow occurring from only the performance impact is shown.

## 7.2. Recommendations

Over the course of the thesis, several points of improvement were identified. These are collected below, with what the point of improvement is, what the hoped improvements are, and if applicable how to approach it. The recommendations are ordered in what is considered the best improvements for the required work.

### 7.2.1. Implement MRT collision operator

The BGK collision operator was used in this thesis, and while it is a good first approach, it has its limitations. One of the limitations are the spurious currents at lower temperature ratios due to the high density gradients. To deal with those currents, a thicker interface width was used, which prevent nucleation from happening. A MRT collision operator will drastically reduce the magnitude of the spurious currents and thus a lower interface width can be used. This means that nucleation is more likely to happen.

OpenLB has a framework for using the MRT collision operator in the simulations. The only addition would be the forcing scheme that was modified to increase stability in combination with the pseudopotential method. Li et al. [78] have published a paper for the D2Q9 lattice and Zhang et al. [153] for the D3Q19.

Implementation of the MRT collision operator was attempted in this thesis, and some findings regarding that can be found in section 3.3. A difficulty with the MRT collision operator is that there are turning parameters that affect the simulations, but do not have a physical property counterpart. As such, a good fundamental knowledge regarding MRT and the parameters are needed, which requires experience. This should not be underestimated and implementing running simulations can be a thesis in of itself.

### 7.2.2. Investigate molecular dynamics nucleation

This recommendation is regarding the nucleation problem, which was explored near the end of the thesis. Techniques specific to LBM, that are already known, are recommended separately. It was mentioned by Li et al. that discussions regarding metastability of vapour-liquid transition "can be found in some studies of non-ideal equations of state and molecular dynamics simulations of boiling." [84] They also include three references to non-LBM literature.

Since in LBM there is a distinct lack of reference to the metastabilty, it would be interesting to know what techniques are used in the non-ideal equation of state and molecular dynamics simulation fields to help initiate phase change, and whether these techniques can also be used in LBM.

### 7.2.3. Boundary conditions

Throughout the thesis various boundary methods have been used. Several different methods for LBM have been proposed in literature which behave differently microscopically but the same macroscopically. Thus it may be that a different boundary method implementation is more suitable for certain scenarios. The walls, inlet and outlet are discussed separately.

#### Wall

In the thesis, the no-slip bounce back method was used for the solid walls. This is a well known and implemented method which is applicable for a wide range of boundaries. However, it is not the most accurate.

The main advantage for choosing a different implementation is accuracy, since simple BB can suffer from

numerical slip flow, or low accuracy. Another advantage is possibly better nucleation, since OpenLB uses a fixed density at the wall and thus a density gradient forms. Other implementations may results in removing this gradient, which would help in the nucleation. The disadvantage for using a different boundary is that the wall contact angle implementation is no longer supported in OpenLB, and needs to be included via the coupling.

Most papers referenced used the Zou-He boundary method. This method is a non-equilibrium bounce back method, with very high accuracy for 2D straight walls, and is not applicable for curved walls. This method is already implemented in OpenLB, but does not yet support setting of wall wettability. The downside is that it suffers from slight stability issues in 2D and has major stability issues in 3D.

To incorporate curved walls, a whole separate boundary implementation may be required. It is possible certainly possible to have curved boundaries with the pseudopotential, as was shown in [26], but they did not mention what boundary method they used. It is very likely that curve specific boundary conditions, such as the Bouzidi scheme, in combination with wall wettability force scheme is good enough. However, the bounce back method can be used, but introduces the staircase effect.

#### Inlet

Only few problems have been identified with the inlet. The main problem is the density gradient forming a compressing of the fluid, which has an interactive effect on the thermal lattice. This can be dealt with in various ways.

First is just accepting the compression of the liquid and re-setting the thermal lattice to remove the secondary effects. This results in acceptable inlet conditions as long as the simulation of interest is far enough downstream.

Another possible way is to set the actual inlet boundary slightly smaller than the domain inlet, theoretically letting the fluid expand to counteract the compression of the wall. This however is just a mitigation and changes depending on the density gradient at the wall. Since the density gradient is determined by the equation of state attractive parameter and the wall wettability, this mitigation method is not recommended.

The last method to deal with the inlet would be to incorporate the density gradient into the inlet. This however would require a definition of the density gradient, or some adaptive inlet conditions.

#### Outlet

The outlet is a more difficult but also more important boundary to fix. Conventional boundary implementations either cannot handle multiphase flow (constant pressure boundary), or distort the multiphase flow (velocity boundary), or leave too little control over the outlet (convective boundary).

A few papers ignore this outlet issue by simulating / using results only to the point where multiphase flow approaches the outlet or assuming an outlet state [13, 44]. This allows the simulation of multiphase flow with single phase open boundaries. Multiphase outlets are not well researched, where only few papers compare various outlet conditions for high density ratio flows [76, 90].

### 7.2.4. Different equation of states

In this thesis, the Peng-Robinson equation of state was used. This is one of the most popular found in literature for simulating water. However, this is because Yuan and Schaefer [150] found that in a comparison between the Carnahan-Starling, Redlich-Kwong, Redlich-Kwong Soave, the Peng-Robinson equation of state best describes water. However, there are equations of states that build up on the Peng-Robinson: the Peng–Robinson-Stryjek-Vera and Peng,-Robinson-Babalola. It would be interesting to see how these equation of states behave in the pseudopotential lattice Boltzmann method, and whether they perform better or worse for nucleation.

### 7.2.5. Hybrid thermal LBM

In this thesis, the thermal problem is solved using the double distribution function on a separate lattice. To decouple the diffusion, the thermal problem is solved via a the Euler method and then reintroduced into LBM. This carries several error sources with it, and a directer, faster, and more accurate approach is to not use LBM for the thermal problem.

This was done by Li et al. [79], where they solved the thermal problem using a fourth order Runge-Kutta solver. Incorporating a hybrid solver into OpenLB would be something new, where there are no indications found in the source code for this.

Probably the best approach would be to store and modify the temperature via a descriptor field of the fluid lattice. The Runge-Kutta calculations can then be performed via a coupling step. The main issue would be implementing and handling the boundaries. That should be possible in combination with the geometry.

Alternatively, a different thermal implementation can be used. By changing the equilibrium density function and the lattice weights, it is possible to modify the thermal diffusivity without changing the relaxation time. This approach was not performed in the series of papers investigated, but is found in a paper by Chen et al. [18].

### 7.2.6. Porous medium

The LBM is known to be excellent in dealing with complex geometry and flows through porous media. It has been used to simulate viscous fingering, relative permeability and effective diffusivity. Pseudopotential LBM has been used to simulate fuel cells, batteries, and flow inside carbon-fibres. The review of the pseudopotential method [20] is a good starting point regarding multiphase porous flows.

Since the TUDelft is also investigating porous microthrusters [144], this would allow for this simulation tool to be used to help design both the channel based and foam based VLM.

### 7.2.7. Aphysical mass transfer

A flaw of the pseudopotential method is that it shows aphysical mass transfer, where smaller droplets get absorbed by larger droplets from afar. This aphysical mass transfer is due to the pure attraction term and spurious currents of the pseudopotential method [46]. A proper fix would require a repulsive term, therefore a multispeed lattice, which is detailed in *coalescence avoidance*. However, if it is possible to reduce this effect without using a multispeed lattice, that would be very beneficial.

### 7.2.8. Implement entropic collision operator

The entropic collision operator has shown to be remarkably stable, and shows potential for multiple complex physical phenomena. This may lead to better stability and thus thinner interfaces, promoting nucleation. However, very little research was done so far on entropic pseudopotential method, but a start is given by [95, 100]. However, since only the dynamics changes, this should be relatively self contained.

A far away goal related to the entropic collision operator is that it allows for supersonic flow to be simulated with a multispeed lattice. This may allow for the possibility to extend the simulation domain up to the throat, greatly increasing the nozzle-chamber interaction modelling accuracy. With very large multispeed lattices, it may be possible to simulate the complete thruster: the heating chamber and nozzle, including slip flow.

### 7.2.9. Coalescence avoidance

A flaw of the pseudopotential method is that it only incorporates an attractive term. This results in that any system of multiple droplets will always coalesce into one droplet given enough time and no other disturbing forces. This is unwanted since tracking of small droplets was one of the goals of this thesis. However, the

disturbance caused by the flow of vapour should help keeping the droplets separated.

Two factors that drive the coalescence of droplets are the spurious currents and thick interfaces. If the interface is thick, then the density gradient can reach multiple lattice nodes, if not tens. This can form a density gradient bridge which then pulls together two droplets. The other factor is the spurious currents, which can carry density from a smaller droplet to a larger droplet. This results in mass transfer over a distance, where droplets get absorbed without collisions.

There is a known extension to the pseudopotential method which can prevent this coalescence. This extension introduces a repulsive force term which keeps the interface forces in balance. However, this repulsive term requires the usage of a multispeed lattice for the force calculations. Since it is only used for the force calculation, it should have minimal computational impact on the collision and streaming step, and only impacting the coupling step. An additional benefit of including the next nearest neighbours is that the spurious currents can be decreased by an order of magnitude. [25]

While there are indications that OpenLB plans to support multispeed lattices, they do not provide any. It may be possible to implement, since only the force calculation is affected. However, a double layer for the boundary will be required, and standard inlet/outlet boundary conditions will most likely not work. Introducing a multispeed lattice is probably beyond the capability of a student to do in a master thesis.

# Bibliography


[1] D. L. Albernaz, M. Do-Quang, and G. Amberg. Lattice Boltzmann Method for the Evaporation of a Suspended Droplet. *Interfacial Phenomena and Heat Transfer*, 1(3):245–258, 2013. ISSN 2169-2785. doi: 10.1615/interfacphenomheattransfer.2013010175.

[2] Y. Ba, H. Liu, J. Sun, and R. Zheng. Color-gradient lattice Boltzmann model for simulating droplet motion with contact-angle hysteresis. *Physical Review E*, 88(4):043306, oct 2013. ISSN 1539-3755. doi: 10.1103/PhysRevE.88.043306.

[3] C. Baldassari and M. Marengo. Flow boiling in microchannels and microgravity. *Progress in Energy and Combustion Science*, 39(1):1–36, feb 2013. ISSN 03601285. doi: 10.1016/j.pecs.2012.10.001.

[4] Y. Bao and J. Meskas. Lattice Boltzmann Method for Fluid Simulations. Technical report, New York University, apr 2011. URL https://cims.nyu.edu/{~}billbao/report930.pdf.

[5] A. Bejan. Multiphase Systems. In *Advanced Engineering Thermodynamics*, chapter 6, pages 213–270. John Wiley & Sons, Durham, North Carolina, fourth edition, 2016. ISBN 978-1-119-05209-8. URL http://www.ams.org/mmono/137.

[6] R. Benzi, S. Succi, and M. Vergassola. The lattice Boltzmann equation: theory and applications. *Physics Reports*, 222(3):145–197, dec 1992. ISSN 03701573. doi: 10.1016/0370-1573(92)90090-M.

[7] R. Benzi, L. Biferale, M. Sbragaglia, S. Succi, and F. Toschi. Mesoscopic modeling of a two-phase flow in the presence of boundaries: The contact angle. *Physical Review E*, 74(2):021509, aug 2006. ISSN 1539-3755. doi: 10.1103/PhysRevE.74.021509.

[8] J. Bernsdorf. *Simulation of complex flows and multi-physics with the Lattice-Boltzmann method*. PhD thesis, University of Amsterdam, 2008. URL http://dare.uva.nl/document/93122.

[9] S. Biringen. A note on the numerical stability of the convection-diffusion equation. *Journal of Computational and Applied Mathematics*, 7(1):17–20, mar 1981. ISSN 03770427. doi: 10.1016/0771-050X(81)90002-4.

[10] B. M. Boghosian, J. Yepez, P. V. Coveney, and A. Wager. Entropic lattice Boltzmann methods. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 457(2007): 717–766, mar 2001. ISSN 1471-2946. doi: 10.1098/rspa.2000.0689.

[11] F. Bösch. *Entropic Lattice Boltzmann Models for Fluid Dynamics*. Doctoral thesis, Swiss Federal Institute of Technology in Zurich, 2017.

[12] G. Brenner, T. Zeiser, K. Beronov, P. Lammers, and J. Bernsdorf. Lattice Boltzmann Methods: High Performance Computing and Engineering Applications. In *Parallel Computational Fluid Dynamics 2002*, pages 3–12. Elsevier, 2003. doi: 10.1016/B978-044450680-1/50002-4. URL https://linkinghub.elsevier.com/retrieve/pii/B9780444506801500024.

[13] J. Cai, X. Huai, B. Liu, and Z. Cui. Numerical prediction of thin liquid film near the solid wall for hydraulic cavitating flow in microchannel by a multiphase lattice Boltzmann model. *International Journal of Heat and Mass Transfer*, 127:107–115, 2018. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2018.06.146.

[14] J. Cen and J. Xu. Performance evaluation and flow visualization of a MEMS based vaporizing liquid micro-thruster. *Acta Astronautica*, 67(3-4):468–482, aug 2010. ISSN 00945765. doi: 10.1016/j.actaastro.2010.04.009.

[15] A. Cervone, B. Zandbergen, D. C. Guerrieri, M. De Athayde Costa e Silva, I. Krusharev, and H. van Zeijl. Green micro-resistojet research at Delft University of Technology: new options for Cubesat propulsion. *CEAS Space Journal*, 9(1):111–125, mar 2017. ISSN 1868-2502. doi: 10.1007/s12567-016-0135-3.

[16] Chapman and Cowling. Chapman_Cowling_1953_the_Mathematical_Theory_of_Non-Uniform_-Gases, 1970.

[17] C.-C. Chen, C.-W. Liu, H.-C. Kan, L.-H. Hu, G.-S. Chang, M.-C. Cheng, and B.-T. Dai. Simulation and experiment research on vaporizing liquid micro-thruster. *Sensors and Actuators A: Physical*, 157(1): 140–149, jan 2010. ISSN 09244247. doi: 10.1016/j.sna.2009.10.025.

[18] L. Chen, Q. Kang, Y.-L. He, and W.-Q. Tao. Pore-scale simulation of coupled multiple physicochemical thermal processes in micro reactor for hydrogen production using lattice Boltzmann method. *International Journal of Hydrogen Energy*, 37(19):13943–13957, oct 2012. ISSN 03603199. doi: 10.1016/j.ijhydene.2012.07.050.

[19] L. Chen, Q. Kang, B. A. Robinson, Y.-L. He, and W.-Q. Tao. Pore-scale modeling of multiphase reactive transport with phase transitions and dissolution-precipitation processes in closed systems. *Physical Review E*, 87(4):043306, apr 2013. ISSN 1539-3755. doi: 10.1103/PhysRevE.87.043306.

[20] L. Chen, Q. Kang, Y. Mu, Y.-L. He, and W.-Q. Tao. A critical review of the pseudopotential multiphase lattice Boltzmann model: Methods and applications. *International Journal of Heat and Mass Transfer*, 76:210–236, sep 2014. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2014.04.032.

[21] M. CHENG and K. C. HUNG. LATTICE BOLTZMANN METHOD ON NONUNIFORM MESH. *International Journal of Computational Engineering Science*, 05(02):291–302, jun 2004. ISSN 1465-8763. doi: 10.1142/S1465876304002381.

[22] S. Chibbaro, G. Falcucci, G. Chiatti, H. Chen, X. Shan, and S. Succi. Lattice Boltzmann models for nonideal fluids with arrested phase-separation. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 77(3):1–9, 2008. ISSN 15393755. doi: 10.1103/PhysRevE.77.036705.

[23] S. S. Chikatamarla and I. V. Karlin. Lattices for the lattice Boltzmann method. *Physical Review E*, 79(4): 046701, apr 2009. ISSN 1539-3755. doi: 10.1103/PhysRevE.79.046701.

[24] C. Choi, J. S. Shin, D. I. Yu, and M. H. Kim. Flow boiling behaviors in hydrophilic and hydrophobic microchannels. *Experimental Thermal and Fluid Science*, 35(5):816–824, 2011. ISSN 08941777. doi: 10.1016/j.expthermflusci.2010.07.003.

[25] K. Connington and T. Lee. A review of spurious currents in the lattice Boltzmann method for multiphase flows. *Journal of Mechanical Science and Technology*, 26(12):3857–3863, dec 2012. ISSN 1738-494X. doi: 10.1007/s12206-012-1011-5.

[26] H. N. Dalgamoni and X. Yong. Axisymmetric lattice Boltzmann simulation of droplet impact on solid surfaces. *Physical Review E*, 98(1):6–8, 2018. ISSN 24700053. doi: 10.1103/PhysRevE.98.013102.

[27] Delft University of Technology. Miniaturization: Micro-Propulsion, 2016. URL https://www.tudelft.nl/en/ae/organisation/departments/space-engineering/space-systems-engineering/research/miniaturization/micro-propulsion/.

[28] Delft University of Technology. Delfi program, 2019. URL https://www.tudelft.nl/lr/delfi-space/delfi-program/.

[29] Y. Deng, F.-S. Lien, and E. Yee. The lattice Boltzmann method for compressible flows at high Mach number. *23rd Annual Conference of the Computational Fluid Dynamics Society of Canada*, 2015.

[30] D. D'Humières and I. Ginzburg. Viscosity independent numerical errors for Lattice Boltzmann models: From recurrence equations to "magic" collision numbers. *Computers and Mathematics with Applications*, 58(5):823–840, 2009. ISSN 08981221. doi: 10.1016/j.camwa.2009.02.008.

[31] S. Dou and L. Hao. Numerical study of droplet evaporation on heated flat and micro-pillared hydrophobic surfaces by using the lattice Boltzmann method. *Chemical Engineering Science*, 229:116032, 2021. ISSN 00092509. doi: 10.1016/j.ces.2020.116032.

[32] A. Fakhari, D. Bolster, and L. S. Luo. A weighted multiple-relaxation-time lattice Boltzmann method for multiphase flows and its application to partial coalescence cascades. *Journal of Computational Physics*, 341:22–43, 2017. ISSN 10902716. doi: 10.1016/j.jcp.2017.03.062.

[33] G. Falcucci, G. Bella, G. Chiatti, S. Chibbaro, M. Sbragaglia, and S. Succi. Lattice Boltzmann models with mid-range interactions. *Communications in Computational Physics*, 2(6):1071–1084, 2007. ISSN 18152406.

[34] N. Frapolli, S. S. Chikatamarla, and I. V. Karlin. Entropic lattice Boltzmann model for compressible flows. *Physical Review E*, 92(6):061301, dec 2015. ISSN 1539-3755. doi: 10.1103/PhysRevE.92.061301.

[35] N. Frapolli, S. S. Chikatamarla, and I. V. Karlin. Entropic lattice Boltzmann model for gas dynamics: Theory, boundary conditions, and implementation. *Physical Review E*, 93(6):063302, jun 2016. ISSN 2470-0045. doi: 10.1103/PhysRevE.93.063302.

[36] N. Frapolli, S. Chikatamarla, and I. Karlin. Theory, Analysis, and Applications of the Entropic Lattice Boltzmann Model for Compressible Flows. 22(3):370, mar 2020. ISSN 1099-4300. doi: 10.3390/e22030370.

[37] C. From. *High-Order Lattice Boltzmann for nonideal fluid mixtures*. Doctoral thesis, Queensland University of Technology, 2020. URL https://eprints.qut.edu.au/200190/.

[38] Geneva University: Scientific and Parallel Computing Group. Palabos Github. URL https://gitlab.com/unigespc/palabos.

[39] I. Ginzburg, F. Verhaeghe, and D. D'Humières. Two-relaxation-time Lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions. *Communications in Computational Physics*, 3(2):427–478, 2008. ISSN 18152406.

[40] I. Ginzburg, F. Verhaeghe, and D. D'Humières. Study of simple hydrodynamic solutions with the two-relaxation-times lattice Boltzmann scheme. *Communications in Computational Physics*, 3(3):519–581, 2008. ISSN 18152406.

[41] S. Gong and P. Cheng. A lattice Boltzmann method for simulation of liquid–vapor phase-change heat transfer. *International Journal of Heat and Mass Transfer*, 55(17-18):4923–4927, aug 2012. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2012.04.037.

[42] S. Gong and P. Cheng. Numerical investigation of droplet motion and coalescence by an improved lattice Boltzmann model for phase transitions and multiphase flows. *Computers & Fluids*, 53(1):93–104, jan 2012. ISSN 00457930. doi: 10.1016/j.compfluid.2011.09.013.

[43] S. Gong and P. Cheng. Lattice Boltzmann simulation of periodic bubble nucleation, growth and departure from a heated surface in pool boiling. *International Journal of Heat and Mass Transfer*, 64:122–132, sep 2013. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2013.03.058.

[44] S. Gong and P. Cheng. Numerical investigation of saturated flow boiling in microchannels by the lattice boltzmann method. *Numerical Heat Transfer; Part A: Applications*, 65(7):644–661, 2014. ISSN 15210634. doi: 10.1080/10407782.2013.836025.

[45] W. Gong, Y. Y. Yan, S. Chen, and E. Wright. A modified phase change pseudopotential lattice Boltzmann model. *International Journal of Heat and Mass Transfer*, 125:323–329, 2018. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2018.04.090.

[46] W. Gong, Y. Yan, and S. Chen. A study on the unphysical mass transfer of SCMP pseudopotential LBM. *International Journal of Heat and Mass Transfer*, 123:815–820, 2018. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2018.03.032.

[47] G. Gonnella, A. Lamura, and V. Sofonea. Lattice Boltzmann simulation of thermal nonideal fluids. *Physical Review E*, 76(3):036703, sep 2007. ISSN 1539-3755. doi: 10.1103/PhysRevE.76.036703.

[48] B. Greenfield, W. F. Louisos, and D. L. Hitt. Impact of Dilute Multiphase Flow in Supersonic Micronozzles. *Journal of Spacecraft and Rockets*, 56(1):190–199, jan 2019. ISSN 0022-4650. doi: 10.2514/1.A34215.

[49] A. K. Gunstensen, D. H. Rothman, S. Zaleski, and G. Zanetti. Lattice Boltzmann model of immiscible fluids. *Physical Review A*, 43(8):4320–4327, 1991. ISSN 10502947. doi: 10.1103/PhysRevA.43.4320.

[50] Z. Guo, B. S. Haynes, and D. F. Fletcher. Numerical simulation of annular flow boiling in microchannels. *Journal of Computational Multiphase Flows*, 8(1):61–82, 2016. ISSN 17574838. doi: 10.1177/1757482X16634205.

[51] Z. Guo, C. Zheng, and B. Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Physical Review E*, 65(4):046308, apr 2002. ISSN 1063-651X. doi: 10.1103/PhysRevE.65.046308.

[52] C. A. J. Hanselaar. Evaporative Two-Phase Micro-Flow Modelling, Master Thesis, Delft University of Technology, 2016. URL http://resolver.tudelft.nl/uuid:030573d7-42c7-4b3e-802a-5b1b9f98ed38.

[53] G. Hazi and A. Markus. On the bubble departure diameter and release frequency based on numerical simulation results. *International Journal of Heat and Mass Transfer*, 52(5-6):1472–1480, 2009. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2008.09.003.

[54] G. Hetsroni. Boiling in micro-channels. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 58(1):155–163, mar 2010. ISSN 0239-7528. doi: 10.2478/v10175-010-0016-4.

[55] G. Hetsroni, A. Mosyak, E. Pogrebnyak, and Z. Segal. Periodic boiling in parallel micro-channels at low vapor quality. *International Journal of Multiphase Flow*, 32(10-11):1141–1159, 2006. ISSN 03019322. doi: 10.1016/j.ijmultiphaseflow.2006.06.005.

[56] A. Hu, L. Li, and R. Uddin. Force method in a pseudo-potential lattice Boltzmann model. *Journal of Computational Physics*, 294:78–89, aug 2015. ISSN 00219991. doi: 10.1016/j.jcp.2015.03.009.

[57] M. R. Kamali, J. J. Gillissen, H. E. Van Den Akker, and S. Sundaresan. Lattice-Boltzmann-based two-phase thermal model for simulating phase change. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 88(3):1–8, 2013. ISSN 15393755. doi: 10.1103/PhysRevE.88.033302.

[58] T. Karayiannis and M. Mahmoud. Flow boiling in microchannels: Fundamentals and applications. *Applied Thermal Engineering*, 115:1372–1397, mar 2017. doi: 10.1016/j.applthermaleng.2016.08.063.

[59] L. S. Kim, H. K. Jeong, M. Y. Ha, and K. C. Kim. Numerical simulation of droplet formation in a microchannel using the lattice Boltzmann method. *Journal of Mechanical Science and Technology*, 22(4):770–779, 2008. ISSN 1738494X. doi: 10.1007/s12206-007-1201-8.

[60] A. Kosar, C.-J. Kuo, and Y. Peles. Suppression of Boiling Flow Oscillations in Parallel Microchannels by Inlet Restrictors. *Journal of Heat Transfer*, 128(3):251, 2006. ISSN 00221481. doi: 10.1115/1.2150837.

[61] M. J. Krause, S. Avis, D. Dapalo, N. Hafen, M. Haußmann, M. Gaedtke, F. Klemens, A. Kummerländer, M.-L. Maier, A. Mink, J. Ross-Jones, S. Simonis, and R. Trunk. OpenLB Release 1.3: Open Source Lattice Boltzmann Code, Zenodo, 2019. URL https://doi.org/10.5281/zenodo.3625967.

[62] M. J. Krause, S. Avis, H. Kusumaatmaja, D. Dapalo, M. Gaedtke, N. Hafen, M. Haußmann, J. Jeppener-Haltenhoff, L. Kronberg, A. Kummerländer, J. E. Marquardt, T. Pertzel, S. Simonis, R. Trunk, M. Wu, and A. Zarth. OpenLB Release 1.4: Open Source Lattice Boltzmann Code, Zenodo, nov 2020. URL https://doi.org/10.5281/zenodo.4279263.

[63] B. Kravets, T. Rosemann, S. R. Reinecke, and H. Kruggel-Emden. A new drag force and heat transfer correlation derived from direct numerical LBM-simulations of flown through particle packings. *Powder Technology*, 345:438–456, 2019. ISSN 1873328X. doi: 10.1016/j.powtec.2019.01.028.

[64] T. Kruger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen. *The Lattice Boltzmann Method*. Springer International Publishing, 2017. ISBN 978-3-319-44647-9. doi: 10.1007/978-3-319-44649-3.

[65] Y. Kuang, W. Wang, J. Miao, X. Yu, and R. Zhuan. Theoretical analysis and modeling of flow instability in a mini-channel evaporator. *International Journal of Heat and Mass Transfer*, 104:149–162, 2017. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2016.08.042.

[66] E. Kulu. Nanosats Database - Figures, 2019. URL https://www.nanosats.eu/{#}figures.

[67] E. Kulu. What is a CubeSat & other picosatellites, 2019. URL https://www.nanosats.eu/cubesat.

[68] A. L. Kupershtokh. Criterion of numerical instability of liquid state in LBE simulations. *Computers and Mathematics with Applications*, 59(7):2236–2245, 2010. ISSN 08981221. doi: 10.1016/j.camwa.2009.08.058.

[69] A. Kurmanbay. Design, Fabrication and Characterization of MEMS based micro heater for Vaporizing Liquid Microthruster, Master Thesis, Delft University of Technology, 2019.

[70] P. Lallemand and L.-S. Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. *Physical Review E*, 61(6):6546–6562, jun 2000. ISSN 1063-651X. doi: 10.1103/PhysRevE.61.6546.

[71] J. Latt, O. Malaspinas, D. Kontaxakis, A. Parmigiani, D. Lagrava, F. Brogi, M. B. Belgacem, Y. Thorimbert, S. Leclaire, S. Li, F. Marson, J. Lemus, C. Kotsalos, R. Conradin, C. Coreixas, R. Petkantchin, F. Raynaud, J. Beny, and B. Chopard. Palabos: Parallel Lattice Boltzmann Solver. *Computers & Mathematics with Applications*, apr 2020. ISSN 08981221. doi: 10.1016/j.camwa.2020.03.022.

[72] C. Law. Recent advances in droplet vaporization and combustion. *Progress in Energy and Combustion Science*, 8(3):171–201, jan 1982. ISSN 03601285. doi: 10.1016/0360-1285(82)90011-9.

[73] T. Lee and C. L. Lin. Rarefaction and compressibility effects of the lattice-Boltzmann-equation method in a gas microchannel. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 71(4):1–10, 2005. ISSN 15393755. doi: 10.1103/PhysRevE.71.046706.

[74] J. Li. *Multiscale and Multiphysics Flow Simulations of Using the Boltzmann Equation*. Springer International Publishing, 2020. ISBN 978-3-030-26465-9. doi: 10.1007/978-3-030-26466-6.

[75] L. Li, R. Mei, and J. F. Klausner. Lattice Boltzmann models for the convection-diffusion equation: D2Q5 vs D2Q9. *International Journal of Heat and Mass Transfer*, 108:41–62, may 2017. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2016.11.092.

[76] L. Li, X. Jia, and Y. Liu. Modified Outlet Boundary Condition Schemes for Large Density Ratio Lattice Boltzmann Models. *Journal of Heat Transfer*, 139(5):052003, 2017. ISSN 0022-1481. doi: 10.1115/1.4036001.

[77] Q. Li, K. H. Luo, and X. J. Li. Forcing scheme in pseudopotential lattice Boltzmann model for multiphase flows. *Physical Review E*, 86(1):016709, jul 2012. ISSN 1539-3755. doi: 10.1103/PhysRevE.86.016709.

[78] Q. Li, K. H. Luo, and X. J. Li. Lattice Boltzmann modeling of multiphase flows at large density ratio with an improved pseudopotential model. *Physical Review E*, 87(5):053301, may 2013. ISSN 1539-3755. doi: 10.1103/PhysRevE.87.053301.

[79] Q. Li, Q. Kang, M. Francois, Y. He, and K. Luo. Lattice Boltzmann modeling of boiling heat transfer: The boiling curve and the effects of wettability. *International Journal of Heat and Mass Transfer*, 85:787–796, jun 2015. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2015.01.136.

[80] Q. Li, K. Luo, Q. Kang, Y. He, Q. Chen, and Q. Liu. Lattice Boltzmann methods for multiphase flow and phase-change heat transfer. *Progress in Energy and Combustion Science*, 52(0):62–105, feb 2016. ISSN 03601285. doi: 10.1016/j.pecs.2015.10.001.

[81] Q. Li and K. H. Luo. Effect of the forcing term in the pseudopotential lattice Boltzmann modeling of thermal flows. *Physical Review E*, 89(5):053022, may 2014. ISSN 1539-3755. doi: 10.1103/PhysRevE.89.053022.

[82] Q. Li, P. Zhou, and H. J. Yan. Improved thermal lattice Boltzmann model for simulation of liquid-vapor phase change. *Physical Review E*, 96(6), 2017. ISSN 24700053. doi: 10.1103/PhysRevE.96.063303.

[83] Q. Li, J. Huang, and Q. Kang. On the temperature equation in a phase change pseudopotential lattice Boltzmann model. *International Journal of Heat and Mass Transfer*, 127:1112–1113, dec 2018. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2018.07.139.

[84] Q. Li, Y. Yu, and Z. X. Wen. How does boiling occur in lattice Boltzmann simulations? *Physics of Fluids*, 32(9):093306, sep 2020. ISSN 1070-6631. doi: 10.1063/5.0015491.

[85] W. Li, Y. Luo, J. Zhang, and W. J. Minkowycz. Simulation of Single Bubble Evaporation in a Microchannel in Zero Gravity With Thermocapillary Effect. *Journal of Heat Transfer*, 140(11):112403, jul 2018. ISSN 0022-1481. doi: 10.1115/1.4040147.

[86] Y. Li and X. Shan. Lattice Boltzmann method for adiabatic acoustics. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1944):2371–2380, 2011. ISSN 1364503X. doi: 10.1098/rsta.2011.0109.

[87] Z. Li, M. Yang, and Y. Zhang. Lattice Boltzmann method simulation of 3-D natural convection with double MRT model. *International Journal of Heat and Mass Transfer*, 94:222–238, mar 2016. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2015.11.042.

[88] C. Y. Lim, C. Shu, X. D. Niu, and Y. T. Chew. Application of lattice Boltzmann method to simulate microchannel flows. *Physics of Fluids*, 14(7):2299–2308, 2002. ISSN 10706631. doi: 10.1063/1.1483841.

[89] J. S. Lopez-Echeverry, S. Reif-Acherman, and E. Araujo-Lopez. Peng-Robinson equation of state: 40 years through cubics. *Fluid Phase Equilibria*, 447:39–71, 2017. ISSN 03783812. doi: 10.1016/j.fluid.2017.05.007.

[90] Q. Lou, Z. Guo, and B. Shi. Evaluation of outflow boundary conditions for two-phase lattice Boltzmann equation. *Physical Review E*, 87(6):063301, jun 2013. ISSN 1539-3755. doi: 10.1103/PhysRevE.87.063301.

[91] K. H. Luo, J. Xia, and E. Monaco. Multiscale Modeling of Multiphase Flow with Complex Interactions. *Journal of Multiscale Modelling*, 01(01):125–156, jan 2009. ISSN 1756-9737. doi: 10.1142/S1756973709000074.

[92] M. Magnini, B. Pulvirenti, and J. Thome. Numerical investigation of hydrodynamics and heat transfer of elongated bubbles during flow boiling in a microchannel. *International Journal of Heat and Mass Transfer*, 59(1):451–471, 2013. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2012.12.010.

[93] A. Márkus and G. Házi. Simulation of evaporation by an extension of the pseudopotential lattice Boltzmann method: A quantitative analysis. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 83(4):1–10, 2011. ISSN 15393755. doi: 10.1103/PhysRevE.83.046705.

[94] T. Mathew. Design of a MEMS micro-resistorjet, Master Thesis, Delft University of Technology, 2011. URL http://resolver.tudelft.nl/uuid:fecce3dd-c613-4a11-860f-7648d8fd42b9.

[95] A. Mazloomi M, S. S. Chikatamarla, and I. V. Karlin. Entropic Lattice Boltzmann Method for Multiphase Flows. *Physical Review Letters*, 114(17):174502, may 2015. ISSN 0031-9007. doi: 10.1103/PhysRevLett.114.174502.

[96] A. Mazloomi M., S. S. Chikatamarla, and I. V. Karlin. Entropic lattice Boltzmann method for multiphase flows: Fluid-solid interfaces. *Physical Review E*, 92(2):023308, aug 2015. ISSN 1539-3755. doi: 10.1103/PhysRevE.92.023308.

[97] P. Mercogliano, K. Takahashi, P. L. Vitagliano, and P. Catalano. *Parallel Computational Fluid Dynamics 2006*. Number May. 2007. ISBN 9780444530356. doi: 10.1016/B978-044453035-6/50027-4.

[98] A. A. Mohamad. *Lattice Boltzmann Method: Fundamentals and Engineering Applications*. Springer London, London, 2019. ISBN 978-1-4471-7422-6. doi: 10.1007/978-1-4471-7423-3.

[99] A. Montessori, G. Falcucci, M. La Rocca, S. Ansumali, and S. Succi. Three-Dimensional Lattice Pseudo-Potentials for Multiphase Flow Simulations at High Density Ratios. *Journal of Statistical Physics*, 161(6):1404–1419, dec 2015. ISSN 0022-4715. doi: 10.1007/s10955-015-1318-6.

[100] A. Montessori, P. Prestininzi, M. La Rocca, and S. Succi. Entropic lattice pseudo-potentials for multiphase flow simulations at high Weber and Reynolds numbers. *Physics of Fluids*, 29(9), 2017. ISSN 10897666. doi: 10.1063/1.5001253.

[101] A. Mukherjee, D. N. Basu, and P. K. Mondal. Mesoscopic Characterization of Bubble Dynamics in Flow Boilling following A Pseudopotential-based Approach. *arXiv*, sep 2020.

[102] S. Mukherjee. *Unravelling Turbulent Emulsions with lattice-Boltzmann simulations*. Doctoral thesis, Delft University of Technology, 2019. URL http://resolver.tudelft.nl/uuid:e98f7bbf-4639-49fc-96a4-b3543da637fc.

[103] S. Mukherjee, A. Zarghami, C. Haringa, K. van As, S. Kenjereš, and H. E. Van den Akker. Simulating liquid droplets: A quantitative assessment of lattice Boltzmann and Volume of Fluid methods. *International Journal of Heat and Fluid Flow*, 70(November 2017):59–78, apr 2018. ISSN 0142727X. doi: 10.1016/j.ijheatfluidflow.2017.12.001.

[104] T. Munekata, T. Suzuki, S. Yamakawa, and R. Asahi. Effects of viscosity, surface tension, and evaporation rate of solvent on dry colloidal structures: A lattice Boltzmann study. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 88(5):1–8, 2013. ISSN 15393755. doi: 10.1103/PhysRevE.88.052314.

[105] R. Muwanga, I. Hassan, and R. MacDonald. Characteristics of Flow Boiling Oscillations in Silicon Microchannel Heat Sinks. *Journal of Heat Transfer*, 129(10):1341, 2007. ISSN 00221481. doi: 10.1115/1.2754946.

[106] National Institute of Standards and Technology. NIST Chemistry WebBook - Water, 2018. URL https://webbook.nist.gov/cgi/cbook.cgi?ID=C7732185.

[107] X. Nie, X. Shan, and H. Chen. Thermal lattice Boltzmann model for gases with internal degrees of freedom. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 77(3):1–4, 2008. ISSN 15393755. doi: 10.1103/PhysRevE.77.035701.

[108] NUMECA. NUMECA International acquires FlowKit, 2018. URL https://www.numeca.com/numeca-acquires-flowkit-expanding-its-lattice-boltzmann-based-multiphysics-applications.

[109] T. Ohwada and K. Xu. The kinetic scheme for the full-Burnett equations. *Journal of Computational Physics*, 201(1):315–332, nov 2004. ISSN 00219991. doi: 10.1016/j.jcp.2004.05.017.

[110] X. F. Pan, A. Xu, G. Zhang, and S. Jiang. Lattice Boltzmann Approach to High-Speed Compressible Flows. *International Journal of Modern Physics C*, 18(11):1747–1764, jun 2007. ISSN 0129-1831. doi: 10.1142/S0129183107011716.

[111] P. Pavlo, G. Vahala, and L. Vahala. Higher order isotropic velocity grids in lattice methods. *Physical Review Letters*, 80(18):3960–3963, 1998. ISSN 10797114. doi: 10.1103/PhysRevLett.80.3960.

[112] C. Peng, S. Tian, G. Li, and M. C. Sukop. Simulation of multiple cavitation bubbles interaction with single-component multiphase Lattice Boltzmann method. *International Journal of Heat and Mass Transfer*, 137:301–317, 2019. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2019.03.096.

[113] Y. Peng, B. Wang, and Y. Mao. Study on Force Schemes in Pseudopotential Lattice Boltzmann Model for Two-Phase Flows. *Mathematical Problems in Engineering*, 2018:1–9, 2018. ISSN 1024-123X. doi: 10.1155/2018/6496379.

[114] R. Poyck. Design, manufacturing and characterisation of a water fed CubeSat micro-resistojet, Master Thesis, Delft University of Technology, 2014. URL http://resolver.tudelft.nl/uuid:87bc9f46-5f40-4a87-9716-65f7e3ac7c29.

[115] F. Qin, A. Mazloomi Moqaddam, Q. Kang, D. Derome, and J. Carmeliet. Entropic multiple-relaxation-time multirange pseudopotential lattice Boltzmann model for two-phase flow. *Physics of Fluids*, 30(3):032104, mar 2018. ISSN 1070-6631. doi: 10.1063/1.5016965.

[116] R. Qiu, Y. You, C. Zhu, and R. Chen. Lattice Boltzmann Simulation for Supersonic/Hypersonic Compressible Flows. In *21st AIAA International Space Planes and Hypersonics Technologies Conference*, pages 1–11, Reston, Virginia, mar 2017. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-463-3. doi: 10.2514/6.2017-2410.

[117] A. R. Rahmati and R. Ehsani. Simulation of Micro-Channel and Micro-Orifice Flow Using Lattice Boltzmann Method with Langmuir Slip Model. *Transport Phenomena in Nano and Micro Scales*, 5(1):1–8, 2017. doi: 10.7508/tpnms.2017.01.001.

[118] D. H. Rothman and J. M. Keller. Immiscible cellular-automaton fluids. *Journal of Statistical Physics*, 52 (3-4):1119–1127, aug 1988. ISSN 0022-4715. doi: 10.1007/BF01019743.

[119] R. Sadeghi, M. S. Shadloo, M. Y. A. Jamalabadi, and A. Karimipour. A three-dimensional lattice Boltzmann model for numerical investigation of bubble growth in pool boiling. *International Communications in Heat and Mass Transfer*, 79:58–66, dec 2016. ISSN 07351933. doi: 10.1016/j.icheatmasstransfer .2016.10.009.

[120] H. Safari, M. H. Rahimian, and M. Krafczyk. Extended lattice Boltzmann method for numerical simulation of thermal phase change in two-phase fluid flow. *Physical Review E*, 88(1):013304, jul 2013. ISSN 1539-3755. doi: 10.1103/PhysRevE.88.013304.

[121] M. Sbragaglia, R. Benzi, L. Biferale, S. Succi, K. Sugiyama, and F. Toschi. Generalized lattice Boltzmann method with multirange pseudopotential. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 75(2), 2007. ISSN 15393755. doi: 10.1103/PhysRevE.75.026702.

[122] V. P. Schulz, N. Abbaspour, T. Baumeister, and T. Röder. Lattice-Boltzmann Simulation and Experimental Validation of a Microfluidic T-Junction for Slug Flow Generation. *ChemEngineering*, 3(2):48, 2019. doi: 10.3390/chemengineering3020048.

[123] M. S. Shadloo. Numerical simulation of compressible flows by lattice Boltzmann method. *Numerical Heat Transfer, Part A: Applications*, 75(3):167–182, feb 2019. ISSN 1040-7782. doi: 10.1080/10407782.2 019.1580053.

[124] X. Shan. Pressure tensor calculation in a class of nonideal gas lattice Boltzmann models. *Physical Review E*, 77(6):066702, jun 2008. ISSN 1539-3755. doi: 10.1103/PhysRevE.77.066702.

[125] X. Shan and H. Chen. Lattice Boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, 47(3):1815–1819, mar 1993. ISSN 1063-651X. doi: 10.1103/PhysRevE.47.1815.

[126] X. Shan, X. F. Yuan, and H. Chen. Kinetic theory representation of hydrodynamics: A way beyond the Navier-Stokes equation. *Journal of Fluid Mechanics*, 550:413–441, 2006. ISSN 00221120. doi: 10.1017/ S0022112005008153.

[127] Y. Shi, T. S. Zhao, and Z. L. Guo. Thermal lattice Bhatnagar-Gross-Krook model for flows with viscous heat dissipation in the incompressible limit. *Physical Review E*, 70(6):066310, dec 2004. ISSN 1539-3755. doi: 10.1103/PhysRevE.70.066310.

[128] M. A. C. Silva, D. C. Guerrieri, H. van Zeijl, A. Cervone, and E. Gill. Vaporizing Liquid Microthrusters with integrated heaters and temperature measurement. *Sensors and Actuators A: Physical*, 265:261–274, oct 2017. ISSN 09244247. doi: 10.1016/j.sna.2017.07.032.

[129] M. A. C. Silva, D. C. Guerrieri, A. Cervone, and E. Gill. Topology Optimization of Heating Chamber of Vaporizing Liquid Microthrusters. In *ESA Space Propulsion 2018 Conference*, Seville, Spain, 2018.

[130] S. Srivastava. *Lattice Boltzmann method for contact line dynamics*. Doctoral thesis, Technische Universiteit Eindhoven, 2014.

[131] M. E. Steinke and S. G. Kandlikar. An Experimental Investigation of Flow Boiling Characteristics of Water in Parallel Microchannels. *Journal of Heat Transfer*, 126(4):518, 2004. ISSN 00221481. doi: 10.1 115/1.1778187.

[132] H. Struchtrup. Failures of the Burnett and super-Burnett equations in steady state processes. *Continuum Mechanics and Thermodynamics*, 17(1):43–50, apr 2005. ISSN 0935-1175. doi: 10.1007/s00161-004-0186-0.

[133] S. Succi, G. Amati, M. Bernaschi, G. Falcucci, M. Lauricella, and A. Montessori. Towards Exascale Lattice Boltzmann computing. *Computers & Fluids*, 181:107–115, mar 2019. ISSN 00457930. doi: 10.1016/j.compfluid.2019.01.005.

[134] K. Sun, T. Wang, M. Jia, and G. Xiao. Evaluation of force implementation in pseudopotential-based multiphase lattice Boltzmann models. *Physica A: Statistical Mechanics and its Applications*, 391(15):3895–3907, aug 2012. ISSN 03784371. doi: 10.1016/j.physa.2012.03.008.

[135] M. R. Swift, W. R. Osborn, and J. M. Yeomans. Lattice Boltzmann Simulation of Nonideal Fluids. *Physical Review Letters*, 75(5):830–833, jul 1995. ISSN 0031-9007. doi: 10.1103/PhysRevLett.75.830.

[136] M. R. Swift, E. Orlandini, W. R. Osborn, and J. M. Yeomans. Lattice Boltzmann simulations of liquid-gas and binary fluid systems. *Physical Review E*, 54(5):5041–5052, nov 1996. ISSN 1063-651X. doi: 10.1103/PhysRevE.54.5041.

[137] J. Thome. Boiling in microchannels: a review of experiment and theory. *International Journal of Heat and Fluid Flow*, 25(2):128–139, apr 2004. ISSN 0142727X. doi: 10.1016/j.ijheatfluidflow.2003.11.005.

[138] Z. Tian, H. Xing, Y. Tan, S. Gu, and S. D. Golding. Reactive transport LBM model for CO2 injection in fractured reservoirs. *Computers & Geosciences*, 86:15–22, jan 2016. ISSN 00983004. doi: 10.1016/j.cageo.2015.10.002.

[139] F. L. Torre. *Gas Flow in Miniaturized Nozzles for Micro-Thrusters*. Phd thesis, Delft University of Technology, 2011.

[140] F. L. Torre, S. Kenjeres, J.-L. P. Moerel, B. Zandbergen, and C. R. Kleijn. Influence of boundary layer formation and surface roughness on the thrust of micro-nozzles. In *5th International Spacecraft Propulsion Conference*, Heraklion-Crete, Greece, may 2008.

[141] R. Trunk, T. Henn, W. Dörfler, H. Nirschl, and M. J. Krause. Inertial dilute particulate fluid flow simulations with an Euler–Euler lattice Boltzmann method. *Journal of Computational Science*, 17:438–445, nov 2016. ISSN 18777503. doi: 10.1016/j.jocs.2016.03.013.

[142] TU Delft Space Institute. Resistojet: a tiny thruster, 2019. URL https://spaceinstitute.tudelft.nl/showcase/resistojet-a-tiny-thruster/.

[143] M. Varmazyar and M. Bazargan. Development of a thermal lattice Boltzmann method to simulate heat transfer problems with variable thermal conductivity. *International Journal of Heat and Mass Transfer*, 59(1):363–371, 2013. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2012.12.014.

[144] H. Versteeg. Novel fabrication method for a hot gas supersonic micro-thruster, Master Thesis, Delft University of Technology, 2020. URL http://resolver.tudelft.nl/uuid:ac2482ad-0f8e-4569-8bd4-fd11bd6327bd.

[145] T. X. V. Wees. Characterization and Testing of a MEMS-Vaporizing Liquid Microthruster for Small Satellite Propulsion, Master Thesis, Delft University of Technology, 2017. URL http://resolver.tudelft.nl/uuid:f698fca7-205b-487d-b521-83d7f85e6fef.

[146] Y. Wi, J. Kim, J. Lee, and J. Lee. Optimal Patterned Wettability for Microchannel Flow Boiling Using the Lattice Boltzmann Method. *MDPI Coatings*, 8(8):288, 2018. doi: 10.3390/coatings8080288.

[147] L. T. Williams, M. McDonald, and M. Osborn. Performance Characterization of a Low Reynolds Number Micro-Nozzle Flow. In *51st AIAA/SAE/ASEE Joint Propulsion Conference*, pages 1–10, Reston, Virginia, jul 2015. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-321-6. doi: 10.2514/6.2015-3924.

[148] A. Xu, T. Zhao, L. An, and L. Shi. A three-dimensional pseudo-potential-based lattice Boltzmann model for multiphase flows with large density ratio and variable surface tension. *International Journal of Heat and Fluid Flow*, 56:261–271, dec 2015. ISSN 0142727X. doi: 10.1016/j.ijheatfluidflow.2015.08.001.

[149] F. Yang, X. Dai, Y. Peles, P. Cheng, J. Khan, and C. Li. Flow boiling phenomena in a single annular flow regime in microchannels (II): Reduced pressure drop and enhanced critical heat flux. *International Journal of Heat and Mass Transfer*, 68:716–724, 2014. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2013.09.060.

[150] P. Yuan and L. Schaefer. Equations of state in a lattice Boltzmann model. *Physics of Fluids*, 18(4):042101, apr 2006. ISSN 1070-6631. doi: 10.1063/1.2187070.

[151] B. Zandbergen. *Course AE4S01 Thermal Rocket Propulsion*. Delft University of Technology, 2018.

[152] A. Zarghami and H. E. A. Van den Akker. Thermohydrodynamics of an evaporating droplet studied using a multiphase lattice Boltzmann method. *Physical Review E*, 95(4):043310, apr 2017. ISSN 2470-0045. doi: 10.1103/PhysRevE.95.043310.

[153] D. Zhang, K. Papadikis, and S. Gu. Three-dimensional multi-relaxation time lattice-Boltzmann model for the drop impact on a dry surface at large density ratio. *International Journal of Multiphase Flow*, 64:11–18, 2014. ISSN 03019322. doi: 10.1016/j.ijmultiphaseflow.2014.04.005.

[154] R. Zhang and H. Chen. Lattice Boltzmann method for simulations of liquid-vapor thermal flows. *Physical Review E*, 67(6):066711, jun 2003. ISSN 1063-651X. doi: 10.1103/PhysRevE.67.066711.

[155] W. Zhao and W.-A. Yong. Relaxation-rate formula for the entropic lattice Boltzmann model. *Chinese Physics B*, 28(11):114701, nov 2019. ISSN 1674-1056. doi: 10.1088/1674-1056/ab48f0.

[156] S. Zhou, X. Xu, and B. G. Sammakia. Modeling of boiling flow in microchannels for nucleation characteristics and performance optimization. *International Journal of Heat and Mass Transfer*, 64:706–718, 2013. ISSN 00179310. doi: 10.1016/j.ijheatmasstransfer.2013.05.031.

[157] B. Zohuri. Table and Graph Compilations. In *Physics of Cryogenics*, pages 565–693. Elsevier, 2018. ISBN 9780128145197. doi: 10.1016/B978-0-12-814519-7.15001-3. URL https://linkinghub.elsevier.com/retrieve/pii/B9780128145197150013.

[158] B. Zohuri. Thermodynamic Relations. In *Physics of Cryogenics*, pages 245–257. Elsevier, 2018. ISBN 9780128145197. doi: 10.1016/B978-0-12-814519-7.00010-0. URL https://linkinghub.elsevier.com/retrieve/pii/B9780128145197000100.

[159] B. Zohuri. Properties of Pure Substances. In *Physics of Cryogenics*, pages 53–79. Elsevier, 2018. ISBN 9780128145197. doi: 10.1016/B978-0-12-814519-7.00002-1. URL https://linkinghub.elsevier.com/retrieve/pii/B9780128145197000021.

[160] Q. Zou and X. He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598, jun 1997. ISSN 1070-6631. doi: 10.1063/1.869307.

# A Trade-off simulation tool

This appendix details the simulation tool selection trade-off process. The trade-off was performed halfway 2019. The resultant simulation tool should be able to model the flow inside the vaporizing liquid microthruster (VLM) currently being developed by the space engineering department at the TUDelft. Inside the VLM, which can be regarded as a heated microchannel microelectromechanical systems (MEMS) device, a complex phase change flow is expected to occur.

## A.1. Applicable models and tools

This section lists all the available models and tools and compares the applicability to model the multiphase flow inside the heating chamber.

A macroscopic solver uses the conventional fluid equations, such as the Navier-Stokes, to solve the fluid flow. The simulation parameters are macroscopic quantities, such as velocity vector, density scalar, and temperature scalar.

Microscopic solvers uses molecular dynamics fluid equations to solve the flow. They usually track individual molecules. Mesoscopic solvers are in between the macroscopic and microscopic. They solve the macroscopic fluid problems using a simplified microscopic equations.

### A.1.1. Macroscopic solvers

There are many open source and commercial macroscopic solvers available, a small list of the most relevant ones are given below, where open source solvers are marked as such:

- OpenFoam (open source)
- COMSOL
- Ansys Fluent

The advantage to OpenFoam is that the source code is available to look at and the mechanism revealed. However, COMSOL and Ansys Fluent have the advantage that they have a graphical interface and meshing is greatly easier. In addition, Ansys fluent has a hybrid solver that swaps from volume of fluid (VoF) to a discrete phase model (DPM) once droplets reached a certain shape. However, this model not conserve mass when the swap happens, resulting in devastating local density fluctuations. Furthermore, Ansys Student is limited to a 512k mesh (https://www.ansys.com/academic/free-student-products), which limits the solution size.

If required to choose from this list, OpenFoam would be selected as the main solver, but meshing would be done via a commercial option such as Ansys ICEM.

### A.1.2. Mesoscopic solvers

Of the mesoscopic solvers, the most applicable method is the lattice Boltzmann method (LBM) since tools for dissipative particle dynamics (DPD) and Bhatnagar-Gross-Krook (BGK) solvers are sparse. There are several tools available which use the lattice Boltzmann method and are listed below.

Commerical options are:

- PowerFLOW https://exa.com/en/company/exa-lattice-boltzmann-technology
- NUMECA OmnisLB https://www.numeca.com/product/omnislb

Open source options are:

- Palabos http://www.palabos.org/
- OpenLB https://www.openlb.net/

Since the access to the commercial solver is limited due to a missing licence and closed source code, a open source option is preferred. Of the two mentioned here, it is important to note the state of development for each of them. Palabos split off of OpenLB years ago and made a strong headstart in development. However, the company behind Palabos, FlowKit, (https://www.flowkit.com/) was acquired by NUMECA in August 2018. Thus the development stopped with the last version released in September 2017. With the 2019 May release OpenLB is catching up to the features of Palabos and has a more active community. Thus if performance wise there is no difference, and both can be executed on the simulation platform, OpenLB is preferred over Palabos.

## A.2. Tool selection

This section will detail the tool selection process, where suitable tools were identified in appendix A.1. The procedure is split into three parts: determining selection criteria, determining criteria weights, and performing the trade-off. The trade-off is scored with a $-$, $\cdot$ or $+$, to show whether the criteria is insufficient, sufficient or satisfactory.

### A.2.1. Determining selection criteria

The selection criteria are used in the trade-off to score and compare various tool. They are introduced below with an explanation of what affects it.

**Features**
This criterium looks into whether the methods implemented in the tool are sufficient in modelling the VLM heating chamber problem.

**Performance**
This criterium shows how well it can solve the heating chamber problem. This takes into account simulation accuracy and simulation duration

**Accessibility**
This criterium is determined by how easy it is to acquire the tool, if there are limitations and if it is open source.

**Support**
This criterium is scored by the available documentation to get started with the tool, and how active the community is to answering questions.

### A.2.2. Determining criteria weights

The selection criteria weights are determined via the Analytic Hierarchy Process (AHP). For this the criteria are weighted between each other, to get a final resultant unbiased score. A consistency score below 10% is acceptable. The AHP decision matrix is shown in table A.1, with the resultant weights, shown in table A.2. The consistency ratio is 1.7%, which is below the maximum threshold, and thus are considered consistent.

For the pair wise comparisons done in table A.1, the decision making is explained for the features and performance criterion. Features was deemed more important, since it does not matter how well a tool performs or how accessible it is, if the necessary methods are missing, since it is not applicable for use. Performance

Table A.1: Tool selection decision matrix

| | Features | Performance | Accessibility | Support |
|---|---|---|---|---|
| Features | 1 | 2 | 2 | 3 |
| Performance | 0.5 | 1 | 1 | 3 |
| Accessibility | 0.5 | 1 | 1 | 2 |
| Support | 0.33 | 0.33 | 0.5 | 1 |

Table A.2: Tool selection criterion priorities

| Criterion | Weight |
|---|---|
| Features | 41.8% |
| Performance | 25.0% |
| Accessibility | 22.3% |
| Support | 11.0% |

is a driving factor since one does not have infinite amount of time, but if the tool usage is limited then the performance is not the driving factor, thus the weight is equal to accessibility.

### A.2.3. Tool trade-off

The tool trade-off table is shown in table A.3. From this trade-off table, the best tool is OpenLB for the following reasons:

Feature wise, OpenLB uses the lattice Boltzmann method to simulate multiphase flow, which has methods for phase change multiphase flows as long as the Mach number is small. However, almost all selected tools can simulate phase change multiphase flow, depending on the model in the package.

The lattice Boltzmann method has a performance advantage over conventional CFD for multiphase flows. Since OpenLB is open source with no limitations, it is very accessible. The support that OpenLB offers is sufficient, since the documentation is in parts missing and while the community is active, it is small compared to OpenFOAM or Ansys.

Table A.3: Trade-off for tool selection

| Tool | Features | Performance | Accessibility | Support |
|---|---|---|---|---|
| OpenFOAM | + | · | + | + |
| Ansys | + | · | · | + |
| COMSOL | + | · | · | + |
| PowerFlow | + | + | − | + |
| OmnisLB | + | + | − | + |
| OpenLB | + | + | + | · |
| Palabos | + | + | + | − |
| LAMMPS | − | · | + | − |

### A.2.4. Post thesis tool selection discussion

In the year between the writing of this thesis and the literature study, one aspect has changed. Palabos was picked up by the Geneva University and as such the support assessment should be revised. When choosing between OpenLB and Palabos it is noteworthy that Palabos is more often used in the academic field, and all other criteria being equal, Palabos wins over OpenLB in that regard.

It should also be noted that while OpenLB and Palabos has multiphase and thermal flow, they were never combined. Thus the assessment on the features criterium was incorrect, since phase change due to heating

is not a method available. In this thesis, OpenLB was extended to include this thermal multiphase flow with phase change, thus making the trade-off table entry for OpenLB correct for the tool selection. Palabos does not have this, and thus its entry is still incorrect.

# B Ideal rocket equation multiphase specific impulse derivation

This chapter details the derivation of the specific impulse derivative of an ideal rocket nozzle with multiphase flow. These derivatives allow for determining the type of multiphase flow occurring inside the nozzle from the change in specific impulse only. This is a helpful analysis tool which can be applied to experimental results without the addition of further hardware.

Four categories of multiphase flow can be identified. These are related to two assumptions that will be taken during the derivations. The multiphase flow can be thermal or isothermal, and accelerating or non-accelerating.

The thermal assumption says that the liquid phase has the same temperature as the gas phase. This is the case when there is enough time for the liquid to exchange its thermal energy to the surrounding gas. The isothermal assumption states that the liquid phase stays at the same temperature as in the chamber, and no heat exchange happens with the surrounding gas.

The accelerating assumptions says that the liquid phase velocity is the same as the surrounding gas phase. For the non-accelerating assumption, the liquid phase velocity is the same as in the chamber, which is approximately 0.

Different types of multiphase flow can be sorted into a different category. Dispersed fine droplets are an example of the thermal accelerating category. Whereas annular flow is an example of isothermal non-accelerating flow.

## B.1. Multiphase energy equation derivation

For a multi-phase flow the effective $I_{sp}$ can be expressed as a weighted sum of all flow components, as shown in eq. (B.1)[151].

$$\mathrm{I_{sp_{eff}}} = \frac{1}{\mathrm{g_0}} \frac{\sum \dot{\mathrm{m}}_\mathrm{i} \cdot \mathrm{U_{e_i}}}{\sum \dot{\mathrm{m}}_\mathrm{i}} \tag{B.1}$$

Thus the theoretical performance impact of droplets is dependent on their mass flow, exit velocity, and impact on the surrounding flow's exit velocity. It is possible to give analytical solutions for specific impulse if certain assumptions regarding the multiphase flow are taken into account. Using the same method as in the ideal rocket theory, the ideal multiphase flow performance can be calculated with the conservation of energy eq. (B.2). Applying the ideal rocket theory assumptions regarding negligible chamber velocity, and assuming only two phases, with the gas being ideal, the equation can be written as given by eq. (B.6).

The following symbols are used: $\dot{m}$ is massflow [kg $s^{-1}$], $c_p$ is specific heat capacity at constant pressure [J $kg^{-1}$ $K^{-1}$], $T$ is temperature [K], $U$ is velocity [m $s^{-1}$], $\phi$ is the liquid massflow fraction [-].

The subscripts $_l$ and $_g$ refer to the liquid and gas phase respectively. For properties at specific locations, $_e$ is at the nozzle exit, $_c$ in the chamber. For convenience, the specific heat capacity for liquid and gas at constant pressure are given by $c_l$ and $c_g$ respectively, dropping the $_p$ for the gas.

Note that in this appendix $T_c$ refers to the chamber temperature, which is most commonly used in literature for ideal rocket theory. In the main report, $T_c$ is the critical temperature. When referring to the chamber

temperature in the main report $T_{ch}$ is used.

$$\sum_i \left[ \dot{m}_i c_{p_i} T_i + \frac{1}{2} \dot{m}_i U_i^2 \right] = \text{const} \tag{B.2}$$

$$\dot{m}_l c_l T_{l_1} + \dot{m}_g c_g T_{g_1} = \dot{m}_l c_l T_{l_2} + \dot{m}_g c_g T_{g_2} + \frac{1}{2} \dot{m}_l U_{l_2}^2 + \frac{1}{2} \dot{m}_g U_{g_2}^2 \tag{B.3}$$

$$\phi = \frac{\dot{m}_l}{\dot{m}_g + \dot{m}_l} \quad \Rightarrow \quad 1 - \phi = \frac{\dot{m}_g}{\dot{m}_g + \dot{m}_l} \tag{B.4}$$

$$\phi c_l T_{l_1} + (1-\phi) c_g T_{g_1} = \phi c_l T_{l_2} + (1-\phi) c_g T_{g_2} + \frac{1}{2} \phi U_{l_2}^2 + \frac{1}{2} (1-\phi) U_{g_2}^2 \tag{B.5}$$

$$\frac{\phi}{(1-\phi)} c_l T_{l_1} + c_g T_{g_1} = \frac{\phi}{(1-\phi)} c_l T_{l_2} + c_g T_{g_2} + \frac{1}{2} \frac{\phi}{(1-\phi)} U_{l_2}^2 + \frac{1}{2} U_{g_2}^2 \tag{B.6}$$

Assuming that both liquid and vapour start with the same temperature $T_{g_1} = T_{l_1} = T_c$, the equation can be rewritten to eq. (B.7).

$$T_c \left( \frac{\phi}{(1-\phi)} c_l + c_g \right) = \frac{\phi}{(1-\phi)} c_l T_{l_2} + c_g T_{g_2} + \frac{1}{2} \frac{\phi}{(1-\phi)} U_{l_2}^2 + \frac{1}{2} U_{g_2}^2 \tag{B.7}$$

## B.2. Multiphase performance impact derivation

For each of the four categories the multiphase specific impulse can be defined. The isentropic relations are used to to determine the temperature at the exit. Additional symbols are: $\gamma$ is the heat capacity ratio [-], $I_{sp}$ is the specific impulse [s], $P$ is pressure [$\text{Nm}^{-2}$], $R_{sp}$ is the specific gas constant [$Jkg^{-1}K^{-1}$].

The linear conservation of momentum and energy for steady state adiabatic flow in terms of the massflow fraction can be expressed as eq. (B.8) and eq. (B.10). The ideal gas law is used to eliminate $\rho_g$ in eq. (B.8), resulting in eq. (B.9). To calculate the exhaust temperature $T_e$ for a category, combine eq. (B.9) and eq. (B.10), apply the assumptions, and integrate.

$$-\frac{\mathrm{d}p}{\rho_g} = \frac{\phi}{1-\phi} U_l \mathrm{d}U_l + U_g \mathrm{d}U_g \tag{B.8}$$

$$-R_{\mathrm{sp,g}} T_g \frac{\mathrm{d}p}{p} = \frac{\phi}{1-\phi} U_l \mathrm{d}U_l + U_g \mathrm{d}U_g \tag{B.9}$$

$$\frac{\phi}{1-\phi} c_l \mathrm{d}T_l + c_g \mathrm{d}T_g + \frac{\phi}{1-\phi} U_l \mathrm{d}U_l + U_g \mathrm{d}U_g = 0 \tag{B.10}$$

### Thermal accelerating

From the thermal assumption $T_{l_2} = T_{g_2} = T_e$ and from the accelerating assumption $U_{l_2} = U_{g_2} = U_e$.

$$T_c \left( \frac{\phi}{(1-\phi)} c_l + c_g \right) = \frac{\phi}{(1-\phi)} c_l T_e + c_g T_e + \frac{1}{2} \frac{\phi}{(1-\phi)} U_e^2 + \frac{1}{2} U_e^2 \tag{B.11}$$

$$= \left( \frac{\phi}{(1-\phi)} c_l + c_g \right) T_e + \frac{1}{2(1-\phi)} U_e^2 \tag{B.12}$$

$$U_e = \sqrt{2 \cdot (1-\phi)\left(\frac{\phi}{(1-\phi)} c_l + c_g\right)(T_c - T_e)} \tag{B.13}$$

$$= \sqrt{2 \cdot \left(\phi\, c_l + (1-\phi)\, c_g\right) \cdot T_c \cdot \left[1 - \left(\frac{P_e}{P_c}\right)^n\right]} \tag{B.14}$$

$$n = \frac{R_{\text{sp,g}}}{\frac{\phi}{(1-\phi)} c_l + c_g} \tag{B.15}$$

$$I_{sp} = \frac{1}{g_0}\left(\phi U_{l_2} + (1-\phi) U_{g_2}\right) \tag{B.16}$$

$$= \frac{1}{g_0} U_e \tag{B.17}$$

## Isothermal accelerating

From the isothermal assumption $T_{l_2} = T_{l_1} = T_c$ and from the accelerating assumption $U_{l_2} = U_{g_2} = U_e$.

$$T_c\left(\frac{\phi}{(1-\phi)} c_l + c_g\right) = \frac{\phi}{(1-\phi)} c_l T_c + c_g T_e + \frac{1}{2}\frac{1}{(1-\phi)} U_e^2 \tag{B.18}$$

$$U_e = \sqrt{2(1-\phi) \cdot c_g \cdot T_c \cdot \left(1 - \left(\frac{P_e}{P_c}\right)^{\frac{\gamma-1}{\gamma}}\right)} \tag{B.19}$$

$$I_{sp} = \frac{1}{g_0}\left(\phi U_{l_2} + (1-\phi) U_{g_2}\right) = \frac{1}{g_0} U_e \tag{B.20}$$

## Thermal non-accelerating

From the thermal assumption $T_{l_2} = T_{g_2} = T_e$ and from the non-accelerating assumption $U_{l_2} = U_{l_1} \approx 0$.

$$T_c\left(\frac{\phi}{(1-\phi)} c_l + c_g\right) = \frac{\phi}{(1-\phi)} c_l T_e + c_g T_e + \frac{1}{2} U_{g_e}^2 \tag{B.21}$$

$$= \left(\frac{\phi}{(1-\phi)} c_l + c_g\right) T_e + \frac{1}{2} U_{g_e}^2 \tag{B.22}$$

$$U_{g_e} = \sqrt{2 \cdot \left(\frac{\phi}{(1-\phi)} c_l + c_g\right) \cdot T_c \cdot \left(1 - \left(\frac{P_e}{P_c}\right)^n\right)} \tag{B.23}$$

$$n = \frac{R_{\text{sp,g}}}{\frac{\phi}{(1-\phi)} c_l + c_g} \tag{B.24}$$

$$I_{sp} = \frac{1}{g_0}\left(\phi U_{l_2} + (1-\phi) U_{g_2}\right) = \frac{1}{g_0}(1-\phi) U_{g_e} \tag{B.25}$$

### Isothermal non-accelerating

From the isothermal assumption $T_{l_2} = T_{l_1} = T_c$ and from the non-accelerating assumption $U_{l_2} = U_{l_1} \approx 0$.

$$T_c \left( \frac{\phi}{(1-\phi)} c_l + c_g \right) = \frac{\phi}{(1-\phi)} c_l T_c + c_g T_e + \frac{1}{2} U_{g_e}^2 \tag{B.26}$$

$$T_c c_g = c_g T_e + \frac{1}{2} U_{g_e}^2 \tag{B.27}$$

$$U_{g_e} = \sqrt{2 \cdot c_g \cdot T_c \cdot \left( 1 - \left( \frac{P_e}{P_c} \right)^{\frac{\gamma-1}{\gamma}} \right)} \tag{B.28}$$

$$I_{sp} = \frac{1}{g_0} \left( \phi U_{l_2} + (1-\phi) U_{g_2} \right) = \frac{1}{g_0} (1-\phi) U_{g_e} \tag{B.29}$$

The isothermal non-accelerating category shows that the exhaust velocity, eq. (B.28), is the same as for single phase. That means that the liquid has no impact at all on the exhaust velocity. The drop in specific impulse is then due to the massflow of the liquid not contributing to the thrust generation.

## B.3. Specific impulse derivative

In this section, the derivative of the specific impulse with respect to $\phi$ is calculated, $\frac{\partial I_{sp}}{\partial \phi}$. Since $\frac{\partial \phi}{\partial \dot{m}}$ is known, see eq. (B.30), the derivative of the $I_{sp}$ with respect to the massflow rate can be found, $\frac{\partial I_{sp}}{\partial \dot{m}}$.

This derivative, $\frac{\partial I_{sp}}{\partial \dot{m}}$, allows for another way to gain insight into the type of multiphase flow occurring. The specific impulse and total massflow rate are important parameters of any thruster test setup. Plotting specific impulse vs massflow rate, a multiphase flow is usually shown as a drop in specific impulse. This drop can be used to calculate the specific impulse derivative experimentally.

Comparing the experimental derivative with the theoretical derivative allows one to categorize the experimental multiphase flow into one of the four categories. Thus additional information of the multiphase flow can be learned from the derivative, without adding anything to the experimental setup.

It is interesting to see that the change in $I_{sp}$ with respect to $\phi$ is a function of itself divided by seemingly error/correction terms. It would be interesting to attribute these error/correction terms to certain physical phenomena.

### $\phi$ derivative

As was mentioned previously, the $\frac{\partial \phi}{\partial \dot{m}}$ is know. The derivation below shows how to get that expression.

$$\begin{aligned}\frac{\partial \phi}{\partial \dot{m}} &= \frac{\partial}{\partial \dot{m}}\left(\frac{\dot{m}_l}{\dot{m}}\right)\\ &= \frac{\partial}{\partial \dot{m}}\left(\frac{\dot{m}-\dot{m}_g}{\dot{m}}\right)\\ &= \frac{\partial}{\partial \dot{m}}\left(1-\frac{\dot{m}_g}{\dot{m}}\right)\\ &= \frac{\dot{m}_g}{\dot{m}^2}\\ &= \frac{1}{\dot{m}}(1-\phi)\end{aligned} \tag{B.30}$$

## Thermal accelerating

Due to the complexity of the derivative, this was done using sympy (symbolic python). The derivative of $n$ with respect to the $\phi$ is neglected.

$$\frac{\partial I_{sp}}{\partial \phi} = -\frac{\sqrt{T_{ch}\left(1-\left(\frac{P_e}{P_{ch}}\right)^n\right)\left(2c_g\left(1-\phi\right)+2c_l\phi\right)}\left(-2c_g+2c_l\right)}{2g_0\left(2c_g\left(1-\phi\right)+2c_l\phi\right)} \tag{B.31}$$

$$\begin{aligned}\frac{\partial I_{sp}}{\partial \phi} &= -\frac{U_e\left(-c_g+c_l\right)}{2g_0\left(c_g\left(1-\phi\right)+c_l\phi\right)}\\ &= \frac{-I_{sp}\left(-c_g+c_l\right)}{2\left(c_g\left(1-\phi\right)+c_l\phi\right)}\end{aligned}$$

$$\begin{aligned}\frac{\partial I_{sp}}{\partial \dot{m}} &= \frac{\partial I_{sp}}{\partial \phi}\cdot\frac{\partial \phi}{\partial \dot{m}}\\ &= \frac{-I_{sp}\left(-c_g+c_l\right)}{2\left(c_g\left(1-\phi\right)+c_l\phi\right)}\cdot\frac{(1-\phi)}{\dot{m}}\end{aligned}$$

$$\frac{\partial I_{sp}}{\partial \dot{m}} = -\frac{I_{sp}}{2\dot{m}}\cdot\frac{\left(c_l-c_g\right)}{\left(c_g+c_l\frac{\phi}{1-\phi}\right)} \tag{B.32}$$

## Isothermal accelerating

$$\frac{\partial I_{sp}}{\partial \phi} = -\frac{1}{g_0}\frac{\partial U_e}{\partial \phi} \tag{B.33}$$

$$= -\frac{1}{g_0}\frac{2 \cdot c_g \cdot T_c \cdot \left(1 - \left(\frac{P_e}{P_c}\right)^{\frac{\gamma-1}{\gamma}}\right)}{2\sqrt{2(1-\phi) \cdot c_g \cdot T_c \cdot \left(1 - \left(\frac{P_e}{P_c}\right)^{\frac{\gamma-1}{\gamma}}\right)}} \tag{B.34}$$

$$= -\frac{1}{g_0}\frac{U_e}{2(1-\phi)} \tag{B.35}$$

$$= -\frac{I_{sp}}{2(1-\phi)} \tag{B.36}$$

$$\frac{\partial I_{sp}}{\partial \dot{m}} = \frac{\partial I_{sp}}{\partial \phi} \cdot \frac{\partial \phi}{\partial \dot{m}} \tag{B.37}$$

$$= -\frac{I_{sp}}{2(1-\phi)} \cdot \frac{(1-\phi)}{\dot{m}} \tag{B.38}$$

$$= -\frac{I_{sp}}{2\dot{m}} \tag{B.39}$$

## Thermal non-accelerating

Due to the complexity of the derivative, this was done using sympy (symbolic python). The derivative of $n$ with respect to the $\phi$ is neglected.

$$\frac{\partial I_{sp}}{\partial \phi} = -\frac{\sqrt{T_{ch}\left(1 - \left(\frac{P_e}{P_{ch}}\right)^n\right)\left(2c_g + \frac{2c_l\phi}{1-\phi}\right)}\left(\frac{2c_l\phi}{(1-\phi)^2} + \frac{2c_l}{1-\phi}\right)}{2g_0\left(2c_g + \frac{2c_l\phi}{1-\phi}\right)} \tag{B.40}$$

$$\frac{\partial I_{sp}}{\partial \phi} = -\frac{U_e}{2g_0}\frac{\left(\frac{2c_l\phi}{(1-\phi)^2} + \frac{2c_l}{1-\phi}\right)}{\left(2c_g + \frac{2c_l\phi}{1-\phi}\right)}$$

$$= -\frac{U_e}{2g_0}\frac{c_l\left(\frac{\phi}{(1-\phi)^2} + \frac{1}{1-\phi}\right)}{\left(c_g + \frac{c_l\phi}{1-\phi}\right)}$$

$$\frac{\partial I_{sp}}{\partial \dot{m}} = -\frac{\partial I_{sp}}{\partial \phi} \cdot \frac{\partial \phi}{\partial \dot{m}}$$

$$= -\frac{U_e}{2g_0}\frac{c_l\left(\frac{\phi}{(1-\phi)^2} + \frac{1}{1-\phi}\right)}{\left(c_g + \frac{c_l\phi}{1-\phi}\right)} \cdot \frac{(1-\phi)}{\dot{m}}$$

$$\frac{\partial I_{sp}}{\partial \dot{m}} = -\frac{I_{sp}}{2\dot{m}}\frac{c_l\left(\frac{\phi}{(1-\phi)} + 1\right)}{\left(c_g + c_l\frac{\phi}{1-\phi}\right)} \tag{B.41}$$

## Isothermal non-accelerating

$$\frac{\partial I_{sp}}{\partial \phi} = \frac{\partial}{\partial \phi}\left(\frac{1}{g_0}(1-\phi)U_e\right) \tag{B.42}$$

$$= \frac{1}{g_0}\frac{\partial(1-\phi)}{\partial \phi}U_e + \frac{1}{g_0}(1-\phi)\frac{\partial U_e}{\partial \phi} \tag{B.43}$$

$$\rightarrow \frac{\partial U_e}{\partial \phi} = 0 \tag{B.44}$$

$$= -\frac{1}{g_0}U_e \tag{B.45}$$

$$= -\frac{I_{sp}}{1-\phi} \tag{B.46}$$

$$\frac{\partial I_{sp}}{\partial \dot{m}} = \frac{\partial I_{sp}}{\partial \phi} \cdot \frac{\partial \phi}{\partial \dot{m}} \tag{B.47}$$

$$= -\frac{I_{sp}}{1-\phi} \cdot \frac{(1-\phi)}{\dot{m}} \tag{B.48}$$

$$= -\frac{I_{sp}}{\dot{m}} \tag{B.49}$$

# C Thermodynamic consistency results 2D

This appendix contains the P-V and density ratio plots for the thermodynamic consistency test. The liquid density is impacted little by the choice of $a$ and $\sigma$ and as such P-$\rho$ plots are not given.

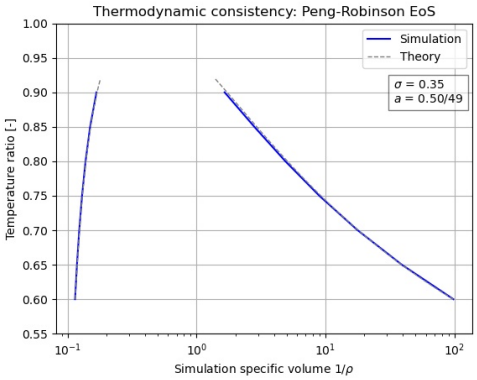



Figure C.1: P-V plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.35$ and $a = \frac{0.5}{49}$

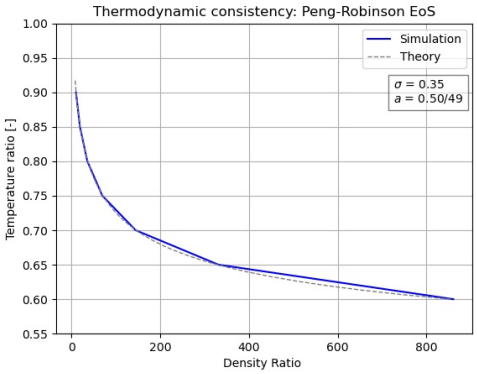



Figure C.2: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.35$ and $a = \frac{0.5}{49}$

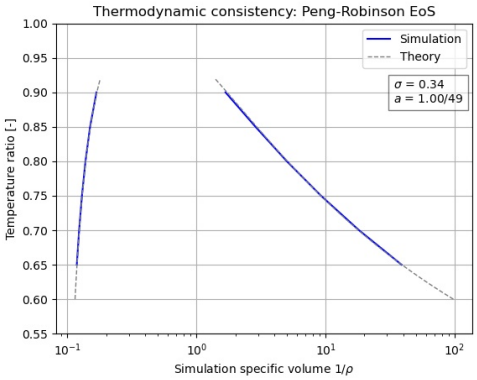



Figure C.3: P-V plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.34$ and $a = \frac{1.0}{49}$

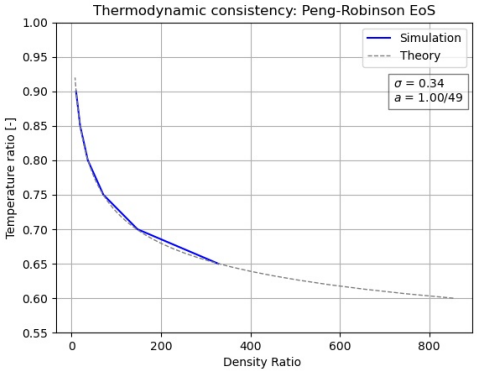



Figure C.4: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.34$ and $a = \frac{1.0}{49}$

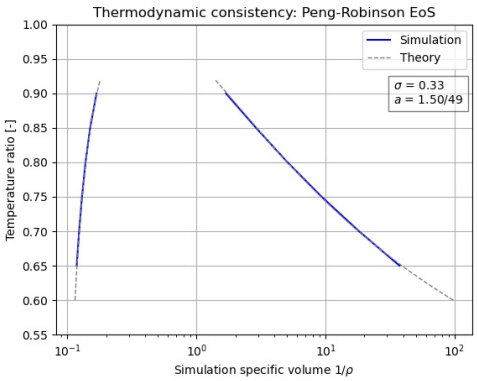



Figure C.5: P-V plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.33$ and $a = \frac{1.5}{49}$

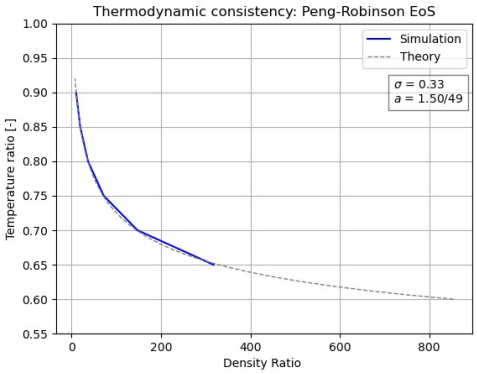



Figure C.6: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.33$ and $a = \frac{1.5}{49}$

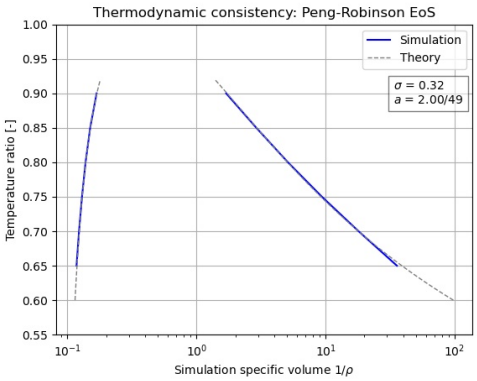



Figure C.7: P-V plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.32$ and $a = \frac{2.0}{49}$

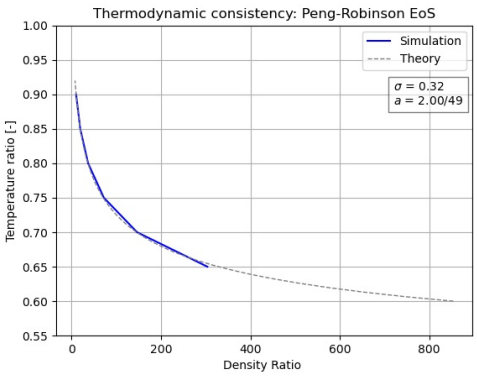



Figure C.8: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.32$ and $a = \frac{2.0}{49}$

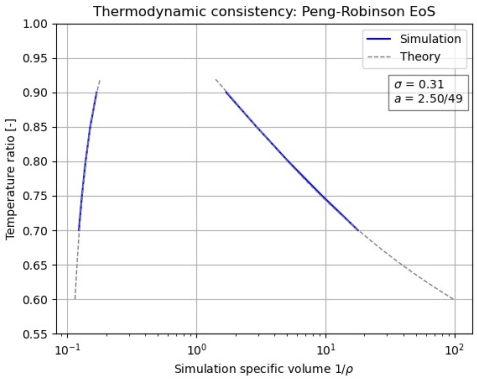



Figure C.9: P-V plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.31$ and $a = \frac{2.5}{49}$

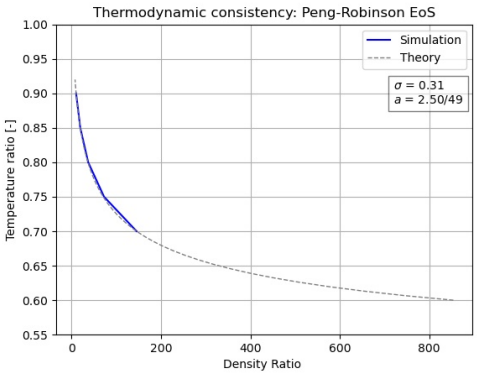



Figure C.10: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.31$ and $a = \frac{2.5}{49}$

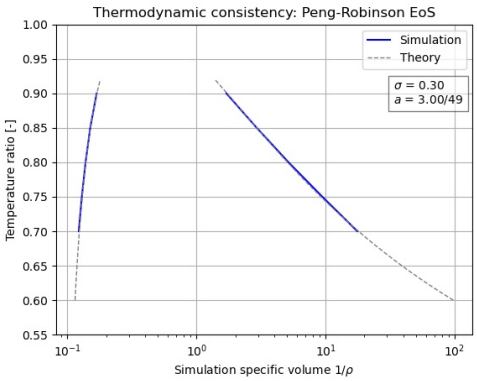



Figure C.11: P-V plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{3.0}{49}$

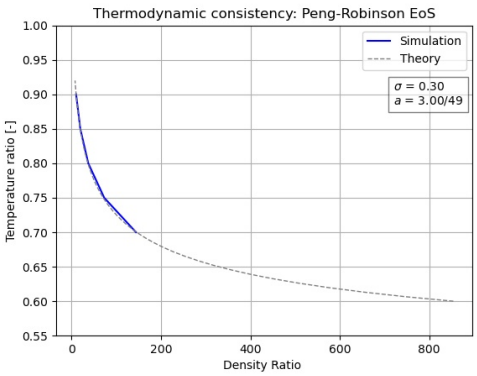



Figure C.12: Density ratio plot of results from 2D thermodynamic consistency using PR EoS with $\sigma = 0.30$ and $a = \frac{3.0}{49}$

# D Wall wettability contour density choice

To determine the wall wettability the contact angle needs to be calculated. Since the simulations include a diffuse interface, it is not always clear where one phase begins and the other ends. As such it is difficult to determine where contact at the wall occurs. Thus determining the contact angle becomes more difficult.

This appendix details the contact angle results of a simulation series of a 3D droplet using the Peng-Robinson equation of state ($a = 0.5/100$, $b = 2/21$, $R = 1$, $G = -1$) and a $\sigma = 0.31$ for the modified Guo forcing scheme, with a simulation domain of 150x150x75 nodes, with periodic xy boundary and no slip wall for z. The wall wettability is changed by giving the wall a fictitious density $\rho_w$.

The problem of the wall contact angle is shown in figs. D.1 to D.2. The contour densities are chosen at $\rho = 4$, since that is in the middle of the density range, and at $\rho = 2$ since that is the wall density. For the contact angle, the wetting length is needed, which is the length that the droplet makes contact with the wall. This is easy to determine for when the contour density is equal to the wall density, but difficult for the in the middle density.

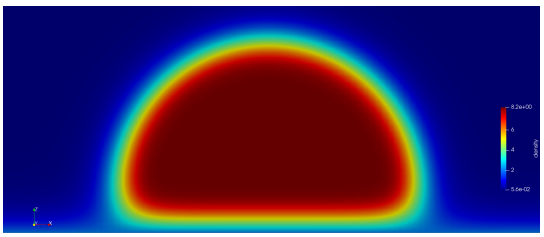


Figure D.1: Simulation result of wall wettability with $\rho_w = 2$

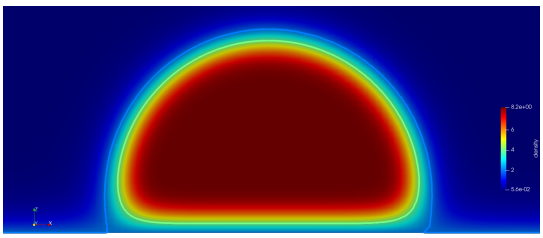


Figure D.2: Simulation result of wall wettability with $\rho_w = 2$, contours at $\rho = 2\&4$

As such there are two ways to measure the contact angle, shown visually in fig. D.3: Using a contour of the wall density, which is easier to measure but is inconsistent since the measuring density changes, or a contour in the interface middle, which is harder to measure but more consistent.

For the halfway the interface method, the wetting length is not well defined, since the contour for the wetting length is curved. As such it was decided that the resultant contact angle has to be tangent to the contour. Changing the wetting length slightly has little impact on the resultant angle, while having a huge impact whether the contact angle line is intersecting the contour twice, or not at all. This resulted in a more consistent measuring method.

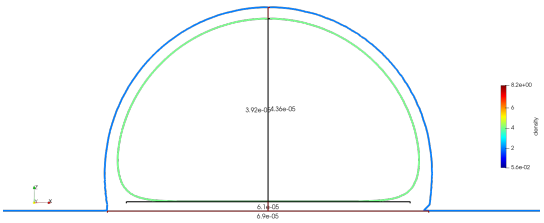


Figure D.3: Droplet height and wetting length measurements for $\rho_w = 2$ with contours at $\rho = 2\&4$

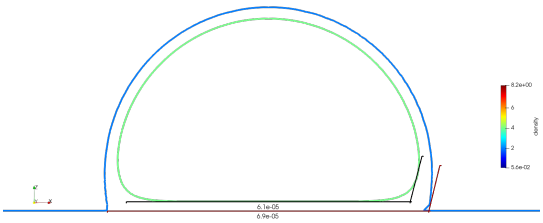


Figure D.4: Resultant contact angles for $\rho_w = 2$ with contours at $\rho = 2\&4$

A study is performed to determine the contact angle for different wall densities for both the wall density and halfway the interface methods. The resultant contact angle contours are shown in figs. D.5 to D.10, tabulated

in table D.1 and table D.2, and plotted in fig. D.11.

As can be seen in fig. D.11, the contact angle follows a linear trend and as such the exact contact angle can be fine tuned. The two methods agree very well with each other. As such, either method can be used to determine the contact angle. Since the wall density method is much simpler and faster, it is recommended to be used.

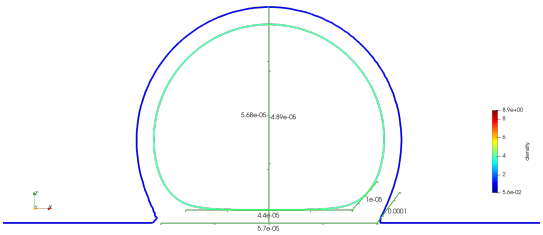


Figure D.5: Droplet height& wetting length measurements with resultant contact angle line for $\rho_w = 1$ with contours at $\rho = 1\&4$

Figure D.6: Droplet height& wetting length measurements with resultant contact angle line for $\rho_w = 2$ with contours at $\rho = 2\&4$

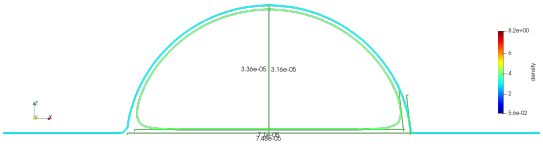


Figure D.7: Droplet height& wetting length measurements with resultant contact angle line for $\rho_w = 3$ with contours at $\rho = 3\&4$

Figure D.8: Droplet height& wetting length measurements with resultant contact angle line for $\rho_w = 4$ with a contour at $\rho = 4$

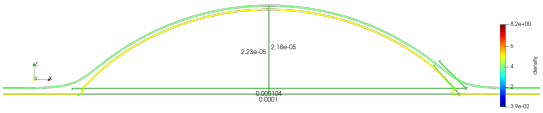


Figure D.9: Droplet height& wetting length measurements with resultant contact angle line for $\rho_w = 5$ with contours at $\rho = 4\&5$

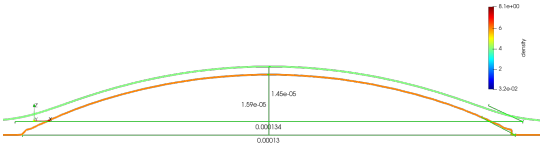


Figure D.10: Droplet height& wetting length measurements with resultant contact angle line for $\rho_w = 6$ with contours at $\rho = 4\&6$

Table D.1: Wall wettability using halfway the interface method

| Wall Density | Droplet Height [m] | Wetting Length [m] | Contact Angle [deg] |
|---|---|---|---|
| 1 | $4.884 \cdot 10^{-5}$ | $4.4 \cdot 10^{-5}$ | 131.5 |
| 2 | $3.92 \cdot 10^{-5}$ | $6.1 \cdot 10^{-5}$ | 104.2 |
| 3 | $3.15 \cdot 10^{-5}$ | $7.1 \cdot 10^{-5}$ | 83.2 |
| 4 | $2.478 \cdot 10^{-5}$ | $7.73 \cdot 10^{-5}$ | 65.3 |
| 5 | $2.175 \cdot 10^{-5}$ | $10.4 \cdot 10^{-5}$ | 45.4 |
| 6 | $1.45 \cdot 10^{-5}$ | $13.4 \cdot 10^{-5}$ | 24.4 |

Table D.2: Wall wettability using wall density method

| Wall Density | Droplet Height [m] | Wetting Length [m] | Contact Angle [deg] |
|---|---|---|---|
| 1 | $5.685 \cdot 10^{-5}$ | $5.7 \cdot 10^{-5}$ | 126.7 |
| 2 | $4.36 \cdot 10^{-5}$ | $6.9 \cdot 10^{-5}$ | 103.3 |
| 3 | $3.358 \cdot 10^{-5}$ | $7.48 \cdot 10^{-5}$ | 83.8 |
| 4 | $2.478 \cdot 10^{-5}$ | $7.73 \cdot 10^{-5}$ | 65.3 |
| 5 | $2.225 \cdot 10^{-5}$ | $10.05 \cdot 10^{-5}$ | 47.8 |
| 6 | $1.59 \cdot 10^{-5}$ | $13 \cdot 10^{-5}$ | 27.5 |

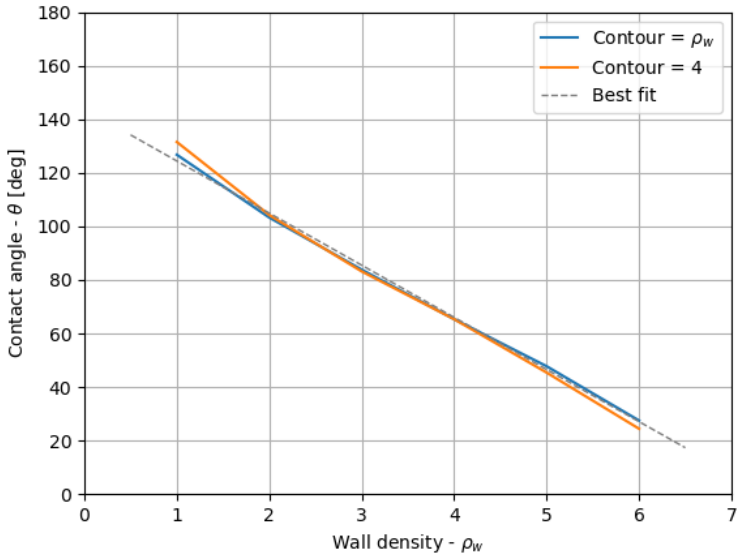



Figure D.11: Wall wettability plot comparing the halfway the interface and wall density method

# E OpenLB extension code

This appendix includes an explanation on how to extend OpenLB v1.3-1 with the source code used. A digital version of all files generated in this thesis is available at the git repository here: https://github.com/JuliusWeinmiller/ThesisLBM

An official cleaner version meant to be a drop in OpenLB extension is available here: https://github.com/JuliusWeinmiller/ThermalMultiphaseOpenLB

## E.1. Bounce Back Thesis

The `BounceBackThesis` class adds the ability to change the wall density and as such the wall wettability, shown in listing E.1. This derived class adds only the `setDensity` function and inherits everything else form the `BounceBack` class.

To allow the `setDensity` function to overwrite the fixed density, the access restriction fo the `BounceBack` class needs to be changed from `private` to `protected` as shown in listing E.3 and listing E.3.

Listing E.1: Addition of setDensity function to BounceBack class

```
template<typename T, typename DESCRIPTOR>
class BounceBackThesis : public BounceBack<T,DESCRIPTOR> {
    public:
    BounceBackThesis(T rho): BounceBack<T,DESCRIPTOR>(rho){};

    void setDensity( T newRho ){
        this->_rho = newRho;
    };

};
```

Listing E.2: Original BounceBack access restriction

```
template<typename T, typename DESCRIPTOR>
class BounceBack : public Dynamics<T,DESCRIPTOR> {
    public:
    [...]
    private:
    T _rho;
    bool _rhoFixed;
};
```

Listing E.3: Changed BounceBack access restriction

```
template<typename T, typename DESCRIPTOR>
class BounceBack : public Dynamics<T,DESCRIPTOR> {
    public:
    [...]
    protected:
    T _rho;
    bool _rhoFixed;
```

```
};
```

## E.2. Calculation of multiphase pressure

In OpenLB, the pressure is calculated assuming single phase flow, with this equation: $p = c_s^2 \rho$. However for the pseudopotential multiphase flow, the following pressure equation should be used: $p = c_s^2 \rho + 0.5 c_s^2 G \psi^2$. Thus pressure must be manually calculated on the super lattice. This is done using the `olb::SuperCalc` functionalities. The resulting code is shown in listing E.4

Listing E.4: "Field calculations for saving pressure"

```
olb::SuperCalcPower2D<BaseType, BaseType> psiSquared (pseudopotential, 2 );
olb::SuperCalcMultiplication2D<BaseType, BaseType> halfPsiSquared ( 0.5*G,
    ↪ psiSquared);
olb::SuperCalcPlus2D<BaseType, BaseType> addedDensity (density, halfPsiSquared);
BaseType cs2 = 1/olb::descriptors::invCs2<BaseType,NSDESCRIPTOR>();
olb::SuperCalcMultiplication2D<BaseType, BaseType> pressureNonDim ( cs2 ,
    ↪ addedDensity);
pressureNonDim.getName() = "Lattice pressure";
```

## E.3. Advection diffusion dynamics implementation

The `thesisAdvectionDiffusionBGKdynamics` implements the advection diffusion dynamics as given by Q.Li et al. (10.1103/PhysRevE.96.063303). The correction for the Tv is given in listing E.6 and for the source term in listing E.5. The source term phi needs to be calculated in the post processor.

Listing E.5: Source term implementation for AD dynamics collide function

```
    const T phi  = *cell.template getFieldPointer<descriptors::PHI_THERMAL>();
    const T phi0 = *cell.template getFieldPointer<olb::descriptors::PREV_PHI>();
    T derivPhi = phi - phi0;
    [...]
    for (int iPop=0; iPop < DESCRIPTOR::q; ++iPop) {
        T weight = descriptors::t<T,DESCRIPTOR>(iPop);
        cell[iPop] += weight * phi;
        cell[iPop] += weight * 0.5 * derivPhi;
        [...]
    }
    [...]
    cell.template defineField<olb::descriptors::PREV_PHI>(&phi);
```

Listing E.6: Tv error term correction factor for AD dynamics collide function

```
    const T* Tv0 = cell.template getFieldPointer<olb::descriptors::PREV_T_V>();
    const T* u   = cell.template getFieldPointer<olb::descriptors::VELOCITY>();
    T temperature = this->_momenta.computeRho( cell ) ;

    T Tv1[ DESCRIPTOR::d];
    T derivTv[ DESCRIPTOR::d];
    for (int iD=0; iD< DESCRIPTOR::d; ++iD){
        Tv1[iD] = temperature * u[iD];
        derivTv[iD] = Tv1[iD] - Tv0[iD];
    }
```

```
    [...]
    T cdTv;
    for (int iPop=0; iPop < DESCRIPTOR::q; ++iPop) {
        T weight = descriptors::t<T,DESCRIPTOR>(iPop);
        cdTv = 0.0;
        for (int iD=0; iD< DESCRIPTOR::d; ++iD){
            cdTv += descriptors::c<DESCRIPTOR>(iPop,iD) * derivTv[iD];
        }
        cell[iPop] += _omegaMod * weight * cdTv * descriptors::invCs2<T,DESCRIPTOR
            ↪ >();
        [...]
    }
    [...]
    cell.template defineField<olb::descriptors::PREV_T_V>(Tv1);
```

## E.4. Fluid properties

The fluid properties such as specific heat and thermal conductivity are different for vapour and liquid. Since the pseudopotential does not directly differentiate between the two phases, density is usually used as a marker to determine the phase. As such local fluid properties are dependent on the local density. A class named `FluidProperties` is written to make it easier to call for these properties. The declaration is given in listing E.7, and the definitions in listing E.8.

Since the properties scale linearly, the class was named `ScalingProperties` for a long time, but was renamed across the whole workspace. If some reference to `scalingProperties` remains, it should now be `fluidProperties`

In addition to the core functions, the print, write and verify functions were added for convenience. The verify function ensures that properties, such as `specificHeat()` indeed returns `specificHeatVapour`. The print and write functions are not shown, but they follow the same layout as the print and write function of OpenLB's `converter`.

Listing E.7: FluidProperties class declaration

```
template <typename T>
class FluidProperties{
    public:
    FluidProperties(T densityVapour, T densityLiquid,
    T specificHeatVapour, T specificHeatLiquid,
    T conductivityVapour, T conductivityLiquid,
    T tempLower, T tempHigher, T tempCritical);


    // SpecificHeat( T ) -> T
    // Returns the specific heat depending on the density
    T specificHeat(const T latticeDensity);

    // conductivity( T ) -> T
    // Returns the conductivity depending on the density
    T conductivity(const T latticeDensity);

    // density( T ) -> T
    // Returns the conductivity depending on the density
    T density(const T latticeDensity);

    // temperatureRatio( T ) -> T
```

```
    // Returns the temperature ratio depending on the lattice temperature
    // Should be fewer computations than using converter to convert from lattice to
        ↪ physical to ratio
    T temperatureRatio(const T latticeTemperature);

    T getDensityVapour();
    T getDensityLiquid();

    T getTempConvertionRatio();

    void verify();
    void print();
    void write(std::string const& title = "FluidProperties");

    protected:
    static T scaleProperty(const T localScale, const T propertyVapour, const T
        ↪ propertyLiquid);

    // ScaleDensity( T ) -> T
    // Returns 1 for liquid and 0 for vapour
    T scaleDensity(const T localDensity);

    private:
    const T _densityVapour;
    const T _densityLiquid;
    const T _specificHeatVapour;
    const T _specificHeatLiquid;
    const T _conductivityVapour;
    const T _conductivityLiquid;
    const T _tempLowerRatio;
    const T _tempRatioDifference;
    const T _tempCrit;
};
```

Listing E.8: FluidProperties class definition

```
template <typename T>
FluidProperties<T>::FluidProperties(
T densityVapour, T densityLiquid,
T specificHeatVapour, T specificHeatLiquid,
T conductivityVapour, T conductivityLiquid,
T tempLower, T tempHigher, T tempCrit ):
_densityVapour(densityVapour), _densityLiquid(densityLiquid),
_specificHeatVapour(specificHeatVapour), _specificHeatLiquid(specificHeatLiquid),
_conductivityVapour(conductivityVapour), _conductivityLiquid(conductivityLiquid),
_tempLowerRatio(tempLower/tempCrit), _tempRatioDifference((tempHigher-tempLower)/
    ↪ tempCrit),
_tempCrit(tempCrit)
{
 verify();
}

template <typename T>
T FluidProperties<T>::scaleDensity(const T localDensity){
```

```
 return (std::clamp(localDensity, _densityVapour, _densityLiquid) - _densityVapour)
     ↪ /(_densityLiquid - _densityVapour);
}

template <typename T>
T FluidProperties<T>::density(const T localDensity){
 return std::clamp(localDensity, _densityVapour, _densityLiquid);
}

template <typename T>
T FluidProperties<T>::specificHeat(const T localDensity){
 return scaleProperty(scaleDensity(localDensity), _specificHeatVapour,
     ↪ _specificHeatLiquid );
}

template <typename T>
T FluidProperties<T>::conductivity(const T localDensity){
 return scaleProperty(scaleDensity(localDensity), _conductivityVapour,
     ↪ _conductivityLiquid );
}

template <typename T>
T FluidProperties<T>::scaleProperty(const T localScale, const T propertyVapour,
    ↪ const T propertyLiquid){
 return propertyVapour*(1-localScale) + propertyLiquid*localScale;
}

template <typename T>
T FluidProperties<T>::temperatureRatio(const T latticeTemp){
 return _tempLowerRatio + (latticeTemp - 0.5)*(_tempRatioDifference);
}

template <typename T>
T FluidProperties<T>::getDensityVapour(){return _densityVapour;}

template <typename T>
T FluidProperties<T>::getDensityLiquid(){return _densityLiquid;}

template <typename T>
T FluidProperties<T>::getTempConvertionRatio(){return _tempRatioDifference;}


template <typename T>
void FluidProperties<T>::verify(){

 T rhoV = getDensityVapour();
 T rhoL = getDensityLiquid();
 T cvV = specificHeat(getDensityVapour());
 T cvL = specificHeat(getDensityLiquid());
 T kV = conductivity(getDensityVapour());
 T kL = conductivity(getDensityLiquid());

 assert( olb::util::nearZero(_densityVapour      - rhoV  ) );
 assert( olb::util::nearZero(_densityLiquid      - rhoL  ) );
 assert( olb::util::nearZero(_specificHeatVapour - cvV  ) );
 assert( olb::util::nearZero(_specificHeatLiquid - cvL  ) );
```

```
70  assert( olb::util::nearZero(_conductivityVapour - kV  ) );
    assert( olb::util::nearZero(_conductivityLiquid - kL  ) );

};
```

## E.5. Additional descriptor fields

For the new pseudopotential method, one additional descriptor field is required to store the scalar pseudopotential. The full implementation of the thermal simulation requires three descriptor fields. The first field is for the scalar source term, the second for the previous scalar source term to get the temporal derivative. The third for the 2/3D vector to store the previous thermal advection to use in the correction factor. These can be added to the `dynamics/descriptorField.h` or included via a separate file listing E.9.

Listing E.9: File to add the required descriptor fields

```
#ifndef DESCRIPTOR_FIELD_EXTENDED_H
2 #define DESCRIPTOR_FIELD_EXTENDED_H


#include "dynamics/descriptorField.h"

7 namespace olb {
    namespace descriptors {

        struct PSI_PSEUDO_RHO      : public DESCRIPTOR_FIELD_BASE<1,  0, 0> { };
        struct PHI_THERMAL         : public DESCRIPTOR_FIELD_BASE<1,  0, 0> { };
12      struct PREV_T_V            : public DESCRIPTOR_FIELD_BASE<0,  1, 0> { };
        struct PREV_PHI            : public DESCRIPTOR_FIELD_BASE<1,  0, 0> { };

    } // end descriptors namespace
} // end olb namespace
17

#endif
```

## E.6. Multichannel geometry setup

This section contains a copy of the geometry setup function used to set up a heating chamber with straight channels with arbitrary dimensions, shown in listing E.10. The geometry is split into four segments: the inlet, the divergent part, the channels and the convergent part. The divergent part connects the inlet to the channels. The convergent part forms the outlet. Care should be taken that the outlet velocity is still within the incompressible domain. For the channels, the number of them as well as the width, length and spacing between them can be varied.

Listing E.10: Geometry setup for arbitary straight multichannel layout

```
2 // Geometry setup
const int numberOfChannels = 4;
const BaseType channelLength = 8*physLength;
const BaseType channelWidth = 0.6*physLength;
const BaseType channelSpacing = 0.4*physLength;
7 const BaseType inletLength = 0.5*physLength;
const BaseType inletWidth = 1*physLength;
const BaseType distributionLength = 2*physLength;
const BaseType convergentLength = 3*physLength;
```

```
const BaseType convergentWidth = 2*physLength;

const BaseType cuboidLength = inletLength+channelLength+distributionLength+
    ↪ convergentLength + 2*conversionLength;
const BaseType cuboidWidth = numberOfChannels*channelWidth + (numberOfChannels-1)*
    ↪ channelSpacing + 2*conversionLength;

// End constants definition
////
const int materialBase = 0;
const int materialFluid = 1;
const int materialColdWall = 2;
const int materialHotWall = 3;
const int materialInlet = 4;
const int materialOutlet = 5;

void prepareGeometry( SuperGeometry2D<BaseType>& superGeometry )
{

    OstreamManager clout( std::cout,"prepareGeometry" );
    clout << "Prepare Geometry ..." << std::endl;

    // Set computational domain to wall
    superGeometry.rename( materialBase, materialColdWall );

    const BaseType widthToInlet = (cuboidWidth - inletWidth)/2.0 ;
    BaseType widthToOutlet = (cuboidWidth - convergentWidth)/2.0 ;
    Vector<BaseType,2> extend;
    Vector<BaseType,2> origin;
    BaseType hypoteneus;
    BaseType angle;
    const BaseType invSqrt2 = 1.0/sqrt(2.0);

    // Create Inlet -> rectangle
    extend = {inletLength+conversionLength, inletWidth};
    origin = {-1.5*conversionLength, widthToInlet};

    olb::IndicatorCuboid2D<BaseType> inletCuboid( extend, origin );
    superGeometry.rename( materialColdWall, materialFluid, inletCuboid );


    extend={1.5*conversionLength, inletWidth};
    olb::IndicatorCuboid2D<BaseType> inletBC( extend, origin );
    superGeometry.rename( materialFluid, materialInlet, materialFluid, inletBC );


    // Create Distribution -> trapezoid

    extend={distributionLength, cuboidWidth};
    origin={inletLength, 0};
    olb::IndicatorCuboid2D<BaseType> distCuboid( extend, origin );
    superGeometry.rename( materialColdWall, materialFluid, distCuboid );


    hypoteneus = std::sqrt( distributionLength*distributionLength + widthToInlet*
        ↪ widthToInlet ) + 10*conversionLength;
```

```
    angle = std::atan2(-(widthToInlet-conversionLength),distributionLength) - M_PI
        ↪ /4.0 ;

    origin = { inletLength + hypoteneus*invSqrt2*std::cos(angle), widthToInlet-
        ↪ conversionLength+hypoteneus*invSqrt2*std::sin(angle)  };
67  olb::IndicatorCuboid2D<BaseType> distCuboid1 ( hypoteneus, hypoteneus, origin,
        ↪ angle + M_PI/4.0);
    superGeometry.rename( materialFluid, materialColdWall, distCuboid1 );

    angle = -(std::atan2(-(widthToInlet-conversionLength),distributionLength) - M_PI
        ↪ /4.0) ;
    origin = { inletLength + hypoteneus*invSqrt2*std::cos(angle), widthToInlet+
        ↪ conversionLength+inletWidth+hypoteneus*invSqrt2*std::sin(angle)  };
72  olb::IndicatorCuboid2D<BaseType> distCuboid2 ( hypoteneus, hypoteneus, origin,
        ↪ angle - M_PI/4.0);
    superGeometry.rename( materialFluid, materialColdWall, distCuboid2 );

    // Create Channels -> rectangles

77  for( int iChannel=0; iChannel < numberOfChannels; iChannel++)
    {
        extend={channelLength, channelWidth};
        origin={inletLength+distributionLength, conversionLength + iChannel*(
            ↪ channelWidth+channelSpacing) };
        olb::IndicatorCuboid2D<BaseType> channelCuboid( extend, origin );
82      superGeometry.rename( materialColdWall, materialFluid, channelCuboid );
    }

    // Create Convergent -> trapezoid

87  extend={convergentLength + 2*conversionLength, cuboidWidth};
    origin={inletLength+distributionLength+channelLength, 0};
    olb::IndicatorCuboid2D<BaseType> convergCuboid( extend, origin );
    superGeometry.rename( materialColdWall, materialFluid, convergCuboid );

92

    BaseType lengthSegment = inletLength+distributionLength+channelLength;
    hypoteneus = std::sqrt( convergentLength*convergentLength + widthToOutlet*
        ↪ widthToOutlet ) + 2*conversionLength ;

    angle = std::atan2( (widthToOutlet),convergentLength) - M_PI/4.0 ;
97  origin = {lengthSegment + hypoteneus*invSqrt2*std::cos(angle), hypoteneus*
        ↪ invSqrt2*std::sin(angle) };
    olb::IndicatorCuboid2D<BaseType> convergCuboid1 ( hypoteneus, hypoteneus, origin
        ↪  , angle + M_PI/4.0  );
    superGeometry.rename( materialFluid, materialColdWall, convergCuboid1 );

    angle = -(std::atan2( (widthToOutlet),convergentLength) - M_PI/4.0);
102 origin = {lengthSegment + hypoteneus*invSqrt2*std::cos(angle), cuboidWidth +
        ↪ hypoteneus*invSqrt2*std::sin(angle) };
    olb::IndicatorCuboid2D<BaseType> convergCuboid2 ( hypoteneus, hypoteneus, origin
        ↪  , angle - M_PI/4.0 );
    superGeometry.rename( materialFluid, materialColdWall, convergCuboid2 );

    // Outlet

107
```

```
    origin = {cuboidLength-1.5*conversionLength, widthToOutlet+conversionLength};
    extend={2.5*conversionLength, convergentWidth-2*conversionLength};
    olb::IndicatorCuboid2D<BaseType> outletBC( extend, origin );
    superGeometry.rename( materialFluid, materialOutlet, materialFluid, outletBC );
112
    // Hot Wall

    extend={convergentLength + channelLength +conversionLength, cuboidWidth+
        ↪ conversionLength};
    origin={cuboidLength - extend[0], 0};
117 olb::IndicatorCuboid2D<BaseType> hotCuboid( extend, origin );
    superGeometry.rename( materialColdWall, materialHotWall, hotCuboid );

    // END

122 // Removes all not needed boundary voxels outside the surface
    superGeometry.clean();
    // // Removes all not needed boundary voxels inside the surface
    superGeometry.innerClean();
    superGeometry.checkForErrors();
127
    superGeometry.print();


    clout << "Prepare␣Geometry␣...␣OK" << std::endl;
132 }
```

# F Latent heat calculation using EoS python file

This chapter outlines the approach to numerically calculate the latent heat via the equation of state (EoS) using the python code base. The spatial conversion ratio between real and lattice units are: $\Delta x = 100e-6$, $\Delta t = \Delta x/210$. The remaining are derived from the equation of state.

The code to find the latent heat, with dimensional units of [J kg$^{-1}$], numerically is given in listing F.1. Since the density range changed, a new class is made based on the `LatticePengRobinson` from the verified `eos.py` file with the only difference being the name and density range used in the `getEquilibriumDensities` function. In the main execution part of the code, the integral is calculated using `scipy.integrate.quad`.

Listing F.1: Calculation of latent heat using realistic Peng Robinson

```
class RealPengRobinson(PengRobinsonInteractionPotential):
    def __init__(self, acentricity = 0.344, Tc = 647.4, Pc = 22.10e6, tr = 0.577, R
        ↪ =8314/18):
        self.tc = Tc
        self.R = R
        self.acentricity = acentricity
        self.pc = Pc
        self.a = 0.45724 * R**2 * Tc**2 /Pc
        self.b = 0.0778  * R    * Tc    /Pc

        self.c = (0.37464 + 1.54226*self.acentricity - 0.26992*self.acentricity*self
            ↪ .acentricity)

        self.t = self.tc*tr
        self._calcAlpha(tr)
        self.rhoc = self.pc/0.3074/self.R/self.tc


    [...]
    if __name__=="__main__":
    [...]
    tempRatio = 0.577
    PRreal = RealPengRobinson( )
    integralFunc = lambda rho: rho**-2 *PRreal.dPdT(rho, tempRatio)
    densityLiquidPRreal, densityVapourPRreal = PRreal.getEquilibriumDensities(
        ↪ tempRatio)[:2]
    integralResult = integrate.quad(integralFunc, densityVapourPRreal,
        ↪ densityLiquidPRreal)[0]

    print("Latent␣heat␣using␣PR␣[J/kg]:␣", tempRatio*647.4 * integralResult)

    >>> Latent heat using PR [J/kg]:  2335249.185632195
```

When non-dimensionalizing the PR EoS there are two ways to ensure that the correct latent heat can be recovered. The first way is to incorporate the reduced pressure, density and temperature factors to convert from one scale to the other as is shown in listing F.2.

Listing F.2: Calculation of latent heat using conversion factors

```
if __name__=="__main__":
[...]
convX = 100e-6
convT = convX/30/Nt
convU = convX/convT
convRho = PRreal.rhoc/PR.rhoc
convPressure = convRho*convU*convU

tempRatio = 0.577
PR = LatticePengRobinson( acentricity = 0.344, a=2/49., b=2./21. , R=1)
integralFunc = lambda rho: rho**-2 * PR.dPdT(rho, tempRatio)
densityLiquidPR, densityVapourPR = PRfix.getEquilibriumDensities(tempRatio)[:2]
integralResult = integrate.quad(integralFunc, densityVapourPR, densityLiquidPR)
    ↪ [0]

print("Latent heat EoS  scale [-]:       ",tempRatio *integral2 )
print("Latent heat Phys scale [J/kg]:    ",tempRatio*PR.tc *integral2 * PRreal.
    ↪ pc/PR.pc * PR.rhoc/PRreal.rhoc )
print("Latent heat lattice scale [-]:    ",tempRatio*PR.tc *integral2 * PRreal.
    ↪ pc/PR.pc /convPressure  )

>>> Latent heat EoS  scale [-]:       7.814324348667083
>>> Latent heat Phys scale [J/kg]:    2336692.925098959
>>> Latent heat lattice scale [-]:    52.98623412922809
```

The other way to recover the correct latent heat is to use the fluid specific non-dimensionalized value for R: $R_{dimless} = \frac{R}{M} * T_c \left(\frac{\Delta x}{\Delta t}\right)^{-2}$. With this value of R, the correct latent heat can be recovered from the equation of state, as shown in listing F.3, using the same conversion factors as in listing F.2 for `convU`.

Listing F.3: Calculation of latent heat using dimensionless R

```
if __name__=="__main__":
[...]
Rsp= 8314.0/18.0152
tempRatio = 0.577
TempCrit = 647.4
PRfix = LatticePengRobinson( acentricity = 0.344, a=2/49., b=2./21. , R=Rsp/(
    ↪ convU**2)*TempCrit )
integralFunc = lambda rho: rho**-2 * PRfix.dPdT(rho, tempRatio)
densityLiquidPRfix, densityVapourPRfix = PRfix.getEquilibriumDensities(tempRatio
    ↪ )[:2]
integralResult = integrate.quad(integralFunc, densityVapourPRfix,
    ↪ densityLiquidPRfix)[0]

print("Nondimensional latent heat using PR [-]:      ", tempRatio *
    ↪ integralResult)
print("Dimensionalized latent heat using PR [J/kg]:  ", tempRatio *
    ↪ integralResult * convU**2)

>>> Nondimensional latent heat using PR [-]:       52.94152795007024
>>> Dimensionalized latent heat using PR [J/kg]:   2334721.3825980974
```

As can be seen by the resultant recovered latent heats, the method of using a non-dimensional R is cleaner to implement and use than to use the conversion factors.