CHALMERS

Spinodal decomposition with the lattice Boltzmann method

A generalized multicomponent model with high order isotropy component interaction

Master of Science Thesis in the Master Degree Programme, Applied Physics

MIKAEL HÅKANSSON

Department of Mathematical Sciences
Division of Xyz
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2012

Report No. xxxx

Report No. xxxx:xx

Spinodal decomposition with the lattice Boltzmann method

A generalized multicomponent model with high order isotropy component interaction

Mikael Håkansson

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2012

Spinodal decomposition with the lattice Boltzmann method
A generalized multicomponent model with high order isotropy component interaction

Mikael Håkansson

Department of Mathematical Sciences
Chalmers University of Technology

Abstract

There is currently interest in the medical sciences in gaining deeper understanding regarding the dynamics arising from spinodal decomposition, especially in combination with other concurrent physical processes. One method of investigation is through numerical simulations. The lattice Boltzmann method (LBM) is a relatively new promising numerical method, showing great potential particularly in simulating multicomponent fluids. Many improvements have recently emerged in research literature, but few examples exist where these are collectively used and evaluated. In this work, recent developments are gathered and critically analyzed. Based on conclusions from said analysis, a numerical model, capable of solving any number of fluid components, is proposed with the purpose of reproducing the physical processes currently of interest - controlled spinodal decomposition in a solidifying ternary mixture. The proposed model is implemented in high performing multithreaded code with a careful choice of data structure and memory model. Also, a brief excursion in GPU (Graphics Processing Unit) computing is made, resulting in excellent computational performance with a factor of $\sim$35 speedup in the best case. The model is further validated through benchmarks, such as phase diagram, fluid-fluid interface profile and the Laplace law, to show good evidence of physical correctness. Simulations showed satisfactory results in reproducing the governing physical processes separately. A full simulation of the solidifying solution managed to reproduce complex secondary effects observed in the real case, but more work is needed to improve on the solidifying process and the numerical performance on the force responsible for the phase separation. The findings validate the LBM as an easily extendable and intuitive numerical method. With some minor improvements, the implemented application shows potential to be a competent tool in examining complex multicomponent fluids.

Keywords: Lattice Boltzmann Method, Multicomponent Flow, Spinodal Decomposition, GPU Computing

Contents

List of Figures

Part I
Background

1 Introduction

The Lattice Boltzmann Method (LBM) has emerged over the past two decades as a competent and promising alternative in computing many types of fluid flow. A fundamental difference in approach allows certain types of problems to be described very intuitively. Among these, the most prominent in literature are fluid flow through complex geometry, such as flow through porous media, and multiphase/multicomponent fluids. Both classes of problems which have been proven difficult to treat previously. In contrast to conventional and more established methods, the LBM does not solve for the macroscopic properties directly. Instead, the Boltzmann transport equation is solved, which describes the time evolution of the statistical distribution in phase-space of particles in a gas. The macroscopic properties are then acquired from moments of the distribution function. The Boltzmann equation can be formulated such that it will satisfy the Navier-Stokes equations of fluid motion, and accurately describes many of its accompanying complex behaviors. Owing to the nature of the method, where a mesoscopic particle-particle interaction approach is taken, certain aspects of fluid phenomena such as complex solid-fluid interaction and phase separation emerges naturally.

As the LBM has matured considerably in recent time, it has grown from a research topic to a viable option in engineering applications as well as an aid in many research areas. A field in which the LBM has seen much development of late is multiphase fluids, where focus lies in modeling thermodynamical phase separation or phase segregation by spinodal decomposition. These related phenomena are indeed strong points of this method, but there still seems to be no clear consensus of which approach to take. An issue not made easier by the vast number of suggested methods in research literature. However, many of the inconsistencies and drawbacks inherent in early models have been mended and improved upon.

Interest regarding multicomponent flows currently exist in the medical sciences. A specific case is the production of pill coatings, as in creating a suitable structure for optimal drug delivery. Here a porous structure is formed by letting two immiscible polymers separate through spinodal decomposition, while concurrently drying and solidifying.

Motivated by the novelty of the LBM and the wealth of potential applications, as well as by current interest in industry and research, this thesis will cover the fundamentals of the LBM, as well as provide detail and critical analysis of a few of the most popular models. Based on conclusions from said analysis, a model suitable for multicomponent flows with spinodal decomposition will be proposed. This model will be implemented, and an attempt in reproducing the process of the pill coating procedure will be made. Although there exists a few available open source LBM solvers, the implementation will be made from the ground up. While initially requiring more work, any improvements or modifications can readily be implemented. Moreover, due to the uncertainty in openness and extendability in existing software, it could be realized at a later stage that a crucial feature is difficult or impossible to implement. As a final point, since this thesis will cover recent research, it is unlikely that any published software reflects all functionality required by the proposed model.

In what follows in Part I, some background will be given as to introduce the reader to the main work in this thesis. The fields of fluid dynamics and kinetic theory are vast, therefore coverage shall be restricted to a few concepts relevant to this work. Explanations will be brief, but should hopefully capture the essence of the matter. Part II covers the Lattice Boltzmann method in general, as well as providing detail and analysis regarding recent progress in different models. Problem description and implementation details are described in Part III. Finally, simulation results, discussion and conclusions are presented in Part IV.

2 Fluid dynamics and the Navier Stokes equations

2.1 The continuum hypothesis

The mathematical treatment of fluids is part of the branch of continuum mechanics. As such, the fluid and its properties are treated as any mathematical quantity which possesses infinitely smooth, continuous properties. Fundamentally, however, as seen from a microscopic viewpoint, a fluid or a gas consists of a discrete number of particles whose collective interactions give rise to the macroscopic behavior we normally observe. The nature of the inherent dynamics at the microscopic scale is governed by collisions and thermal effects, giving rise to a brownian pattern of motion. As a result of this, the kinetic properties of individual particles, or small collections thereof, are fluctuating wildly over space and time. One is then tempted to pose the question as to in which regime continuum fluid mechanics is valid, as the continuous properties certainly breaks down at some point. This issue brings us to one of the fundamental underpinnings of fluid mechanics, namely the *continuum hypothesis*. Consider a small hypothetical volume where we observe and measure the averaged velocity and number density of all particles within it. For a volume where the scale in on the order of the particles themselves, the properties will fluctuate heavily for consecutive measurements, as previously mentioned. If the volume is made larger, these fluctuations will reduce in magnitude, and will eventually be so small as to be rendered negligible. Extending on this concept further; if the volume is made big enough, the measured properties will start changing with increased size. This time because part of the neighboring fluid, having different macroscopic properties, is included in the volume. The continuum hypothesis states that there exists a region in between these extremes where the control volume can be made smaller, but without a corresponding change in macroscopic properties, and without adding fluctuations. On these grounds, and within these limits, the continuous treatment is valid. Moreover, the region in which this holds true spans down to a very small spatial scale. For example, a volume of 10^{-9} cm^3 (cube with 10 μm side) containing water still holds on the order of 10^{13} molecules, which is well enough by a wide margin to smooth out any molecular fluctuations.

2.2 The equations of fluid motion

When describing the motion of a single viscous fluid in the macroscopic continuum regime, two equations are normally used. These are the *continuity equation* and the *Navier-Stokes equation*. For compressible flow these are

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

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \nabla \cdot \mathbf{T} + \boldsymbol{f} \tag{2.2}$$

respectively, where ρ is the density, $\boldsymbol{u}$ the fluid velocity, and p the pressure. Further, $\mathbf{T}$ is the deviatoric stress tensor, and f corresponds to body forces per unit volume. Body forces are of the nature that they affect the whole volume, with gravity being the most obvious example. The full stress tensor σ consists of an isotropic part (p) and a deviatoric part, or the non-equilibrium part, ($\mathbf{T}$) and is on the form

$$\sigma = -p\mathbf{I} + \mathbf{T} \tag{2.3}$$

where $\mathbf{I}$ is the identity matrix. The first two terms on the RHS in (2.2) can then be identified as the divergence of the stress tensor. For a fluid at rest the deviatoric part vanishes, hence the name non-equilibrium stress tensor. The deviatoric stress tensor determines how the fluid behaves in shearing motion, and is hence intimately connected to viscosity. It should be mentioned here that for any discrete fluid model, it is important to validate that the resulting stress tensor is symmetric. If this is not the case, anisotropic behavior and galilean invariance will be the result.

For incompressible flows the density is constant, and thus the term $\frac{\partial \rho}{\partial t}$ vanishes. The resulting equations of motion reduce to

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.4}$$

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

where μ is the dynamic viscosity. In further explaining the terms of the momentum equation, the first term on the LHS in (2.5) represents the acceleration resulting from a local change in momentum over time. The second term describes the acceleration from a change in the bulk flow, or more precisely from a change in the flow field with respect to position. This is known as convective derivative or convective acceleration. In the right hand side, the deviatoric stress tensor have taken on a form in which the symmetry properties only depends on the velocity.

2.3 Viscosity

A defining property of a fluid is its *viscosity*. This is a measure of the fluids resistance to change due to shear stress. A fluid with a high nominal value of viscosity is said to be viscous, and has a high resistance to change. That is, a given shear stress results in a relatively small change in motion in the fluid. One could say the the fluid responds with a strong counterforce to resist deformation. These are 'thick' fluids like oil and honey. Low viscosity fluids then obviously occupy the other end of the spectrum. A zero viscosity fluid is termed a superfluid, and can be thought of as the hydrodynamic equivalent of superconductors. Zero viscosity is an exotic condition displaying plenty of counterintuitive behavior, and can be achieved with liquid Helium below 2.17 degrees Kelvin.

Two types of viscosity are normally referred to, *dynamic* viscosity (μ) and *kinematic* viscosity (ν). The relation between the two is simply

$$\nu = \frac{\mu}{\rho} \tag{2.6}$$

The effect of the viscosity can be decomposed into two independent components, shear viscosity and bulk viscosity. The former is local in scope and describes fine grained motion, while the latter affects large scale velocity fluctuations.

2.4 Reynolds number

An important property with fluids is an intimate connection between scale and viscosity. Broadly speaking, a large system with a high viscosity fluid will exhibit the same type of motion as a small system with a low viscosity. This property of equivalence is a very useful tool, and is measured by the *Reynolds number*, a dimensionless quantity given by

$$Re \equiv \frac{\rho u L}{\mu} = \frac{uL}{\nu} \tag{2.7}$$

Here u is a typical velocity, and L is a typical length scale. Unless used in a well defined manner, this can be a somewhat arbitrary measure and perhaps best used as a crude guideline. Normally, Reynolds numbers associated with unsteady flow are at the order of $\sim 10^2$, while the onset of turbulence is somewhat higher at around $\sim 10^3$ and increase with degree of turbulence.

3 Spinodal decomposition

When mixing different liquids, a complex interaction between the different species of molecules takes place. Certain, interactions or forces strive to separate the components, while others work in the favor of mixing. Summing up all interactions, the resulting force will tell us whether the multicomponent liquid will mix or separate. In order to gain a basic understanding of the mixing or unmixing of liquids, one can employ the *regular solution model.* This is a mean-field-theory, and assumes a simplified view of species interaction on the basis of entropy of mixing. Here, all particle interactions and thermal energy are bundled together to form a dimensionless interaction parameter, χ, defined as

$$\chi = \frac{z}{2k_BT}(2\epsilon_{AB} - (\epsilon_{AA} + \epsilon_{BB})) \tag{3.1}$$

where ϵ denotes the interactions and self-interactions between species A and B, and z is the number of nearest neighbors. While hiding a lot of physical detail, the advantage with the interaction parameter approach is a clear picture of the mechanism of separation itself without having to deal with the underlying specifics. The aim with this method is to find an expression for the *free energy of mixing*, that is the difference between free energy before mixing and after mixing,

$$F_{mix} = F_{A+B} - (F_A + F_B), \tag{3.2}$$

where $F = U - TS$ is the Helmholtz free energy, and observe how changes in χ and initial volume fraction of A and B affect this energy. Since the system always strives to minimize the free energy, certain fundamental conclusions can be drawn.

One immediate observation is that as the interaction parameter χ is increased, a certain threshold value exists where it is no longer energetically favorable for the liquid to remain mixed for certain volume fractions. Moreover, the degree of separation increases with an increasing value of χ. Considering that $\chi \propto 1/T$, this is so far pretty intuitive. A drop in temperature increases χ, which in turn cause the phases to separate. The model reveals that for certain initial volume fractions, any change in fractional composition will decrease the free energy. This mode is globally unstable and a continuous separation will ensue, a process known as *spinodal decomposition.* Further, there are volume fractions where a small change in composition will yield an increase in free energy, thus implying stability. However, between the distinct cases of continuous separation and stability exists a region of local stability only. Here, a small change in composition leads to an increase in free energy, but thermal fluctuations are large enough to overcome the energy barrier for a phase separation. This is a *metastable* condition. The point where the metastable region and the unstable region meet is called the spinodal. By keeping track of where the spinodal is while changing χ, one can obtain a phase diagram as a function of χ and the volume fraction of one of the phases.

An interesting consequence of unmixing is that matter is transported from regions of low concentration to regions of high concentration. This is a complete reversal of the normal diffusion process. Hence, spinodal decomposition is sometimes referred to as 'uphill diffusion'. To explain this we must consider the chemical potential. When interaction forces exist, it is not the concentration that must be uniform at equilibrium, but the chemical potential. This quantity is related to the derivative of the free energy with respect to concentration, i.e. $\partial_\phi F$, thus matter will flow according to the derivative of the chemical potential, or the second derivative of the free energy, $\partial_\phi^2 F$. In the spinodal region, the second derivative is by definition negative, which implies that matter will flow in the opposite direction of 'normal' diffusion. In the stable region the sign is positive, and normal diffusion will take place.

4 Kinetic gas theory and the Boltzmann equation

Consider a large collection of particles in a confined space, such as a container filled with a gas. Individually these particles behave much like rigid bodies following newtonian dynamics, traveling in a straight line until colliding with another particle, followed by an exchange of momentum. In theory, one could propagate this system in time, particle by particle, by integrating Newton's laws of motion. Naturally this is going to be a very time consuming operation, and is unfeasible for any system larger than the microscopic scale. Consider instead the hypothetical volume described in section 2.1 (the continuum hypothesis), and make it small enough for molecular fluctuations to appear. If consecutive measurements of the velocity are taken for a gas or a fluid at rest, they will still vary. However, plot the distribution of a large number of measurements and a pattern will appear. For an ideal gas this will be the *Maxwell Boltzmann distribution.* This describes the distribution of particle speeds in an ideal gas close to equilibrium. Instead of working with particles individually, we can use the fact that they follow certain statistical rules. The treatment of particle distributions is in the realm of *statistical mechanics*, and the *Boltzmann equation* describes the time evolution of such a system. The Boltzmann equation on differential form is

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla_x f + \frac{1}{\rho} \boldsymbol{F} \cdot \nabla_\xi f = \Omega(f) \tag{4.1}$$

where $f = f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ is the particle probability distribution as a function of space, velocity, and time, $\boldsymbol{F}$ is the total body force and ρ the local particle density. The term $\Omega(f)$ on the right hand side is known as the *collision term*, and is yet to be defined. The essential meaning of this equation is very simple; the total change in the particle distribution function (LHS) should be equal to the result of the collision process. A shorter way of writing the Boltzmann equation which makes this obvious, is simply

$$\frac{d}{dt} f(\boldsymbol{x}, \boldsymbol{\xi}, t) = \Omega(f) \tag{4.2}$$

To show that this is the same as (4.1), the LHS is first written more explicitly as

$$\frac{d}{dt} f(\boldsymbol{x}, \boldsymbol{\xi}, t) = \frac{\partial f}{\partial t} + \frac{\partial \boldsymbol{x}}{\partial t} \cdot \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{\xi}}{\partial t} \cdot \frac{\partial f}{\partial \boldsymbol{\xi}} = \frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}} + \boldsymbol{a} \cdot \frac{\partial f}{\partial \boldsymbol{\xi}} \tag{4.3}$$

Since $\frac{\partial}{\partial \boldsymbol{x}} \equiv \nabla_x$ and analogously for ξ, and with $\boldsymbol{a} = \boldsymbol{F}/\rho$ according to Newton's second law on differential form, we arrive at (4.1).

The types of processes that are suitable to describe in this representation reside in the *mesoscopic* domain. That is, sufficiently large to render the treatment of individual particles unfeasible and irrelevant, but small enough that we wish to keep the particle viewpoint, i.e. where the collective behavior of particles is important. Depending on the nature of the system to be described, the collision term is modeled accordingly. To capture the dynamics of fluid flow, a simple and common choice of collision term is the one suggested by Bhatnagar-Gross-Krook,

$$\Omega(f) = -\frac{1}{\tau}(f - f^{eq}) \tag{4.4}$$

where τ is the *relaxation time* of the system, i.e. the mean time between particle collisions, and f^{eq} is the Maxwell Boltzmann velocity distribution

$$f^{eq} = \left(\frac{\rho}{2\pi k_B T}\right)^{3/2} \exp\left[-\frac{\rho}{2k_B T}(\boldsymbol{\xi} \cdot \boldsymbol{\xi})\right] \tag{4.5}$$

The interpretation of the BGK collision term is that the system relaxes towards the equilibrium function, which is the distribution of particle velocities in an ideal gas. And it does so at a certain rate, determined by τ. However, it should be mentioned that this model is unphysical, but when used with the lattice Boltzmann approximation it recovers the macroscopic properties of the Navier-Stokes equation in the limit of small τ.

As with any reasonably complex physical scenario, the governing equations are at best very difficult to solve, and closed form solutions are usually restricted to simple ideal cases. This is where numerical methods come into play, where the governing equations are discretized and solved up to a certain order of accuracy. With these words, it comes naturally to introduce the lattice Boltzmann method.

Part II

The Lattice Boltzmann Method

5 Overview of the lattice Boltzmann method

When the lattice Boltzmann method (LBM) started to gain attention about two decades ago, it showed promising capabilities in modeling many types of flow, which had previously been more difficult to achieve. Since its emergence, the LBM has evolved into a viable alternative to existing numerical methods, as many of its initial drawbacks and limitations have been overcome or improved upon. The type of applications that have been most pronounced are flows through complex geometry and multiphase/multicomponent liquids, since these in particular have been very difficult to implement and control with traditional methods. However, due to the particular nature of the LBM, these types of problems and their inherent physics translate very well into its framework. To better understand the reason behind this, it is illuminating to take a brief look at the history and origins of the lattice Boltzmann method.

What sets the LBM apart from most other numerical schemes is its foundation in kinetic theory. Rather than solving a set of continuum macroscopic equations, a mesoscopic particle-based approach is taken. Indeed, the LBM originates from lattice gas automaton (LGA) models. In these types of models, a very simplified view of particle-particle interaction is adopted. Here both space and velocity is discretized onto a grid, which in turn is populated with 'virtual' particles. These are then transported around the grid along the links between the grid points, colliding with other particles according to very simple rules. To further simplify things, a link can either have one particle or no particle at all, indicating the boolean nature of the model. For a system with N links between connecting grid points, the evolution equation for a typical LGA model is written on the form

$$n_i(x + e_i, t + 1) - n_i(x,t) = \Omega_i(n(x,t)), \qquad i = 0,1,...,N \tag{5.1}$$

where n_i is a boolean variable indicating the presence or absence of a particle at link i, and e_i are the spatial directions of each link. When two particles are transported towards a common grid point, their subsequent change in momenta is determined by the *collision term* Ω. The following step is advecting, or *streaming*, the particles along their new direction of motion. The idea of this picture, despite its remarkable simplicity, is that the general physical behavior of the underlying particles will emerge on the large scale, and in particular that the governing macroscopic equations will be satisfied in the limit of an infinitely small lattice spacing. In 1986, Frisch *et al.* [9] managed to successfully recover the Navier-Stokes equations using a symmetric hexagonal lattice. One major drawback of this model is that suffers from statistical noise. To overcome this one has to choose a lattice spacing so small that computations are no longer feasible, or to use some sort of statistically averaged pre-smoothing technique. It was the latter which led to the development of the LBM, as it was soon discovered (McNamara & Zanetti, 1998) [29] that this could be used exclusively on its own, and thus the connection to LGA could be dropped entirely.

The main idea of the LBM, as seen in connection with the LGA, is then to replace the binary particle populations $n_i \in \{0,1\}$ with smoothly varying distribution functions $f_i \in \mathbb{R}_+$, which can be seen as ensemble averages of the former. As a result the macroscopic properties, retrieved from the underlying particle kinetics, will be free of statistical noise. Moreover, to address the collision operator, which previously only amounted to boolean logic but now subject to more rigorous statistical kinematic treatment, Higuera and Jiménez [17] assumed that the particle distributions on any given lattice site is close to equilibrium state, and can thus be linearized. Further, they proposed a specific form of the collision term, still among the most widely used today, which uses a single relaxation time to relax all populations, or distribution functions, towards their equilibrium state. This collision term is known as the BGK collision term, after Bhatnagar, Gross and Krook (1958) [3]. The equilibrium distribution is then chosen such that the Navier-Stokes equations are recovered in the continuum limit.

In fact, one can arrive at the lattice Boltzmann equation either by starting from the LGA model, or entirely from the continuum Boltzmann equation with a discrete set of velocities [5], using an expansion around a small Mach number.

Mesoscopic viewpoint

With this in mind, one can describe the scale at which the lattice Boltzmann method operates as being larger than the microscopic domain of molecular dynamics, while still smaller than the macroscopic scale since it deals with particle kinetics rather than continuum dynamics. This is also known as the mesoscopic scale. Owing to the nature of this particle viewpoint, many complex fluid phenomena emerges naturally by the implementation of intuitive kinetic mechanisms. Solid-fluid boundary conditions can be treated as rigid particles bouncing off a flat surface, making the interaction with complex geometries a trivial matter. An issue normally much more complicated with continuum methods. Multicomponent fluids can be simulated by inserting additional sets of particle distribution functions, and their miscibility controlled by nearest neighbor mean-field interaction, spontaneously giving rise to phase-separation and interfacial dynamics. Though it has been shown, as the model have gradually matured, that often great care must be taken to keep the results physically sound and valid. Without rigorous analysis of the kinetics involved, it is easy to end up with a numerically unstable and unphysical model.

Other uses

Although the most reported use of LBM seems to be with flows through porous media and multiphase flows, it can be applied to many other areas in fluid dynamics and other regimes altogether. Providing a few examples of this; Yu & Girimaji performed a study of turbulent jet flow with high Reynolds number (Re $\sim$180,000) [48]. Mendoza & Muñoz successfully recovered the Maxwell equations by using auxiliary population vectors and a modified equilibrium function [31]. The lattice Boltzmann method can also be used in quantum mechanics by modifying it to solve the time dependent Schrödinger equation, as reviewed by Succi & Benzi *et al.* [43]. Recently progress has also been made by Lapitski & Dellar in solving the Dirac equation (to first order) [25]. Pattison *et al.* modeled magnetohydrodynamic flow to good agreement with benchmarks [33]. Plenty of other examples exist in literature for the curious minded.

On simulation

What makes the LBM such an attractive option in many cases is not only its great algorithmical simplicity, but stemming from this is also easy implementation and excellent computational efficiency. The advancement of the simulation in time consists basically of a collision/relaxation step and an advection/streaming step. The collision step is completely local in space, lending itself extremely well to fine grained computational parallelism. The streaming step is realized as a simple integer shift of variables in memory. While this step is not completely local, it only extends to neighboring nodes. Moreover, the entire simulation grid can easily be decomposed and distributed over a large cluster, with little data communication between each advancing step. Indeed, it has been shown in many instances [1, 8, 34] that near linear speedup can be achieved over large supercomputer clusters. Another interesting computational route is that of implementation on modern graphics cards, also known as GPU compute devices. These chips contain a large number of computational cores compared to a normal CPU, and can perform very well in cases where fine grained data parallelism can be achieved [38]. Certain problem types display performance gains up to two orders of magnitude compared to a single CPU implementation. In the case of LBM, varying speedups of factors ~5-20 have been reported for single component flow [12, 37, 46].

Limitations

While there are many attractive properties due to the intrinsics of the LBM, it is important to mention its limitations and drawbacks, many of which arise from the very same principles which make it easy to use. One of the most obvious limitations is that of low Mach number flows. In a basic implementation, stable and accurate simulations are obtained only when $\mathrm{Ma} \lesssim 0.3$. There is also a relatively low limit to attainable Reynolds numbers ($500 \lesssim \mathrm{Re} \lesssim 2000$) with the BGK collision term in particular, compared to advanced discrete macroscopic methods. Although, by special treatment of the collision term [7] and viscosity manipulation [18], one can achieve much higher Reynolds numbers. The simplest implementation of the solid-fluid boundary condition (bounce-back) is viscosity dependent with the BGK collision term, and only of first order accuracy and thus have a degrading effect on the simulation. This can be ameliorated by different choice of collision term and a more elaborate solid-fluid treatment. Fundamentally the LBM is athermal, but temperature can be embedded as an extra set of distribution functions, following a modified collision term. When modeling multiphase flows, it can be difficult to choose the right forcing model and approach in underlying phase separation mechanism. The outcome is often an inconsistent model with incorrect interfacial dynamics and spurious velocity currents, even though results might appear correct at first glance.

It is promising that many of the shortcomings can be greatly improved simply with a modified collision term. However, in order to acquire a physically correct model for a particular case, advanced theoretical analysis is often required, which could be potentially daunting for non-experts. Further, the outcome of this are modifications that can be difficult to grasp, possibly obfuscating of the physics at hand, and in the worst case takes us far from the intuitive clear picture of the model that made it so attractive to begin with. With that said, thanks to the contributions from the scientific community, many of the shortcomings have been addressed with reported solutions being readily implementable, and new improvements are constantly being published.

5.1 Lattice structure and methodology

To provide orientation and overview, this section will describe the main components and terminology of the LBM, revealing all the necessary detail involved in a basic implementation. For simplicity, in the following description a 2-dimensional lattice will be used, together with the BGK collision term.

The fundamental building blocks are the lattice structure along with its discrete set of lattice vectors, the evolution equation, and the hydrodynamic relations to recover the macroscopic quantities from the particle distribution functions. However, solid-fluid interactions, initial conditions and boundary conditions will also be brought to attention. Starting with the space and velocity discretization, the system of lattice and lattice vectors follow a naming convention according to DdQq, where d defines the spatial dimension, and q the number of discrete velocities. Here a 2D system with 9 unique velocities will be treated, that is, the D2Q9 model. In this case, space is discretized in a cartesian grid, and the lattice vectors link the current cell with all the 8 nearest neighbors. Remaining is the zero vector, which allows for a stationary distribution. Note that the vectors in this model do not have equal length, and this is indeed not a requirement. Instead, different weights are associated with the different vectors to maintain isotropy and symmetry. The

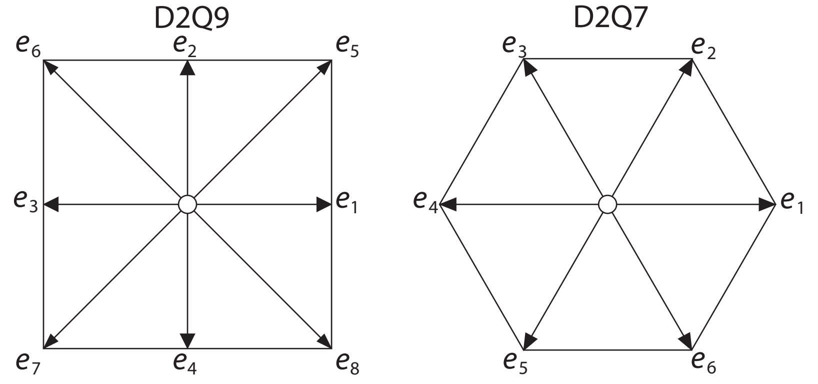

Figure 1: Lattice structures for the D2Q9 and D2Q7 models. Apart from the shown velocities, there is also a zero vector $\boldsymbol{e}_0$, representing a stationary distribution.

discretization does not need to be cartesian, a common model is D2Q7, where a hexagonal lattice is used, with 6 nearest neighbor vectors of equal length and one stationary vector. Figure 1 shows the lattice cells for both the D2Q9 and D2Q7 models. The lattice vectors $\boldsymbol{e}_i$ for the D2Q9 model are

$$(\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_8) = \begin{pmatrix} 0, & 1, & 0, & -1, & 0, & 1, & -1, & -1, & 1 \\ 0, & 0, & 1, & 0, & -1, & 1, & 1, & -1, & -1 \end{pmatrix} \tag{5.2}$$

Space is discretized in suitable number of lattice cells N_x and N_y in each dimension. Obviously, this is done in a way consistent with the chosen velocity space discretization. Each lattice cell stores Q distribution functions, one for each lattice vector, giving a total number of $N_x \cdot N_y \cdot 9$ values representing the entire fluid.

With all the necessary background given, it is now suitable to present the evolution equation, which naturally has the same appearance as (5.1), but with the BGK collision term in place, and the binary particle function $n(\boldsymbol{x},t)$ replaced with the continuous distribution function $f(\boldsymbol{x},t)$:

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i \Delta t, t + \Delta t) - f_i(\boldsymbol{x},t) = -\frac{1}{\tau}[f_i(\boldsymbol{x},t) - f_i^{eq}(\rho,\boldsymbol{u})], \qquad i = 0,1,...,N \tag{5.3}$$

where τ is the relaxation rate. The right hand side describes the process of collision and relaxing f towards its equilibrium state, f^{eq}. The equilibrium function, in turn, depends only on the local density and velocity. These macroscopic quantities are given by the hydrodynamic moments of f

$$\rho = \sum_i f_i, \qquad \rho\boldsymbol{u} = \sum_i f_i \boldsymbol{e}_i \tag{5.4}$$

The equilibrium function is derived from a Taylor expansion of the Maxwell-Boltzmann distribution function, and is given in lattice units (to second order) by

$$f_i^{eq} = w_i \rho \left(1 + 3(\boldsymbol{e}_i \cdot \boldsymbol{u}) + \frac{9}{2}(\boldsymbol{e}_i \cdot \boldsymbol{u})^2 - \frac{3}{2}\boldsymbol{u}^2 \right) \tag{5.5}$$

where w_i are a set of weights obeying $\sum w_i = 1$. These are related to the choice of lattice vectors, and for the D2Q9 model amount to

$$w = \frac{1}{36}(16, 4, 4, 4, 4, 1, 1, 1, 1) \tag{5.6}$$

The algorithm for the evolution of one time step is detailed below (also see Figure 2). For each lattice cell, do the following

- Calculate macroscopic properties with relations (5.4)
- Evaluate equilibrium functions according to (5.5), using ρ and $\boldsymbol{u}$ obtained from previous step.
- Update distribution functions:

$$\tilde{f}_i(\boldsymbol{x},t) = f_i(\boldsymbol{x},t) - \frac{1}{\tau}[f_i(\boldsymbol{x},t) - f_i^{eq}(\rho,\boldsymbol{u})]$$

- Stream populations from current node to neighboring nodes:

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i \Delta t, t + \Delta t) = \tilde{f}_i(\boldsymbol{x},t)$$

In the last step, because of data dependencies, a temporary storage buffer is needed for the destination sites of the streaming populations. This implies that twice the amount of memory is required, although as we shall see later, there are other ways of accomplishing this with a much reduced memory footprint. In our discrete system, the lattice spacing and time step are referred to as Δx and Δt, respectively. Normally however, one uses normalized lattice units, meaning $\Delta x = \Delta t = 1$. In

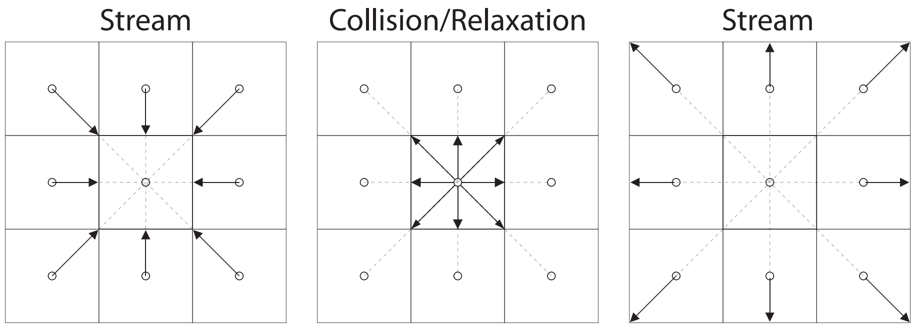

Figure 2: Overview of the stream/collide procedure, focused on one lattice cell and a set of populations/distribution functions. In the step leading up to the collision, particle populations are streaming into the current node. These populations are used to compute the equilibrium functions and subsequent relaxation. Next, these populations are streaming out along their respective direction and the process is repeated. (Note that no cell is ever 'empty'. Other populations have simply been omitted for clarity.)

these units, the sound speed becomes $c_s = 1/\sqrt{3}$, and in deriving the Navier-Stokes equations from the LBE, one can identify the kinematic fluid viscosity [26] as

$$\nu = c_s^2 \Delta t \left(\tau - \frac{1}{2}\right) = \frac{1}{3}\left(\tau - \frac{1}{2}\right). \tag{5.7}$$

Moreover, the pressure is simply given by an equation of state (EOS), such that $p = c_s^2 \rho = \rho/3$. One can reintroduce physical quantities in the system by setting for example either Δx or Δt, and other free parameters, such as the relaxation time, to fixed values, then derive the quantities from the physical relations.

An important issue in any fluid solver is that of solid-fluid boundary treatment. Fortunately solid obstacles are very easily introduced into the LBM, indeed this is one of the models greatest strengths. In its most basic implementation, one simply defines lattice nodes as being solid or fluid. At the wall boundary we want to achieve the no-slip condition, meaning the macroscopic fluid velocity is zero at this interface. The LBM solution is to have the particle populations streaming towards the wall bounce off of it, in such a way that the streaming direction is completely *reversed* (not reflected). To clarify; if the population was traveling in the direction $\boldsymbol{e}_5$ before hitting the boundary, it would travel along $\boldsymbol{e_7} = -\boldsymbol{e_5}$ after the interaction. This is called the *bounce-back* boundary condition, and the rule is formally written as

$$f_{\bar{i}}(\boldsymbol{x},t+\Delta t) = f_i(\boldsymbol{x},t), \qquad \boldsymbol{e}_{\bar{i}} = -\boldsymbol{e}_i \tag{5.8}$$

Although easy to implement, it has been showed that this treatment is only accurate to first order, and should be avoided for cases with much solid boundary exposure, such as flows through porous media. There exist plenty of other more accurate options [32], however they are usually not as straight forward in implementation.

Other types of boundary conditions to mention are those for the grid boundaries, for example inlet or outlet flows. Defining these is equivalent to specifying Dirichlet and Neumann type boundary conditions in continuum mechanics. Common in LBM is to use the so called Zou-He boundary conditions [52], where either a given velocity profile or pressure/density can be specified across the domain border. This involves using the moment representations (5.4) and some additional assumption (such as zero tangential velocity) to solve a set of equations. Solving these gives the values of the distribution functions at the boundary, such that the prescribed macroscopic conditions are satisfied.

Finally, some detail on initial conditions should be provided. If the velocity field is prescribed for the entire grid, one can use the equilibrium function (5.5), together with desired density ρ, to compute the distribution functions $f = f^{eq}$ for the corresponding nodes. However, this assumes that all kinetic moments are at equilibrium, and is consequently only valid for the restricted case of a stationary flow with zero velocity and pressure gradients. If the flow problem is of the nature that the initial transient is of minor importance, this approach can likely still be used. For cases where it is essential to have an accurate initial condition, and where the initial pressure is not given, other measures have to be taken. One option suggested by Mei, Luo, *et al* [30]. is an iterative approach to find the initial distribution functions, constructed in such a way as to be easily adopted into the LBM framework. This generates an initial flow field with an error of order $O(u^3)$ in the pressure field.

5.2 Deriving the lattice Boltzmann equation

One can derive the lattice Boltzmann equation in a few different ways, for example, either by a Chapman-Enskog multiscale expansion [5] or by asymptotic analysis [21]. Here, following the derivation in Chen & Doolen [5], the starting point will be the continuum Boltzmann equation, as first done by He & Luo [14], with a prescribed equilibrium function and using the BGK collision term. The equation describing the evolution of the distribution function $g = g(\mathbf{x},\boldsymbol{\xi},t)$ is then written as

$$\frac{\partial g}{\partial t} + \boldsymbol{\xi} \cdot \nabla_x g = -\frac{1}{\varepsilon\tau}(g - g^{eq}) \tag{5.9}$$

where $\boldsymbol{\xi}$ denotes particle velocity space, τ is the relaxation parameter, ε is a small parameter on the same order as the time scale, and g^{eq} is the equilibrium function, here described by the Maxwell-Boltzmann distribution function:

$$g^{eq} = \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{(\boldsymbol{\xi} - \boldsymbol{u})^2}{2RT}\right) \tag{5.10}$$

where D is the spatial dimension of the system. The macroscopic quantities for the kinetic system are acquired by taking moments of the distribution functions. The zeroth, first and second order moments yields the density, momentum (mass flux) and energy, respectively, and are given by

$$\rho = \int g d\boldsymbol{\xi}, \qquad \boldsymbol{j} = \rho\boldsymbol{u} = \int g\boldsymbol{\xi} d\boldsymbol{\xi}, \qquad \rho DT/2 = \int g(\boldsymbol{\xi} - \boldsymbol{u})^2 d\boldsymbol{\xi} \tag{5.11}$$

Using the ideal gas law, the sound speed is $c_s = \sqrt{RT}$. Normalizing the equilibrium function and the velocity by $\sqrt{3RT}$, yielding instead a sound speed of $c_s = 1/\sqrt{3}$, and assuming the fluid velocity $\boldsymbol{u}$ is small compared to the sound speed, we can Taylor expand around this parameter and arrive at

$$g^{eq} = \frac{\rho}{(2\pi/3)^{D/2}} \exp\left(-\frac{\boldsymbol{\xi}^2}{2c_s^2}\right)\left(1 + \frac{\boldsymbol{\xi} \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{\xi} \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u}^2}{2c_s^2} + O(\boldsymbol{u}^3)\right) \tag{5.12}$$

At this point it is suitable to make the transition to the discrete velocity space, where the continuous variable $\boldsymbol{\xi}$ is replaced by the discrete set $\boldsymbol{e}_i \subset \boldsymbol{\xi}$, where $i = \{1,2,...,N\}$ denotes the number of lattice vectors (sometimes referred to as links). It is also convenient, and customary in the context of LBM, to write the corresponding distribution functions as $g_i(\boldsymbol{x},t) \equiv g(\boldsymbol{x},\boldsymbol{e}_i,t)$. This means that for equation (5.9), the equation only have to be solved for this discrete subset, rather than for all of velocity space. This is a huge simplification, but nevertheless, as we shall see, this model still maintains sufficient kinetic detail to describe the physics of interest at the macroscopic level. To obtain the macroscopic variables from the distribution function in discrete space, one can use Gaussian quadrature to approximate the equations in (5.11). Gaussian quadrature in its general form is written as

$$\int_S f(x)dx \approx \sum_i w_i f(x_i), \qquad x_i \in S \tag{5.13}$$

where x_i are a set of points inside integration space, and w_i their corresponding weights. It should be pointed out that, if f is a polynomial, for a certain choice of x_i and w_i, the integral $\int f(x)dx$ can be evaluated *exactly* up to a certain limited polynomial order. The density and momentum can be approximated as

$$\rho(\boldsymbol{x},t) = \sum_i W_i g_i(\boldsymbol{x},t), \qquad \rho\boldsymbol{u}(\boldsymbol{x},t) = \sum_i W_i g_i(\boldsymbol{x},t)\boldsymbol{e}_i \tag{5.14}$$

where W_i are weights, independent of $\boldsymbol{x}$ and t, yet to be determined. Considering the pre-factor in equation (5.12), it is here convenient to perform a simple transformation of the distribution function

$$f_i(\boldsymbol{x},t) \equiv W_i g_i(\boldsymbol{x},t) \tag{5.15}$$

Rewriting (5.14) and (5.12) then turns into the simplified forms

$$\rho(\boldsymbol{x},t) = \sum_i f_i(\boldsymbol{x},t), \qquad \rho\boldsymbol{u}(\boldsymbol{x},t) = \sum_i f_i(\boldsymbol{x},t)\boldsymbol{e}_i \tag{5.16}$$

and

$$f_i^{eq} \equiv W_i g^{eq} = w_i \rho \left(1 + \frac{\boldsymbol{\xi}\cdot\boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{\xi}\cdot\boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u}^2}{2c_s^2}\right) \tag{5.17}$$

respectively, where $w_i = W_i/(2\pi/3)^{D/2}$. He & Luo successfully retrieved the weights by approximating the integrals (5.11) with third-order Hermite polynomials, then proceeded by solving these exactly with Gaussian quadrature [14]. In this case, the specific form of the quadrature used was

$$\int f(x)\mathrm{e}^{-x^2/2} = \sum_{i=1}^{n} W_i f(x_i) \tag{5.18}$$

which is exact when $f(x)$ is a polynomial of order k, and $0 \leq k \leq 2n-1$ holds. Since the mass and velocity are conserved moments, it is indeed vital that the relations of the hydrodynamic moments in (5.14) hold exactly. The resulting expression allows the weights W_i to be identified, when comparing with (5.14). One can also use conservation properties and symmetry of the lattice for a general approach. The sound speed in lattice units is given by

$$\sum_i w_i e_{i,\alpha} e_{i,\beta} = \delta_{\alpha\beta} c_s^2, \quad \alpha, \beta = 1,...,D \tag{5.19}$$

where α, β denote the cartesian components, and D the spatial dimension. Furthermore, conservation laws imply that

$$\sum_i w_i = 1, \qquad \sum_i w_i \boldsymbol{e}_i = 0 \tag{5.20}$$

Using relations (5.19) and (5.20), one can retrieve precisely the weights w_i as with the Hermite formula approach. As a final step, the evolution equation (5.9) with the transformation (5.15) is discretized to *first* order. However, it has been shown [42] that because of the nature of the resulting discretization error in the evolution equation, the result is actually *second* order accurate in space and first order in time. The time derivative term then becomes

$$\frac{\partial f_i}{\partial t} \approx \frac{f_i(\boldsymbol{x},t+\Delta t) - f_i(\boldsymbol{x},t)}{\Delta t} \tag{5.21}$$

Using the upwind scheme on the advective term, it is approximated by

$$\boldsymbol{e}_i \cdot \nabla f_i \approx \frac{f_i(\boldsymbol{x},t) - f_i(\boldsymbol{x} - \Delta x \boldsymbol{e}_i, t)}{\Delta x} \tag{5.22}$$

Lastly, following the work of Cao *et al.* [4], using the collision term, ie the right hand side, in the downwind direction by simply replacing $\Omega_i(\boldsymbol{x})$ with $\Omega_i(\boldsymbol{x} - \Delta x \boldsymbol{e}_i)$, all the components of the discrete evolution equation are known. Assembling them results in

$$\begin{aligned} &\frac{f_i(\boldsymbol{x},t+\Delta t) - f_i(\boldsymbol{x},t)}{\Delta t} + \frac{f_i(\boldsymbol{x},t) - f_i(\boldsymbol{x} - \Delta x \boldsymbol{e}_i, t)}{\Delta x} = \\ &\qquad = -\frac{1}{\varepsilon\tau}(f_i(\boldsymbol{x} - \Delta x \boldsymbol{e}_i, t) - f_i^{eq}(\boldsymbol{x} - \Delta x \boldsymbol{e}_i, t)) \end{aligned} \tag{5.23}$$

Using lattice units ($\Delta t = \Delta x = \varepsilon = 1$), replacing $\boldsymbol{x}$ with $\boldsymbol{x} + \Delta x \boldsymbol{e}_i$ and rearranging finally gives the LBE as a finite difference equation for f_i:

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i, t+1) - f_i(\boldsymbol{x}, t) = -\frac{1}{\tau}(f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)) \tag{5.24}$$

To show that the derived LBM satisfies the Navier-Stokes equations, the most common approach is to employ the Chapman-Enskog expansion. Here the terms in the LBM are expanded around a small parameter, and subsequently reinserted. After some algebraic manipulation the resulting terms can be identified with those in the N-S equation, up to a certain order. This derivation has been performed in many papers and is not repeated here. The interested reader is referred to [5] for the Chapman-Enskog procedure, and to [21] for the asymptotic analysis by Junk.

5.3 LBM models

While the most widely used variant of the LBM is the single component fluid with bounce-back boundary conditions and the BGK collision term, there are many different options to choose from. Different models come with a different set of advantages and drawbacks, and choosing the right one for the purpose can be critical. In the following sections are descriptions and discussions on a few main aspects of the LBM, such as different collision terms, multicomponent flows and body forces.

5.3.1 BGK collision term

Despite conclusive evidence that the BGK single relaxation time collision term is inferior to other more sophisticated methods [7, 10], it remains in popularity. The BGK collision term is a simplification of the more general form which, instead of a single scalar as a relaxation parameter, involves a full matrix M for the relaxation step:

$$\Omega_i = -M_{ij}(f_j - f_j^{eq}) \tag{5.25}$$

where Einstein summation over repeated indices are assumed. For the BGK term, this corresponds to setting $M_{ij} = \delta_{ij}\frac{1}{\tau}$, which implies a single relaxation time for all hydrodynamic (conserved) and kinetic (non-conserved) moments. This limits the accuracy and stability of the numerical model [7]. However, if one is studying flows well within the stability region of this model, two benefits are obvious: a very easy implementation and a light computational burden. Attempts have been made to improve on this model by using higher order discretization schemes [36], expanding the equilibrium function to third order [11], giving

$$f_i^{eq} = w_i\rho\left(1 + 3(\boldsymbol{e}_i\cdot\boldsymbol{u}) + \frac{9}{2}(\boldsymbol{e}_i\cdot\boldsymbol{u})^2 - \frac{3}{2}\boldsymbol{u}^2 + \frac{27}{6}(\boldsymbol{e}_i\cdot\boldsymbol{u})^3 - \frac{9}{2}\boldsymbol{u}^2(\boldsymbol{e}_i\cdot\boldsymbol{u})\right) \tag{5.26}$$

and also by equilibrating the density fluctuation rather than the total density [13]. The latter involves substituting the equilibrium function (to second order) for

$$f_i^{eq} = w_i\left[\rho_0 + \delta\rho\left(1 + 3(\boldsymbol{e}_i\cdot\boldsymbol{u}) + \frac{9}{2}(\boldsymbol{e}_i\cdot\boldsymbol{u})^2 - \frac{3}{2}\boldsymbol{u}^2\right)\right] \tag{5.27}$$

where the $\delta\rho$ is the density fluctuation, and the total density over a lattice site is given by $\rho = \rho_0 + \delta\rho$. This reduces compressibility effects and results in a *nearly incompressible* model, rather than a weakly compressible.

5.3.2 MRT collision term

Developed by d'Humières (1992) [6], the Multiple Relaxation Time (MRT) model is an approach to model the collision term in a way which is consistent with the formulation of the general LBE. Here all the different kinetic moments are allowed to relax independently from each other, meaning each moment has an independently adjustable relaxation time. This treatment greatly improves the numerical stability, accuracy and physical limits of the model [7], in comparison with the BGK collision term. This manifests in, for example, lower attainable viscosity, and also significantly reduced spurious (unphysical) velocities at interfaces in multicomponent flow [51]. Because relaxation times can be set independently, the shear viscosity and the bulk viscosity are no longer coupled, as in the BGK model. By setting a high bulk viscosity, one can artificially dampen spurious pressure oscillations that can arise with low shear viscosities, and thus acquire increased stability. Also, the remaining relaxation times can be set to further tune the stability of the model. There are no perfect values that will work for every case, so a certain degree of trial and error might be required to find satisfactory values. These improvements come at the cost of a somewhat more complicated implementation and an increased computational burden.

What constitutes the MRT model is a collision term with the full matrix, resulting in the evolution equation

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i\Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = -\Lambda_{ij}[f_j(\boldsymbol{x},t) - f_j^{eq}(\rho,\boldsymbol{u})], \quad i = 1,...,N \tag{5.28}$$

where Λ is the introduced collision matrix, and N is the number of lattice vectors, as usual. To find out how to define and use Λ, d'Humieres carefully constructs a linear mapping from velocity space to moment space. First, introduce a set of vectors $\{\phi^\beta|\beta = 1,...,N\}$, such that the scalar product $\phi_i^\beta f_i$ gives the corresponding kinetic moment m_β. For example, ϕ^1 would appropriately be a vector of all ones, with the result that the scalar product would add all the populations together such that $\phi_i^1 f_i = m_1 = \rho$, thus yielding the first hydrodynamic moment, i.e. density. Given that the underlying lattice structure is chosen properly such that is possesses necessary symmetry, the vectors ϕ^β can be constructed by a Gram-Schmidt procedure from the lattice vectors $\boldsymbol{e}_i$ to form an orthogonal dual basis to the space spanned by f. Here it is said that the space spanned by f is the velocity space, and conversely that the space spanned by m is the moment space. By constructing a matrix M, where each row is a vector ϕ^β, that is $M_{ij} = \phi_j^i$, this matrix will represent the linear mapping of f from velocity space to moment space according to $m_i = M_{ij}f_j$. Since it is a linear operation by an orthogonal matrix, the unique inverse mapping $f_i = M_{ij}^{-1}m_j$ exists. Now let the matrix Λ be defined such that its eigenvectors correspond to the vectors ϕ^β. By this construction, the relaxation process will naturally take place by the matrix multiplication performed in the evolution equation (5.28). But to explicitly be able to choose the relaxation rates, the orthogonality of M is exploited to construct a diagonal matrix $\hat{\Lambda}$ according to

$$\hat{\Lambda} = M\Lambda M^{-1} \equiv Is \tag{5.29}$$

where I is the N by N identity matrix, and $s = (s_1, s_2, ..., s_N)$ are the (inverse of) relaxation rates for each corresponding moment. Equation (5.28) can now be written in the following form

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i\Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = -M_{ij}^{-1}\hat{\Lambda}_{jk}[m_k(\boldsymbol{x}, t) - m_k^{eq}(\rho,\boldsymbol{j})] \tag{5.30}$$

This way the relaxation step is carried out in moment space. The expressions for the moment space equilibrium functions m^{eq} can be found by optimizing the isotropy and the galilean invariance for the model. For details of this analysis, see Lallemand & Luo [22]. These expressions are given for the D2Q9 model in [23], and for the D3Q15 and D3Q19 models in [7]. One notable case here regarding the hydrodynamic moments ρ and $\boldsymbol{j}$, is that $m^{eq} = m$, since they are conserved quantities.

To summarize the steps in which the MRT relaxation step is normally carried out, first the distribution functions are transformed to moment space. Then, using the hydrodynamic components of m, the equilibrium functions $m^{eq} = m^{eq}(\rho,\boldsymbol{j})$ are computed. Relax the different moments, each according to its own rate: $m'_i = s_i(m_i - m_i^{eq})$. Note that since $m^{eq} = m$ for ρ and $\boldsymbol{j}$, $m' = 0$ regardless of choice of relaxation rate for these particular moments. However, this only applies to the special case of a single component fluid, and in the absence of body forces. Finally, what remains to be done is to transform m' to velocity space by $f' = M^{-1}m'$, since the streaming step is performed in velocity space. On a practical note, the coefficients of M^{-1} can be calculated by using the fact that it is an orthogonal matrix, and constructed from the vectors ϕ^i. Therefore

$$M_{ij}^{-1} = \phi_i^j/(\phi_k^j \phi_k^j) \tag{5.31}$$

(note that the superscript indices are not summation indices here). Because of a minimal number of arithmetic operations, the formulation above will yield a somewhat more numerically accurate result, compared to applying a general matrix inverse algorithm. And, obviously, this matrix only needs to be computed once.

In the following section, some detail is provided for the D3Q19 lattice, since this is one of the most widely used models. The moment vector m is arranged as

$$m = (\rho, e, \varepsilon, j_x, q_x, j_y, q_y, j_z, q_z, 3p_{xx}, 3\pi_{xx}, p_{ww}, \pi_{ww}, p_{xy}, p_{yz}, p_{xz}, m_x, m_y, m_z) \tag{5.32}$$

To describe the terms of physical interest so far unmentioned, e relates to the energy, ε to the energy squared, $q_{x,y,z}$ is the energy flux independent of the mass flux, and p_{ij} makes up the traceless ($p_{xx} + p_{yy} + p_{zz} = 0$) viscous stress tensor, where $p_{ww} \equiv p_{yy} - p_{zz}$. In order to set the corresponding relaxation times, the following form is proposed

$$s = (0, s_e, s_\varepsilon, 0, s_q, 0, s_q, 0, s_q, s_p, s_\pi, s_p, s_\pi, s_p, s_p, s_p, s_m, s_m, s_m) \tag{5.33}$$

where the reuse of similar relaxation times for different moments is motivated by symmetry (isotropy, galilean invariance) of the model. The values for s range between 0 and 2. Here the relaxation rates for the conserved moments are set to zero. It is important to note though, that for the inclusion of a body force or for multicomponent flows, these values have to be non-zero (details are provided in the forces section). The moment equilibrium functions are often chosen [7,35] as follows

$$\begin{aligned}
e^{eq} &= -11\rho + 19\left(\frac{j_x^2 + j_y^2 + j_z^2}{\rho_0}\right) \\
\varepsilon^{eq} &= w_{\varepsilon 1} + w_{\varepsilon 2}\left(\frac{j_x^2 + j_y^2 + j_z^2}{\rho_0}\right) \\
q_{x,y,z}^{eq} &= -\frac{2}{3} j_{x,y,z} \\
p_{xx}^{eq} &= \frac{1}{3}\left(\frac{2j_x^2 - j_y^2 - j_z^2}{\rho_0}\right) \\
p_{ww}^{eq} &= \frac{j_y^2 - j_z^2}{\rho_0}
\end{aligned} \tag{5.34}$$

$$
\begin{aligned}
p_{xy}^{eq} &= \frac{1}{\rho_0} j_x j_y \\
p_{yz}^{eq} &= \frac{1}{\rho_0} j_y j_z \\
p_{xz}^{eq} &= \frac{1}{\rho_0} j_x j_z \\
\pi_{xx}^{eq} &= w_{xx} p_{xx}^{eq} \\
\pi_{ww}^{eq} &= w_{xx} p_{ww}^{eq} \\
m_{x,y,z}^{eq} &= 0
\end{aligned}
$$

where $w_{\varepsilon 1}, w_{\varepsilon 2}$ and w_{xx} are free parameters. For optimized stability, Lallemand & Luo [22] found from linear analysis that $w_{\varepsilon 1} = 0, w_{\varepsilon 2} = -475/63$ and $w_{xx} = 0$ are good choices.

The kinematic viscosity ν of the model is related to the viscous stress tensor, thus

$$\nu = \frac{1}{3}\left(\frac{1}{s_p} - \frac{1}{2}\right) \tag{5.35}$$

while the bulk viscosity is determined by

$$\zeta = \frac{2}{9}\left(\frac{1}{s_e} - \frac{1}{2}\right) \tag{5.36}$$

As such, the viscosity of the fluid is defined by setting the relaxation rates s_p and s_e. The remaining parameters do not have much impact on the physical properties of the flow, but are rather used to tune numerical stability. Values for these can be obtained from linear stability analysis for specific cases, or simply through trial and error.

At first impression, it might seem as if this model imposes a great computational burden, as compared to the BGK model. However, since the matrices M and its inverse contain a lot of zeros (see Appendix A), one can write explicit expressions for the transformation of each moment, rather than performing a full matrix multiplication. Moreover, many of the rows in M contain similar coefficients, so by careful combinations of row elements these can be reused, effectively reducing the number of arithmetic operations required. Optimal use of this strategy results in the MRT model using up approximately 20% more computational time over the BGK model [7]. However, the memory increase is virtually none, since the distributions in moment space do not need storing, and thus the only extra memory required is for the matrices M and M^{-1}, which is neglible compared to the storage for the distribution functions.

5.4 Body force schemes

In the models described so far, the term responsible for the body force has been omitted. The reasons for this are that it is not always necessary to include a body force, but also that there are many ways of incorporating this term and a separate treatment is justified for clarity. In multicomponent fluids in particular, the implementation of body force has a large impact on the resulting dynamics, therefore a review of a few forcing schemes is detailed below.

The continuous Boltzmann equation with the force term included, has the appearance

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla_x f + \frac{1}{\rho}\boldsymbol{F} \cdot \nabla_\xi f = \Omega \tag{5.37}$$

This continuous force term ends up in the LBE as a separate term S

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i \Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = \Omega_i(\boldsymbol{x},t)\Delta t + S_i(\boldsymbol{x},t) \tag{5.38}$$

Depending on how the force term in (5.37) is treated, the LBE force term in (5.38) will take on different forms. Possibly the most straight forward way of implementing a body force is to simply set it as

$$S_i = w_i \frac{\boldsymbol{F} \cdot \boldsymbol{e}_i}{\rho c_s^2} \tag{5.39}$$

and calculate the momentum of the fluid as normally done by $\rho \boldsymbol{u} = \sum f_i \boldsymbol{e}_i$. However, due to the LBE being a finite difference form of the continuous counterpart, this implies the forcing term above is constant over the time interval Δt. For cases where the body force is indeed time invariant, this might work well, but for complex inter-particle forces in multicomponent fluids a more accurate treatment is beneficial.

5.4.1 Shan-Chen forcing

One of the early forcing strategies, most commonly used for multiphase flows, was the Shan-Chen forcing scheme [40]. This involves a special treatment of the velocity based on the laws of motion, and is in its current form only applicable for the BGK model. If a force $\boldsymbol{F}$ acts on the fluid particles, a momentum contribution of $\tau \boldsymbol{F}$ would be added to these particles. This added momentum would then have the particles reach a new equilibrium state $\rho \boldsymbol{u}^{eq}$ during the time τ. Based on Newton's laws of motion the equilibrium velocity is then given by

$$\boldsymbol{u}^{eq} = \boldsymbol{u} + \frac{\tau \boldsymbol{F}}{\rho} \tag{5.40}$$

In the Shan-Chen forcing model, this equilibrium velocity is then used to compute the equilibrium function as $f^{eq} = f^{eq}(\rho, \boldsymbol{u}^{eq})$. However, the actual velocity of the fluid is not $\boldsymbol{u}$, but instead given by

$$\boldsymbol{u}^* = \boldsymbol{u} + \frac{\Delta t \boldsymbol{F}}{2\rho} = \frac{1}{\rho}\sum_i f_i \boldsymbol{e}_i + \frac{\Delta t \boldsymbol{F}}{2\rho} \tag{5.41}$$

While performing well in single-phase flows, this forcing has been shown to be thermodynamically inconsistent in multiphase flows, meaning it doesn't match the predicted phase-coexistence curve [19]. It also shows a strong dependence between surface tension and relaxation time, when it should be independent.

5.4.2 He, Shan & Doolen force model

Another treatment, proposed by He *et al.* [15], suggests an approximation to the gradient $\nabla_\xi f$. Since the equilibrium function f^{eq} is the leading term in the expansion of f, it is thus the major contribution. Therefore, one could make the approximation

$$\nabla_\xi f \approx \nabla_\xi f^{eq} = -\frac{\boldsymbol{\xi} - \boldsymbol{u}}{RT} f^{eq} \tag{5.42}$$

The continuous Boltzmann equation with the BGK collision term then becomes

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla_x f - \frac{\boldsymbol{F} \cdot (\boldsymbol{\xi} - \boldsymbol{u})}{\rho RT} f^{eq} = -\frac{f - f^{eq}}{\lambda} \tag{5.43}$$

In order to solve (5.43) the whole equation is first discretized in time, then integrated over one time step. The collision term can be assumed to be constant over one time step [15], in accordance to previous derivations of the LBE. However, this assumption does not hold for the force term, prompting the use of trapezoidal integration for a more correct treatment. This results in a semi-discrete Boltzmann equation

$$\begin{aligned} f(\boldsymbol{x} + \boldsymbol{\xi}\Delta t, \boldsymbol{\xi}, t + \Delta t) - f(\boldsymbol{x}, \boldsymbol{\xi}, t) &= \\ = -\frac{\Delta t}{\lambda}(f - f^{eq}) + \frac{\Delta t}{2}(G|_{t+\Delta t} + G|_t) \end{aligned} \tag{5.44}$$

where $G \equiv \frac{\boldsymbol{F} \cdot (\boldsymbol{\xi} - \boldsymbol{u})}{\rho RT} f^{eq}$. But since there are now quantities from time step $t + \Delta t$ in two terms, this equation is implicit. To resolve this, new variables h and h^{eq} are introduced as transformations of f and f^{eq} according to

$$h = f - \frac{\Delta t}{2} G, \qquad h^{eq} = f^{eq} - \frac{\Delta t}{2} G \tag{5.45}$$

With (5.45), the new evolution equation becomes explicit in h

$$h(\boldsymbol{x} + \boldsymbol{\xi}\Delta t, \boldsymbol{\xi}, t + \Delta t) - h(\boldsymbol{x}, \boldsymbol{\xi}, t) = -\frac{\Delta t}{\lambda}(h - h^{eq}) + G\Delta t \tag{5.46}$$

Following a similar procedure as in the derivation in section 5.2, using $\tau = \lambda/\Delta t$, equation (5.46) turns into the LBE form

$$h_i(\boldsymbol{x} + \boldsymbol{e}_i\Delta t, t + \Delta t) - h_i(\boldsymbol{x}, t) = -\frac{1}{\tau}(h_i - h_i^{eq}) + \frac{\boldsymbol{F} \cdot (\boldsymbol{e}_i - \boldsymbol{u})}{\rho c_s^2} f_i^{eq} \Delta t \tag{5.47}$$

with the equilibrium function f_i^{eq} as used previously, and where

$$h_i^{eq} = f_i^{eq} - \frac{\boldsymbol{F} \cdot (\boldsymbol{e}_i - \boldsymbol{u})}{2\rho c_s^2} f_i^{eq} \Delta t \tag{5.48}$$

Inserting (5.48) in (5.47) results in the LBE evolution equation in its final form

$$\begin{aligned} h_i(\boldsymbol{x} + \boldsymbol{e}_i\Delta t, t + \Delta t) - h_i(\boldsymbol{x}, t) &= \\ -\frac{1}{\tau}(h_i - f_i^{eq}) + \left(1 - \frac{1}{2\tau}\right) \frac{\boldsymbol{F} \cdot (\boldsymbol{e}_i - \boldsymbol{u})}{\rho c_s^2} f_i^{eq} \Delta t \end{aligned} \tag{5.49}$$

Since h is transformed, the following relations must be used to retrieve the correct hydrodynamic moments

$$\rho = \sum_i h_i, \qquad \rho \boldsymbol{u} = \sum_i h_i \boldsymbol{e}_i + \frac{\Delta t}{2}\boldsymbol{F} \tag{5.50}$$

In an actual implementation, one would normally not use the untransformed variable f directly, thus the naming convention h can be replaced for f in equation (5.49) and (5.50) as long as one keeps this in mind, and always uses (5.50) to calculate the velocity.

In a comparison of different forcing schemes for single-phase and multi-phase fluids, Krafczyk *et al.* [19] found this scheme to be among the most physically accurate, absent of some of the numerical artifacts and physical inconsistencies that plagues the Shan-Chen forcing model.

5.4.3 He, Shan & Doolen forcing for MRT model

Although He *et al.* only derived the result of the their forcing scheme for the BGK collision model, similar treatment on the basis of the approximation (5.42) has been done by McCracken & Abraham for the MRT model [28]. To be consistent with the MRT approach and to improve computational efficiency, the addition of the forces is also performed in moment space, thus requiring a transformation of S. This is done analytically according to $\hat{S} = \hat{S}(\boldsymbol{u},\boldsymbol{F}) = MS$. Which in turn means S does not have to be evaluated explicitly in the collision procedure, much like m^{eq} has a predetermined expression. For clarity in derivations, let us from here on adopt the typing conventions $f^+ \equiv f(\boldsymbol{x}+\boldsymbol{e}_i\Delta t, t+\Delta t)$, and $S^+ \equiv S|_{t+\Delta t}$. The MRT evolution equation with trapezoidal integration of the force term then becomes

$$f_i^+ - f_i = -\Lambda_{ij}(f_j - f_j^{eq}) + \frac{\Delta t}{2}(S_i^+ + S_i) \tag{5.51}$$

where Λ_{ij} is the collision matrix, and S_i defined as

$$S_i \equiv \frac{\boldsymbol{F}\cdot(\boldsymbol{e}_i - \boldsymbol{u})}{\rho c_s^2} f_i^{eq} \tag{5.52}$$

To remove the implicity, f is transformed as previously where

$$h_i = f_i - \frac{\Delta t}{2} S_i \tag{5.53}$$

Inserting (5.53) in (5.51) gives

$$h_i^+ + \frac{\Delta t}{2}S_i^+ - h_i - \frac{\Delta t}{2}S_i = -\Lambda_{ij}(h_j + \frac{\Delta t}{2}S_j - f_j^{eq}) + \frac{\Delta t}{2}S_i^+ + \frac{\Delta t}{2}S_i \tag{5.54}$$

Cancelling out terms of S^+, and rearranging those of S results in the explicit equation

$$h_i^+ - h_i = -\Lambda_{ij}(h_j - f_j^{eq}) + \left(I_{ij} - \frac{1}{2}\Lambda_{ij}\right) S_j \Delta t \tag{5.55}$$

where $I_{ij} = \delta_{ij}$ is the identity matrix. As with the forcing model of He, to obtain the correct momentum, one must use the relation in (5.50). Following the same transformation procedure as for the MRT model in section 5.3.2 results in

$$h_i^+ - h_i = -M_{ij}^{-1} \left[\hat{\Lambda}_{jk}(m_k - m_k^{eq}) + \left(I_{jk} - \frac{1}{2}\hat{\Lambda}_{jk} \right) \hat{S}_k \Delta t \right] \tag{5.56}$$

where

$$\begin{aligned} m_i &= M_{ij} h_j & (5.57) \\ m_i^{eq} &= M_{ij} f_j^{eq} & (5.58) \\ \hat{S}_j &= M_{ij} S_j & (5.59) \end{aligned}$$

and $\hat{\Lambda} = \mathrm{diag}(s_1, s_2, ..., s_N)$ contains the inverse of the relaxation times for each moment. To provide example, $\hat{S}$ for the D2Q9 lattice is given by

$$\hat{S} = MS = \begin{bmatrix} 0 \\ 6(\boldsymbol{u} \cdot \boldsymbol{F}) \\ -6(\boldsymbol{u} \cdot \boldsymbol{F}) \\ F_x \\ -F_x \\ F_y \\ -F_y \\ 2(u_x F_x - u_y F_y) \\ u_x F_y + u_y F_x \end{bmatrix} \tag{5.60}$$

For $\hat{S}$ in the D3Q19 lattice, see section 6.2. One of the most pronounced benefits of using MRT with this forcing model is greatly reduced spurious velocities at the interface in multicomponent fluids [51], compared to BGK model with Shan-Chen forcing or similar/equivalent scheme.

5.5 Multicomponent flows

The simplicity of simulating multiphase and multicomponent fluids is one of the strong points of the lattice Boltzmann model. Due to the particle nature of the underlying model, multiple components and their mixing or separation can be implemented without much additional work. Over time, a number of different approaches to multiphase flows have been developed. Here, a few of the more established models will be covered in some detail. First a note of caution; one must be careful to distinguish between multi*phase* and multi*component* fluids. The former refers mainly to a fluid which exist in different phases in the thermodynamical sense, for example H_2O coexisting in liquid phase and gas phase. A particular case might be to investigate bubble dynamics. In this case there is often a large density ratio across the interface, a condition which puts extra demand on a stable model. In multicomponent flow however, some number of different fluids are mixed, which are usually in the same thermodynamical phase (meaning liquid or gas). In these cases the density ratio between two regions of unmixed aggregate fluids is typically low. Here the miscibility condition between the components determine if unmixing (by spinodal decomposition) is taking place, a process sometimes also referred to as phase separation, although spinodal decomposition and thermodynamical phase separation are two distinctly different types of processes. From here on phase separation shall refer to the thermodynamical process, while unmixing or spinodal decomposition refers to the miscibility between liquids. However, two regions of unmixed fluid components close to equilibrium might be referred to as two different phases.

5.5.1 Interaction potential (Shan-Chen type)

Among the many types of multicomponent flow models used today, one could identify three classes of models appearing regularly in literature. The first is the Shan-Chen type model, where non-local interactions are being incorporated into the model through a forcing term, effectively modifying the equation of state to a non-ideal one. The result is a spontaneously emergent phase separation, or unmixing, provided the interaction parameter is above a certain threshold value. This is a so called bottom-up approach, where the physics is incorporated at the kinetic level. This type of implementation adheres to the fundamental mesoscopic nature of the LBM. Although consistent, one drawback is the difficulty to predict the behavior *a priori*. Rather, the resulting surface tension and interfacial properties have to be measured from the resulting simulation.

Representing the general multicomponent fluid is one set of distribution functions for each component σ, where each component follows the evolution equation

$$f_i^\sigma(\boldsymbol{x} + \boldsymbol{e}_i \Delta t, t + \Delta t) - f_i^\sigma(\boldsymbol{x}, t) = \Omega_i^\sigma(\boldsymbol{x}, t)\Delta t + S_i^\sigma(\boldsymbol{u}, \boldsymbol{F}_\sigma) \tag{5.61}$$

To set the interaction strength between two arbitrary components σ and $\bar{\sigma}$, an interaction matrix $G_{\sigma\bar{\sigma}}$ is introduced. This could be seen as an effective global temperature. If the interaction value is set high enough, unmixing/phase segregation will occur, otherwise the components remain mixed. The force responsible for the particle interaction in this model is then described by

$$\boldsymbol{F}_\sigma(\boldsymbol{x}) = -\Psi[\rho_\sigma(\boldsymbol{x})] \sum_{\bar{\sigma}} G_{\sigma\bar{\sigma}} \sum_i w_i \Psi[\rho_{\bar{\sigma}}(\boldsymbol{x} + \boldsymbol{e}_i)] \boldsymbol{e}_i \tag{5.62}$$

where $\Psi[\rho]$ is a local density functional, effectively an equation of state which can take on many forms depending on physical context. Two commonly used definitions are $\Psi = \rho$, and $\Psi = 1 - \exp(-\rho)$. Normally the sum over i in equation (5.62) is taken over the set of nearest neighbors.

The original Shan-Chen model [40] suffered from breakdown of local conservation of momentum, inconsistent thermodynamical behavior, and a viscosity dependent surface tension. Much of this seem to stem from the way the force is incorporated into the model (refer to Shan-Chen type forcing in section 5.4.1). Yu and Fan [51] incorporated the force in moment space with the MRT model, using the method by McCracken and Abraham (see section 5.4.3), and also extended the range of the mean-field interaction potential. In doing so they reduced the spurious velocities by the interface, got viscosity independent surface tension, and attained a means of controlling surface tension and thickness by modifying the influence of the long range interaction force.

Other attempts in improving on the Shan-Chen model include a modified BGK collision term in combination with a equilibrium function to third order [11], and defining alternative non-ideal equations of state such as the van der Waals and

Carnahan-Starling EOS [15, 19]. These modified models have all shown improvement over the original form, in regards to thermodynamical consistency, viscosity independent properties, and spurious velocities arising near interfaces.

5.5.2 Free energy models

A second important class of models are based on the free energy approach, first developed by Swift *et al.* [44, 45]. In contrast to the Shan-Chen interaction potential model, free energy based models infuse prescribed macroscopic properties, which is known as a top-bottom approach. Some argue that this 'shoehorning' of a desired behavior onto the model goes against the idea of the LBM. However one wants to look at this, the results agree well with experiment and theory [20], and gives the benefit of being able to predetermine specific properties of the fluid. On the downside, there have been signs of an issue with mass conservation with high Reynolds numbers [20]. Although this might be a result of an inadequate forcing strategy, as this particular implementation uses the simplest form possible (see (5.39) in forcing section). Also, early models had problems with galilean invariance, but that issue has now been resolved. The free energy approach uses one set of distribution functions to evolve the fluid according to the Navier-Stokes (N-S) equations, and another set to solve the Cahn-Hilliard (C-H) equation for an order parameter which keeps track of the density difference, and thus defines the interface.

Due to the nature of the C-H equation, the corresponding set of distribution functions can operate on a reduced symmetry lattice, thus making it computationally efficient as well. For a 3D grid, the C-H system only needs 7 lattice vectors (D3Q7), as compared to the D3Q19 set normally used to solve the N-S equations.

5.5.3 Index tracking models

Finally a third class introduces the concept of an index tracking method. Here an extra set of distribution functions is again employed, for the evolution of an *index fluid*. This "fluid" satisfies a non-ideal equation of state, and is meant to capture the molecular interaction between fluid components. This interaction gives rise to phase segregation and interface tension, and is incorporated into the LBM model by adding a term which is a function of the gradient of the index fluid. Although numerically robust, it does not capture the C-H equation to the same extent as the free energy models [20].

5.5.4 Remarks

The vast majority of multiphase models describe a two-phase or binary fluid implementation. The only model which provide a straight forward extension to any number of components is the Shan-Chen class, although the work done on more than two fluid components is very scarce. In its original form, the Shan-Chen model would have been an undesirable model to consider for many component flow. However, thanks to the many improvements developed recently, this is now a viable option.

5.6 Physical validity of numerical model

The LBM is normally implemented in lattice units, i.e. a dimensionless system where the time step and the lattice spacing is set to 1 and so on. This can be helpful in ease of implementation, and provides a common ground to compare properties between different implementations like stability regions, relative accuracy and computational efficiency to name a few. But in order to be useful in engineering applications and scientific research, the dimensions will at some point have to be translated into actual physical quantities. Moreover, in a real world case, one must also assess how accurately the physics involved are represented. In more precise terms, the regions of parameter space where the physics of interest are reflected to desired accuracy must be known, and thus definable. This information will also tell us whether a particular problem can be modeled at all. The accuracy can be measured, or quantified, in several ways. Initially, one can get a good sense of the capabilities by analysis of the theory behind the model. In the case of the LBM, there are plenty of theoretical work done. As was stated before, it has been shown that the LBM is of second order accuracy in space and first order in time. Furthermore, we know that only the long wavelength and low frequency features of fluid flow are accurately portrayed, since this is the assumption upon which the model is derived. This implies a restriction on the attainable lower limit of viscosity. Also, empirical measurements show that the model normally starts to break down at lattice velocities of $\sim$0.2, which translates to Ma$\approx$0.3. This clearly shows that for high Mach flows this is not an appropriate model. Moreover, if one wants to achieve accurate high Reynolds number flows, it is essential to implement either the MRT collision term or the Smagorinsky subgrid model [41, 49], preferably both. Other means of testing validity is setting up a benchmark case and comparing the results with both experimental data and analytical solutions, where applicable.

Part III
Model and Implementation

6 Model requirements

As with all computational modeling, one will have to closely examine the case at hand and determine precisely which physical aspects needs to be captured. This can be a difficult task, since even seemingly irrelevant processes may have a large impact on the end result. If the numerical model is too simple, it will fail to represent the physical process. Add too much detail, and it will become difficult to work with and demand excessive computational resources. In this section, a description of the governing physics of a specific pill coating procedure will be given, followed by an identification of equivalent numerical requirements for the model.

The pill coating mixture consists of an liquid three-component blend of ethanol, EC and HCP, where the two latter are polymers. The relative concentration ratio is 0.85 : 0.10 : 0.05, respectively. Initially the solution is liquid and completely mixed, owing to the high fraction of ethanol. As the ethanol evaporates, the miscibility condition changes and the solution enters the spinodal region, giving rise to unmixing and phase segregation of the polymer components. However, at this stage in the process the segregated regions are not purely single component. One phase consists of a large fraction of EC and a small fraction of HCP, and vice versa for the other phase. Later in the process these intra-phase fractions will again separate, resulting in a secondary process of spinodal decomposition. During this entire process, the ethanol is mixed with both polymer phases. As the ethanol evaporates further, the now segregated mixture starts a process of solidifying. This is done gradually as mobility is successively reduced. Finally the mixture becomes fully solid by way of a glass transition. The governing physics in the semi-solidified state is somewhat unclear. Since the components in question are long polymers, there is reason to believe the solution behaves in a non-newtonian manner as a viscoelastic fluid. Some kind of reptation model might be needed to accurately describe the physics here.

So far the mixture has been described from an immersed point of view, in the sense that one could say we are *in* the liquid, looking at what happens. To acquire a full physical picture, the other aspect that need to be considered is the process as seen 'externally'. Since the initial volume fraction of ethanol is large compared to that of the polymers, the mixture will undergo significant shrinking as the ethanol is evaporating, all the while the spinodal decomposition process is taking place. This obviously has an impact on the segregated domains thus formed. Moreover, the liquid mixture, as composed of its initial fractions, is periodically added to the solidifying structure. This has the effect of re-liquefying the upper layers of the semi-solid structure it is added on top of, as the ethanol diffuses into the underlying mixture. An image of what the final structure looks like can be seen in Figure 3. The scale of the microscopic domains are on the order of 1μm.

Turning now to the modeling of the physical phenomena described above. To state the obvious, the numerical model will have to be able to solve the Navier-Stokes equations for a general three-component fluid. It will further have to selectively describe the mixing and unmixing of components, meaning there must be embedded control for how each fluid component interacts with itself and the other components through inter-particle forces. Component properties like viscosity must not only have individual control, but must also be allowed to vary locally as a function of concentration of other components. Particularly the viscosity of polymer components will be dependent on ethanol concentration. Moreover, the unmixing process

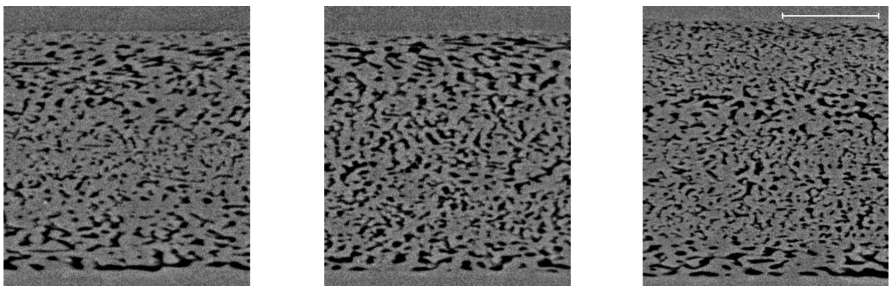
Figure 3: Scanning electron microscope of cross-sections of the solidified pill coating mixture. The scale bar on the top right is 10 μm.

should follow the physics of spinodal decomposition, and in doing so satisfy the Cahn-Hilliard equation, directly or indirectly. To ensure correctness, the resulting interfacial dynamics also need to be accurately captured, i.e. segregated domains following the Laplace law, and displaying correct interfacial profile and interface tension. Since the densities of the fluid components are similar, no special treatment will have to be incorporated to salvage the numerical instabilities that arise in high density ratio flows. An evaporation mechanism must also be present, as to capture the correct distribution of ethanol throughout the mixture. Finally the polymer containing domain must be able to gradually reduce in volume to get a good agreement with experiment. This implies the possible need for an adaptive grid resolution to resolve details.

6.1 Proposed model

Judging from the list of requirements in the previous section, and seeing to the capabilities of the LBM listed in this thesis, most aspects can be covered to a good extent. Certain phenomena such as secondary spinodal decomposition will require more work and will not be implemented at this stage. Also, while there are methods for adaptive grid refinement available for the LBM [50], this will not be addressed due to time constraints. A more resource wasteful approach will have to be taken, where the grid resolution is simply set high enough globally to resolve the necessary detail. With these delimitations mentioned, attention is first directed at which multicomponent model to use. While the free energy models provide a solid physical foundation, all existing implementations (so far found by the author) deal explicitly with two-component, high density ratio flows. Extending to general n-component flow, or at least 3-component, might not be straight forward. The situation is similar for index tracking methods, although it seems free energy models are preferential over this class anyway. In light of this, the Shan-Chen type multicomponent model with the D3Q19 velocity set is suggested. This model can in its current form be easily extended to any number of fluids, and the interaction between fluid components, including self-interaction, is readily defined and easily adjustable. The particular velocity set is chosen because it is a good trade-off between accuracy and efficiency, as the D3Q15 set exhibits some instabilities, and the D3Q27 set is computationally expensive but only marginally more accurate than the D3Q19 set. In the suggested implementation, one set of distribution functions for each component will be used.

Moreover, incorporating some recent improvements with this model should provide the required numerical stability and physical accuracy. Thus, the Shan-Chen multicomponent model will be used in conjunction with the Multiple Relaxation Time (MRT) collision term, as it has shown to be superior over the BGK collision term in a wide range of applications. Further, in modeling the inter-particle forces responsible for the phase-segregation, an extended mid-range potential will be employed, providing additional control over interfacial properties, and reduced spurious velocity currents near the interface. The forcing scheme will be according to He *et al*, as developed for the MRT model by McCracken and Abraham (section 5.4.3). This seems to be one of the most physically accurate forcing schemes developed to date. One of its few drawbacks is its limited ability to model high density ratio flow when compared to other schemes, a non-issue in the case at hand. Naturally, including these improvements into the Shan-Chen model will increase computational burden, however it should provide excellent numerical stability, versatility, and accuracy compared to a more basic implementation. Indeed, the model properties suggested above have been cherry-picked from some of the most promising results of recently published research.

6.2 Model details

In this section the details of the proposed LBM model are given. The D3Q19 velocity set is given by

$$(\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_{18}) = \\ = \left(\begin{array}{ccccccccccccccccccc} 0, & 1, & -1, & 0, & 0, & 0, & 0, & 1, & 1, & 1, & 1, & -1, & -1, & -1, & -1, & 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 1, & -1, & 0, & 0, & 1, & -1, & 0, & 0, & 1, & -1, & 0, & 0, & 1, & 1, & -1, & -1 \\ 0, & 0, & 0, & 0, & 0, & 1, & -1, & 0, & 0, & 1, & -1, & 0, & 0, & 1, & -1, & 1, & -1, & 1, & -1 \end{array}\right) \tag{6.1}$$

with the corresponding lattice weights

$$w_i = \begin{cases} 1/3 & i = 0 \\ 1/18 & i = 1,...,6 \\ 1/36 & i = 7,...18 \end{cases} \tag{6.2}$$

The D3Q19 velocity set is visualized in Figure 4. The evolution equation for each fluid component $\sigma \in \{1,2,...,n\}$ in an n-component flow is written as

$$f_i^\sigma(\boldsymbol{x} + \boldsymbol{e}_i \Delta t, t + \Delta t) - f_i^\sigma(\boldsymbol{x}, t) = \\ -M_{ij}^{-1} \left[\hat{\Lambda}_{jk}[m_k^\sigma(\boldsymbol{x}, t) - m_k^{eq}(\rho_\sigma, \boldsymbol{j}_\sigma)] + \left(I_{ij} - \tfrac{1}{2}\hat{\Lambda}_{ij} \right) \hat{S}_j(\boldsymbol{u}_\sigma, \boldsymbol{F}_\sigma) \Delta t \right] \tag{6.3}$$

where $\boldsymbol{u}_\sigma$ is the component fluid velocity, given by the relation

$$\rho_\sigma \boldsymbol{u}_\sigma = \sum_i f_i^\sigma \boldsymbol{e}_i + \Delta t \boldsymbol{F}_\sigma / 2 \tag{6.4}$$

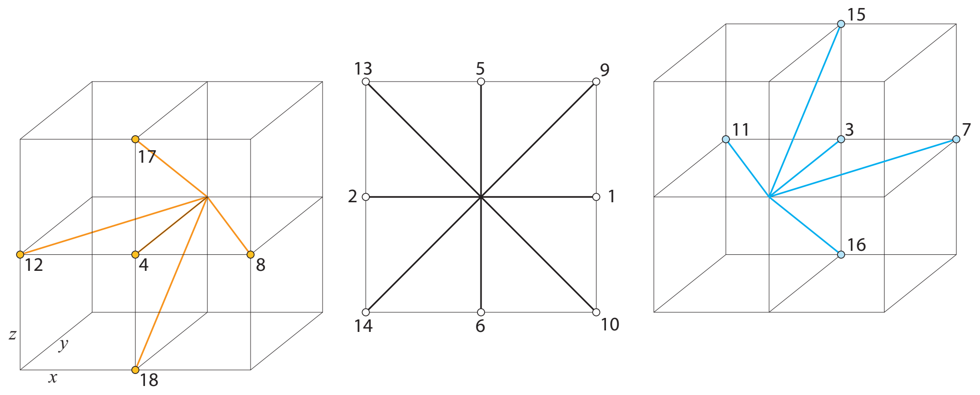

Figure 4: Visualization of the D3Q19 lattice velocity set. The illustration is 'exploded' in the y-direction to reduce clutter.

where the term $\Delta t\boldsymbol{F}_\sigma/2$ enters because of the transformation used in the forcing scheme. The total fluid velocity is then given by

$$\boldsymbol{u} = \frac{\sum_\sigma \rho_\sigma \boldsymbol{u}_\sigma}{\sum_\sigma \rho_\sigma} = \left\{ \rho_\sigma = \sum_i f_i^\sigma \right\} = \frac{\sum_\sigma (\sum_i f_i^\sigma \boldsymbol{e}_i + \Delta t \boldsymbol{F}_\sigma/2)}{\sum_\sigma \sum_i f_i^\sigma} \tag{6.5}$$

Using this velocity, the fluid component momentum $\boldsymbol{j}_\sigma$ is defined as

$$\boldsymbol{j}_\sigma \equiv \rho_\sigma \boldsymbol{u} \tag{6.6}$$

The inter-particle force $\boldsymbol{F}_\sigma$ is calculated by the expression

$$\boldsymbol{F}_\sigma(\boldsymbol{x}) = -\Psi[\rho_\sigma(\boldsymbol{x})] \sum_{\bar{\sigma}} G_{\sigma\bar{\sigma}} \sum_i w_i \Psi[\rho_{\bar{\sigma}}(\boldsymbol{x} + \boldsymbol{e}_i)]\boldsymbol{e}_i \tag{6.7}$$

where the commonly chosen relation

$$\Psi[\rho(\boldsymbol{x})] = 1 - \exp(-\rho(\boldsymbol{x})) \tag{6.8}$$

is used. For n-component flow, the inter-particle interaction matrix G takes on a symmetric n by n form, where element $(\sigma, \bar{\sigma})$ specifies the repulsion strength between component σ and $\bar{\sigma}$. The expressions for vectors $\hat{S}$ and m^{eq} are explicitly given in (6.12). The moment matrix M is listed for the D2Q9 and D3Q19 lattices in appendix A. Note that in evaluating m^{eq}, the velocity and momentum must be used as defined in (6.5) and (6.6). This has the implication that the hydrodynamic moment representing momentum is no longer equal to the corresponding equilibrium function, since $\boldsymbol{j}^{eq} = \rho_\sigma \boldsymbol{u} \neq \rho_\sigma \boldsymbol{u}_\sigma$. What this means is that the momentum is no longer conserved within each component. This is entirely physical, since fluid particles of all components collide with each other and thus exchange momentum. However, total momentum of all components is conserved at each lattice site. To prove this, it can easily be shown that the sum of momenta for all components is equal to the sum of momenta for the equilibrium function for all components. That is, we want to show

$$\sum_\sigma \rho_\sigma \boldsymbol{u}_\sigma = \sum_\sigma \rho_\sigma \boldsymbol{u} \tag{6.9}$$

This is accomplished by simply using the definition of $\boldsymbol{u}$ in (6.5), and inserting in the RHS of (6.9), which results in

$$\sum_\sigma \rho_\sigma \boldsymbol{u} = \boldsymbol{u} \sum_\sigma \rho_\sigma = \frac{\sum_\sigma \rho_\sigma \boldsymbol{u}_\sigma}{\sum_\sigma \rho_\sigma} \sum_\sigma \rho_\sigma = \sum_\sigma \rho_\sigma \boldsymbol{u}_\sigma \tag{6.10}$$

Since momentum is migrating between components, the corresponding relaxation times can no longer be arbitrary, or zero. By setting them to unity, momentum is conserved. This results in the multicomponent diagonal collision matrix $\hat{\Lambda}$ taking the form

$$\hat{\Lambda} = \text{diag}(0, s_e, s_\varepsilon, 1, s_q, 1, s_q, 1, s_q, s_p, s_\pi, s_p, s_\pi, s_p, s_p, s_p, s_m, s_m, s_m) \tag{6.11}$$

Naturally, this matrix can be different for each component. Up to this point it has been assumed that $\hat{\Lambda}_\sigma = \hat{\Lambda}$, but this is not a requirement. On the contrary, setting different viscosities for each fluid species demands component specific relaxation times, since $\nu = \nu(s_p)$.

$$m^{eq} = \begin{bmatrix} \rho \\ -11\rho + 19(\boldsymbol{j}\cdot\boldsymbol{j})/\rho_0 \\ -\frac{475}{63}(\boldsymbol{j}\cdot\boldsymbol{j})/\rho_0 \\ j_x \\ -\frac{2}{3}j_x \\ j_y \\ -\frac{2}{3}j_y \\ j_z \\ -\frac{2}{3}j_z \\ (2j_x^2 - j_y^2 - j_z^2)/\rho_0 \\ 0 \\ (j_y^2 - j_z^2)/\rho_0 \\ 0 \\ j_x j_y/\rho_0 \\ j_y j_z/\rho_0 \\ j_x j_z/\rho_0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{S} = \begin{bmatrix} 0 \\ 38(\boldsymbol{u}\cdot\boldsymbol{F}) \\ -11(\boldsymbol{u}\cdot\boldsymbol{F}) \\ F_x \\ -\frac{2}{3}F_x \\ F_y \\ -\frac{2}{3}F_y \\ F_z \\ -\frac{2}{3}F_z \\ 2(2u_xF_x - u_yF_y - u_zF_z) \\ -(2u_xF_x - u_yF_y - u_zF_z) \\ 2(u_yF_y - u_zF_z) \\ -(u_yF_y - u_zF_z) \\ u_xF_y + u_yF_x \\ u_yF_z + u_zF_y \\ u_zF_x + u_xF_z \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.12}$$

6.2.1 Boundary conditions

Since there will be limited interaction with solids, the bounce-back condition will be used at any solid-fluid interfaces. The solid-fluid interactions will arise in part at the 'bottom' of the container, or simulation domain, where the standard bounce-back implementation will be used. There will also be an interaction at the top of the container, at the free surface, to stop the liquid from escaping the domain. Moreover, the free surface interface at the top will move downwards as the ethanol is evaporating. To model the shrinking of the liquid, the fluid will be constrained by a solid at the top, acting as the free surface, gradually moving downwards. Here arises a need for an arbitrarily placed wall, moving at a varying rate. The standard bounce-back only supports interactions where the solid is aligned with the grid in discrete steps. Thus, functionality needs to be added for a more general solid-fluid interface which does not necessarily coincide with the lattice nodes. Moreover, the moving wall will have to exert a force on the fluid to maintain momentum balance and mass conservation. Both these issues have been treated by Lallemand & Luo [24], and will be used here. For an arbitrarily placed wall, consider a reference node $\boldsymbol{x}_r$ inside the solid, and a distance $0 \leq q < 1$ from the reference node to the actual wall. Instead of just using the reversed population, i.e. $f_{\bar{i}}(\boldsymbol{x},t+\Delta t) = f_i(\boldsymbol{x},t)$, the post-interaction populations are calculated according to an interpolation scheme. The force from the wall, moving at velocity $\boldsymbol{u}_w$, is given by $F_i = 3w_i(\boldsymbol{e}_i \cdot \boldsymbol{u}_w)$. After the streaming step, the populations which have interacted with the wall and are now moving away from the solid are computed in place according to

$$\begin{aligned} f_{\bar{i}}(\boldsymbol{x}_j,t) = q(1+2q)f_i(\boldsymbol{x}_j+\boldsymbol{e}_i\Delta t,t) + (1-4q^2)f_i(\boldsymbol{x}_j,t)- \\ -q(1-2q)f_i(\boldsymbol{x}_j-\boldsymbol{e}_i\Delta t,t) + 3w_i(\boldsymbol{e}_i\cdot\boldsymbol{u}_w), \qquad q<\frac{1}{2} \end{aligned} \tag{6.13}$$

or

$$\begin{aligned} f_{\bar{i}}(\boldsymbol{x}_j,t) = \frac{1}{q(2q+1)}f_i(\boldsymbol{x}_j+\boldsymbol{e}_i\Delta t,t) + \frac{2q-1}{q}f_i(\boldsymbol{x}_j-\boldsymbol{e}_i\Delta t,t)- \\ -\frac{2q-1}{2q+1}f_i(\boldsymbol{x}_j-2\boldsymbol{e}_i\Delta t,t) + \frac{3w_i}{q(2q+1)}(\boldsymbol{e}_i\cdot\boldsymbol{u}_w), \qquad q\geq\frac{1}{2} \end{aligned} \tag{6.14}$$

depending on the value of q, i.e. the wall position.

6.2.2 Extended interaction potential

The extended mid-range interaction potential is implemented according Sbragaglia *et al.* [39]. The motivation behind this approach is to reduce the unphysical spurious velocities that arise near fluid-fluid interfaces. Sbragaglia *et al.* show that the spurious velocities arise as a result of lack of isotropy in the interaction potential forcing term. By expanding the set of lattice points used in calculating the interaction force, one can push the anisotropies to successively higher orders in the Taylor expansion.

For the D2Q9 model, 4th, 6th, 8th, and 10th order of isotropy forces are implemented, while for the D3Q19 model the 4th, 6th, and 8th order of isotropy. For preliminary tests the 6th order force is adequate, while for the full simulations the 8th order (D3Q19) and 10th order (D2Q9) forces have been used. Figure 5 shows the different sets of lattice vectors for the D2Q9 model.

6.2.3 Evaporation and solidification

The mechanisms for evaporation and solidification are model independent, in the sense that their implementation does not differ between choices of collision and forcing schemes etc. The former can be modeled by removing a certain amount of ethanol as it enters the interfacial region between the mixture and air. This can be achieved by simply maintaining a fraction of the populations entering the fluid-air interface, expressed as $f_i'(\boldsymbol{x},t) = af_i(\boldsymbol{x},t)$, where $0 \leq a < 1$. A better way is to set a fixed velocity boundary condition by the interface, for example with the Zou-He

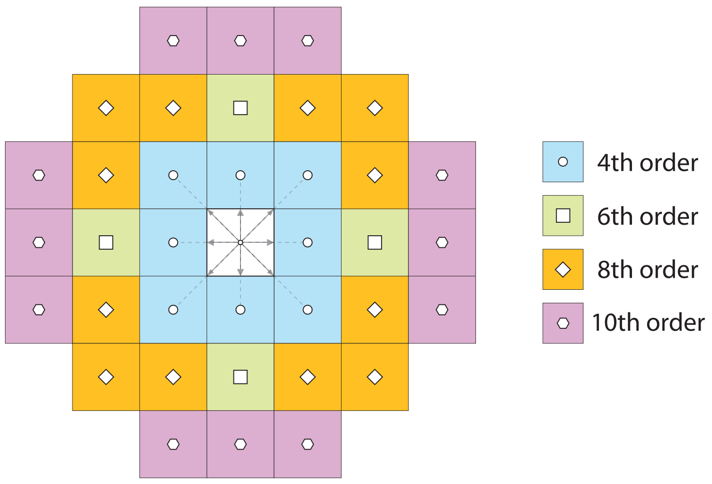

Figure 5: Visualization of different envelopes in calculating the pseudo-molecular interaction force for different orders of isotropy in the D2Q9 lattice.

implementation. Having the velocity pointing out from the domain for the ethanol component only will ensure that matter flows out of the fluid region. The magnitude of the velocity will determine the rate of outflux/evaporation.

This simple evaporation model is accurate enough for our current purpose of phenomenological modeling, but might need to undergo a more rigorous treatment at a later stage to account for e.g. temperature dependent evaporation.

In its native form, the LBM does not provide any intuitive mechanism to achieve solidification. Therefore, in modeling this phenomenon one must find a physical process that mimics the gradual reduction of mobility. One such way is to emulate solidification by flow through homogeneous porous media. By successively decreasing permeability of the porous medium, the liquid will find transportation increasingly 'difficult', thus reducing mobility and effectively appearing as a gradual transition to a solid state. The equation for flow in porous media is governed by the Brinkman equation

$$\langle \nabla p \rangle = -\frac{\mu}{k}\boldsymbol{u} + \mu_e \nabla^2 \langle \boldsymbol{u} \rangle \tag{6.15}$$

where $\langle\ \rangle$ denotes volume average, k is the permeability of the porous medium, and μ_e is an effective parameter controlling the viscosity and increasing with solid fraction such that $\mu_e/\mu > 1$. The Brinkman equation is a generalized form of Darcy's law, where boundary effects between regions of different porosity are being considered. According to [27], this behavior can be incorporated into the LBM model by an additional force term

$$\boldsymbol{F}_b = -\frac{\mu}{k}\boldsymbol{u} = -\frac{\nu\rho}{k}\boldsymbol{u} \tag{6.16}$$

Moreover, the relaxation time is set as $\tau = \tau_e = 3(\mu_e/\rho) + 1/2$. Hence, there are two free parameters, μ_e and k, to set in controlling the degree of solidification (fluid mobility) of the ethanol-polymer mixture.

7 Implementation

This section aims at providing motivation and detail regarding the choice of computer implementation of the proposed LBM model. Aspects such as algorithms, object oriented programming, data structures, multi-threading and code optimization will be covered. Though its main purpose is describing the programming approach of this thesis, it is also meant to serve as a guide for anyone interested in implementing their own LBM variant or improving/building upon the current one suggested in this document.

7.1 Programming model

Before choosing implementation approach, considering platform and programming model, one must have a clear picture of the inherent problematics and requirements. It is intuitive to think of the LBM with all its variants as a modular system. Different collision terms could be considered one class of models, and different lattice structures another, for example. Here, the choice of lattice structure should ideally not affect the inherent workings of the collision term, as the algorithm is the same. With a suitable level of abstraction, the LBM could be constructed of a different set of independent modules, each dealing with their inherent tasks and communicating the result with the other modules. This construction makes it relatively easy to build a library of functionality and improve or add new modules as the need arise, providing good scalability. It also facilitates in comparing computational efficiency between different programmatical approaches. Although care must be taken not to introduce a level of abstraction too high, as the computational efficiency might be compromised. This is indeed another important practical consideration. Fluid simulations, especially in three dimensions, require an immense amount of data to be processed, and are as such inherently computationally expensive. Therefore, it is of great importance to choose an efficient programming model which can utilize near peak computer performance. This immediately rules out convenient scripted programming languages such as Matlab.

7.1.1 Programming language

With the main requirements being modularity and computational efficiency, the choice falls upon C++. This is a natural choice since it is a mature, widely used, and object oriented language, with a proven track record of robustness and efficiency. Using GNU/Linux the code can be readily compiled using the standard open source GNU-C++ compiler (`g++`). For performance, one can choose between a few parallel computing extensions such as OpenMP, MPI and Posix threads. In this work, OpenMP has been utilized because of its ease of use. This is a straight forward extension which works by adding parallel *directives* in the code. Most often the workload in large loop bodies is distributed over all CPU cores. If there are no complex data dependencies, as is the case for the LBM collision step, existing code can easily be parallelized without much additional effort.

7.1.2 GPU computing

Another interesting approach in high performance computing is that of GPU (Graphics Processing Unit) computing. GPUs were originally intended for real time processing of computer graphics. Their construction differs from a CPU in that they are specialized to process vast amounts of data in a similar manner, and therefore consist of a large number of computational cores that all perform the same operations to a large set of data. This approach is called SIMD (Single Instruction Multiple Data). It was soon noticed that this tremendous computing power could be harnessed for computations of a more general nature. GPU computing lends itself extremely well to applications which are inherently data parallel. Since the LBM exhibits data parallelism to a large degree, a brief excursion into GPU computing is also covered. Here the programming language used is CUDA C, developed by Nvidia.

7.1.3 Objects and inheritance

One of the central properties of object oriented programming in C++ is that of *inheritance*. Since the implementation relies on inheritance, a brief explanation of the concept is provided. Objects in C++ are most often realized by *classes*. A class can be described broadly as a collection of data and/or functionality. Often the functionality is related to the data belonging to the class. Central in the object oriented paradigm and in utilizing the flexibility that comes with it, is the ability to create new objects, or classes, on the back of existing ones. In doing so, the new object inherits the properties of the existing one, all the while new functionality can be added. As an example, consider a class describing a geometric object, call it `Polygon`. Belonging to this class we define a set of points describing the shape. Now, we can define new classes, *deriving* from this class, such as `Rectangle` and `Triangle`, since they both share the property that they consist of a set of points. However, for example, the area of the derived objects are calculated differently, and thus have to be defined in each derived class. Here, we save the effort of having to re-implement point storage for the two derived classes, as this is *inherited* from the *base* class. Obvious is also the different levels of abstraction between the base class and the derived class. One could further derive a class `Square` from the `Rectangle` class, and so on.

The field of object oriented programming (OOP) and all the concepts and functionality that comes with it is vast. The above explanation is merely meant to touch upon the very fundamental idea that underpins OOP. For details on this topic, there is an endless supply of books to consult.

7.2 Framework

At its most abstract level, the LBM implementation consists of several types of modules, as mentioned above, each of which handles a separable part of the LBM. At the time of writing, these are

- **LBMSystem**
  This module provides the base for the simulation environment. Some global

parameters such as grid dimensions are held here, but its main purpose is to tie the different modules together and synchronize some operations between them. All parts which comprise the entire system can be accessed from this module. The particle population data is also accessed from here.

- **Kernel**
The Kernel defines the various steps, and the ordering of these, in the main time evolution algorithm. It invokes all the necessary procedures involved in propagating the simulation one time step, such as boundary conditions, collision and streaming. However, the functionality of these procedures are determined elsewhere, in keeping with the object oriented approach.

- **Collision**
As the name implies, everything related to the collision phase of the algorithm is handled here. This, together with the stream node, are the most computationally intensive modules of the framework. It is thus of utmost importance that this part is highly optimized, if peak performance is desired.

- **Stream**
This module takes care of the streaming step. Depending on data structures and data dependencies, this can be done in a few different ways. In certain algorithms the collide and stream steps are fused into one, reducing memory bandwidth, although this requires a special algorithm (described in section 7.4).

- **Lattice**
Here details regarding the lattice structure, velocity set, and their corresponding weights are kept.

- **Solid**
This module holds information about any solid obstacles in the fluid domain, and stores interfacial population data used in solid-fluid boundary treatments, such as the bounce-back boundary condition.

For a simple schematic overview how these are connected, see Figure 6. Each of

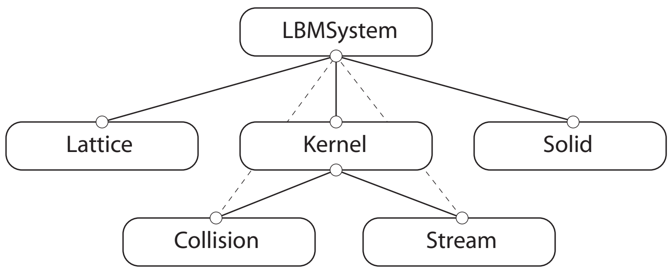

Figure 6: Simplified schematic showing the connections between the different modules. There are also additional classes providing functionality for visualization and initial conditions.

these modules are realized in C++ as classes. To avoid code duplication, and to make the framework extendable, inheritance is utilized where appropriate. The most abundant use of inheritance is with the Collision class, since many different collision models have been implemented, while they still share common properties to a certain extent. A subset of the Collision inheritance tree can be seen in Figure 7.

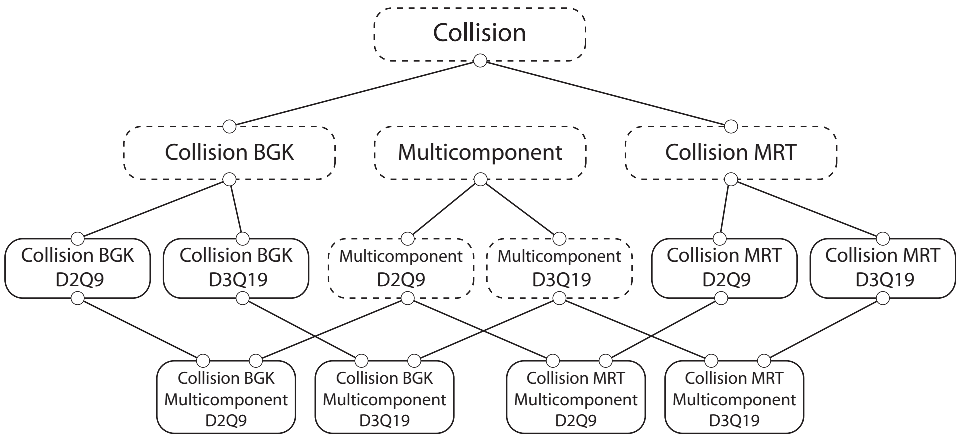

Figure 7: Schematic view over inheritance tree for multicomponent collision models with the BGK and MRT collision term. The dashed boxes represent abstract classes which cannot be used on their own. Note that this is only a subset of the available models.

7.3 Data structure

An important consideration in any application where performance is required, is the choice of data structure. That is, how the information that is to be processed is organized in computer memory. Fundamentally, the reason for this is that memory access speed is slow compared to the CPU processing speed. A situation which has only become more accentuated in the past years since CPU computational speed is increasing much faster than memory access speed. This brings the implication that it usually takes longer to fetch a value from memory than to perform the desired arithmetic operations on it. In such situations it is said that memory access is a bottleneck, or that the computation is *bandwidth limited*.

7.3.1 Cache

To circumvent memory bottlenecking, an interim memory between the CPU and main memory exists, called a *cache*. This memory is small and fast, and can provide data to the CPU much faster than from the main memory. However, for this configuration to be effective, data must be copied into the cache before it is used. Which in turn means the computer, or the cache handler, will have to guess what data will be used next, in order to have it available in cache before the CPU asks for it. The only reasonable assumption it can make here, is that this data resides close in memory space to the previous data just accessed. So if the CPU first accesses data from memory address `100`, the cache handler will immediately start copying data from address `101`, `102`, `103`,... etc into the cache. By accessing the data in the order it is loaded into cache, i.e. sequentially, optimal memory bandwidth will be achieved, thus minimizing bottlenecking. The data access pattern is usually referred to in terms of stridedness. Sequential access is small strided, or 1-strided.

Reading data from memory regions far apart is called large strided access, and is to be strictly avoided for computationally intensive parts of the code, if possible, as this is detrimental for performance.

In reality, usually 3 levels of cache exist, L1, L2 and L3 cache, in a hierarchy. Here L1 cache is the smallest and fastest, while L3 is the largest and slower than L1 and L2. However, the principles described above still apply, and in following these one should produce optimized code.

7.3.2 Data ordering

While it seems trivial that data is stored and accessed in a sequential manner, there are a few different approaches to be considered. Since each grid element consists of a number of populations (distribution functions), and in some cases multiple sets of populations for multicomponent flow, different ordering among these can be employed. For single component flow, the options are to store grid *node ordered* or *direction ordered*. In the former, all populations that 'live' in one lattice node are residing in contiguous memory space, whereas in the latter all populations of a certain lattice direction are contiguous (see Figure 8). One can also arrange data

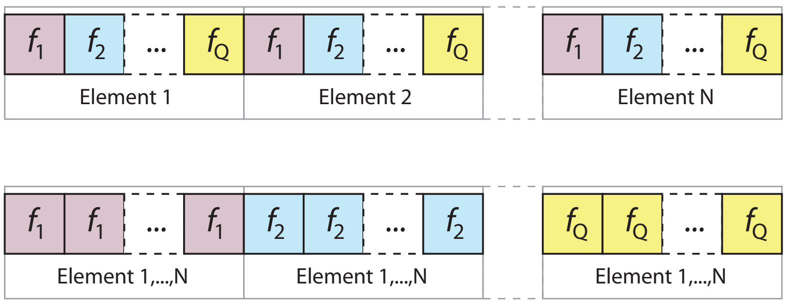

Figure 8: Different data ordering schemes. Top: Node ordered. Bottom: Direction ordered.

in a form of periodic direction ordering, such that only small subsets of the grid are repeatedly directionally ordered. The main difference with these approaches is that the grid element ordering supports local memory access for the collision step, but must perform large strides for the streaming step. With the direction ordered approach, large strides have to be taken for gathering data for the collision step, while the streaming step becomes more sequential in nature. However, the large stridedness in the collision step can be avoided by pre-calculating and storing the density and velocity for the entire grid, thus requiring more steps in the algorithm and more memory, whereas in the grid ordered approach this is not necessary (for single component fluids).

7.3.3 Memory locality and domain decomposition

Although small, the cache hierarchy does allow room for some maneuverability. If data is accessed within a memory region small enough to fit inside the cache, data access can be non-sequential while still maintaining high performance. Code which is utilizing this feature is said to exhibit good *memory locality*, or *cache locality*. This issue is especially important to consider in the streaming step, since this is a non-local operation where data is shifted from one memory location to another.

Preferably then, considering locality, the memory space for the surrounding nodes should be as close as possible. However, because memory is linear, when constructing the simplest possible grid, large strides will be inevitable for certain streaming directions (see Figure 9). A common workaround to improve on this scenario is to

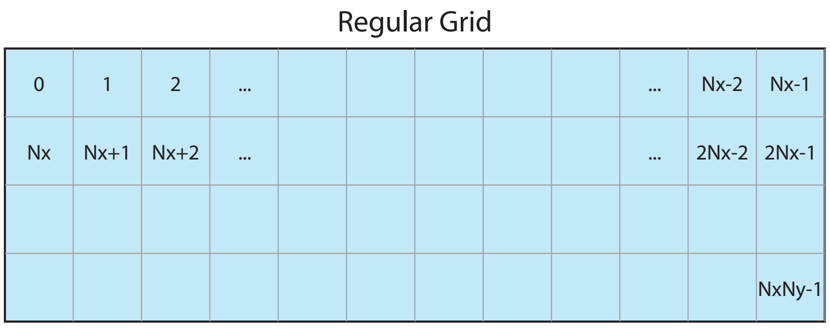

Figure 9: Memory addressing for a regular grid in 2D. Note that big strides in memory will be made when streaming in y-directions.

split the main grid into smaller subgrids, a method referred to as *domain decomposition*. By computing one subgrid at a time, cache locality is improved due to its smaller size (Figure 10). Furthermore, by constructing the subgrids such that the

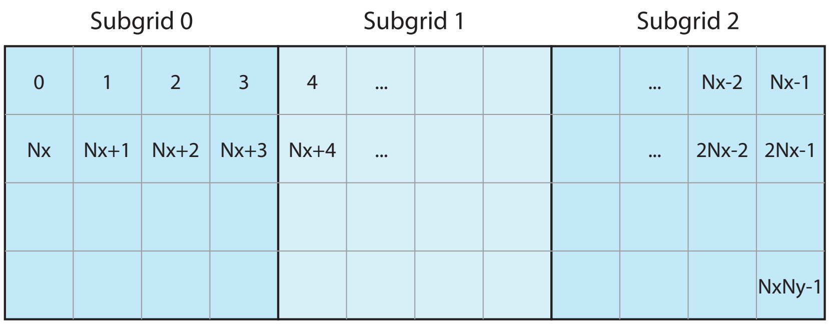

Figure 10: Main grid decomposed in domains, or subgrids. Strides will still be large, but since the subgrid is small, a bigger portion of it should reside in cache and there should be performance gains.

memory layout is contiguous within each subgrid, locality is improved even further (Figure 11, 12). This brings both advantages and disadvantages. Apart from the improved locality, each subgrid can be more easily handled separately, and thus is advantageous for certain algorithmic improvements and also for distributing the workload on a bigger cluster. The obvious disadvantage is that the communication across the borders between different subgrids becomes somewhat more convoluted.

7.3.4 Data dependencies

In performing the streaming step, the populations of the current grid element are distributed to the surrounding grid elements according to their corresponding lattice direction. In doing so, the previous populations residing in these locations will obviously be overwritten. This creates a data dependency, in which a temporary

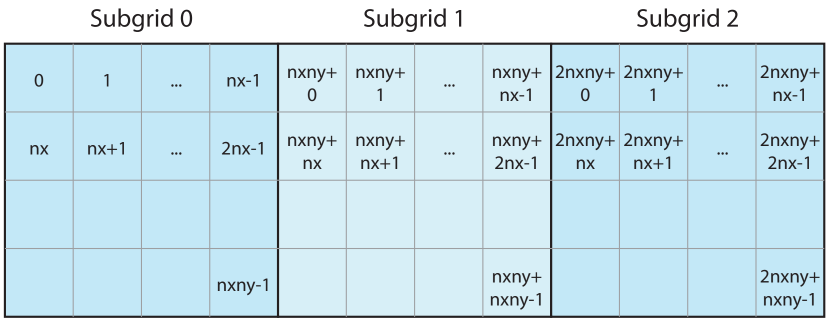

Figure 11: Alternative memory addressing for subgrids. Here each subgrid is contiguous in memory, and thus memory locality should be optimal within the domain.

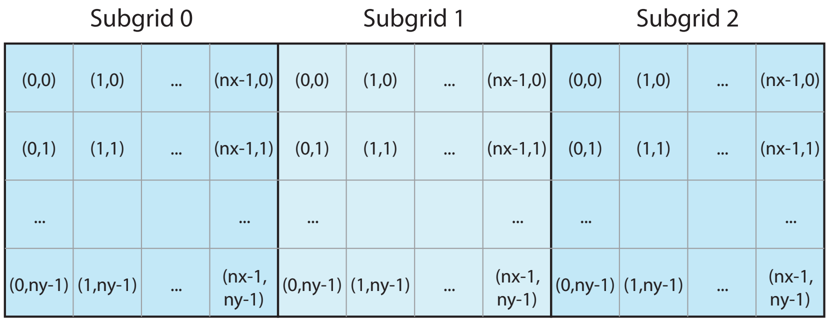

Figure 12: Subgrids with contiguous memory, showing (x,y) local coordinates instead of memory addressing. To access a particular subgrid, one simply have to calculate a fixed memory offset. Here it becomes obvious that this data ordering makes it straight forward to distribute computational parcels over a large cluster; in the computation step, no information is needed about the neighbors or where we are in the global grid.

memory storage have to be introduced to resolve it. The simplest approach here would be to allocate two separate grids A and B, thus requiring twice the amount of memory. The post collision populations would be calculated for the entire grid and stored in B, followed by streaming back to A. A more efficient method would be to fuse the collision and streaming step and write to grid A or B for even or odd time steps respectively. So in the first time step data would be read from A and written to B. The following time step would read from B and write to A, and so on. This eliminates memory accesses by half, thus greatly reducing the limitation set by memory bandwidth. However, this still requires twice the memory. Considering a 3D multicomponent fluid can easily take up several gigabytes of memory, this is preferably avoided. Employing domain decomposition, one can reduce memory footprint by computing one domain at a time, and thus only require temporary storage the size of the domain. Although care needs to be taken with the populations streaming out of the domain, and into the neighboring domains. Here is another data dependency which needs to be addressed. This is normally solved by padding each domain with a layer of 'ghost nodes' or buffer elements (Figure 13). These elements serve as a temporary storage, and at the end of the computation step these will be transferred to their correct location. Naturally, this involves some extra computation overhead, but makes up only a small fraction of the total computational time. It is a small sacrifice for a large reduction in memory

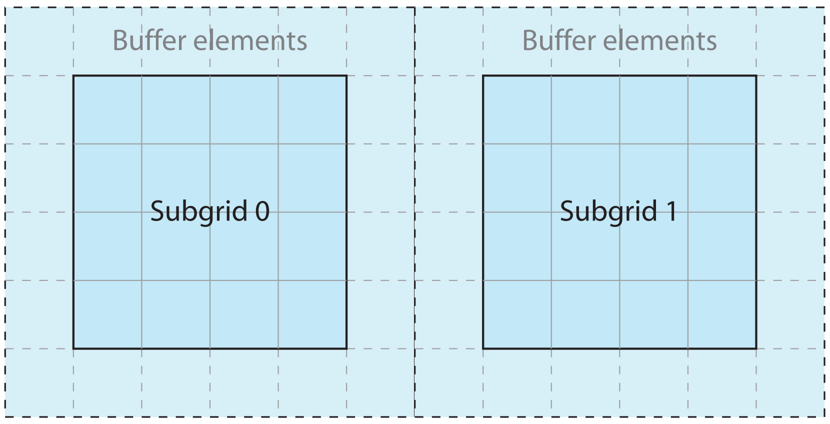

Figure 13: To make each subgrid computation step independent, and to resolve data dependencies, a layer of buffer elements are added. The populations residing in the buffers, as a result of streaming, will be transferred to the corresponding elements after all the subgrids have been computed.

usage.

7.3.5 Chosen data structures

A few basic tests for single component flows were initially carried out, and it was found that the grid element ordered approach was fastest for the computer architecture at hand. For multicomponent flows, each additional set of populations are stored in succession (see Figure 14). Moreover, the grid is decomposed in sub-

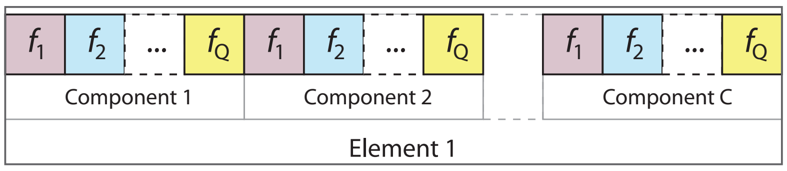

Figure 14: The chosen data ordering scheme for multicomponent fluids.

grids, with the memory contiguous in each subgrid as described above (Figure 11). Without the subgrid approach, the streaming step scales very badly with parallel computing, providing essentially no speedup at all when running on all four cores compared to a single core. Using the subgrid, the computational performance scales very well with number of cores and problem size. In addressing the data dependency issue with the streaming step, memory space corresponding to one additional subgrid is allocated. The post collision values are stored in the temporary subgrid, and then streamed back to the main grid. Further, each subgrid is padded with a layer of ghost nodes, or buffers, to accommodate for the populations streaming out of the domain, thus increasing the subgrid dimensions by two elements for each spatial dimension. The fractional increase in the total memory requirement per subgrid then becomes

$$\frac{(n_x+2)(n_y+2)(n_z+2)}{n_x n_y n_z} \tag{7.1}$$

where n_i is the number of grid elements in the spatial dimension $i \in \{x, y, z\}$. For a typical subgrid size of $n_x = n_y = n_z = 40$ this amounts to a 16% increase over the storage for the main grid.

It should be noted however, that an exhaustive investigation was not pursued, and that further performance increases could possibly be had with careful choice of data structure, such as partial direction ordering as described above.

7.3.6 Indexing and data addressing

With the data structure in place, it should be described how to access the desired populations. For a three-dimensional grid, where each grid element supports any number of fluid components, and where each component consist of a set of lattice populations, we are essentially addressing a 5-dimensional structure. That is, to access a certain population we need to specify 3 spatial indices for the grid element, 1 fluid species index, and 1 population index. Although C++ supports "multidimensional" data storage, this is not possible to utilize when passing data between functions through memory pointers. Also, since all computer memory is one-dimensional, the multidimensional array indexing is just a mapping to one-dimensional space. Therefore, all populations are stored in a large one-dimensional array to begin with, where the mapping from 5 indices to 1 address element have to be determined 'manually'.

First, let us find the *local* index to an element in a subgrid. The total number of elements, or populations, for one subgrid will be

$$N_{f,sub} = n_x n_y n_z CQ \tag{7.2}$$

where C is the number of fluid components, Q the number of populations per component, and n_i the number of grid elements in each spatial dimension for the subgrid. With the data ordering determined previously, the index idx_{sub} to population q, for fluid component c in subgrid element (i,j,k) is calculated as

$$idx_{\text{sub}} = (kn_x n_y + jn_x + i)CQ + cQ + q \tag{7.3}$$

where all indices are 0-based, that is $q \in [0,\ Q-1]$ and analogously for the other indices. Since the main grid consists of several subgrids, the memory offset to the current subgrid must also be computed. Let the main grid consist of d_x, d_y, d_z subgrids in each spatial dimension, respectively, such that the total number of grid elements along one dimension is $N_x = d_x n_x$. The memory offset to access the elements in subgrid (I,J,K) is then

$$idx_{\text{offset}} = (Kd_x d_y + Jd_y + I)N_{f,sub} \tag{7.4}$$

Thus, we can calculate the global index to the main grid as

$$\begin{gathered} idx_{\text{global}} = idx_{\text{offset}} + idx_{\text{sub}} = \\ (Kd_x d_y + Jd_y + I)N_{f,sub} + (kn_x n_y + jn_x + i)CQ + cQ + q \end{gathered} \tag{7.5}$$

While this might appear cumbersome to deal with, some sub-expressions like idx_{offset} are precomputed once and used to create a pointer to the current subgrid. As such, only the local indices need to be computed, and it transparently maps to the global indices without the user having to think about it. It is this feature that makes this particular memory model so favorable for distributed cluster computing.

7.4 CUDA model

With some modifications the existing code can be adapted to the CUDA C language, whereby the large computational power provided by GPU based hardware can be utilized. When designing code for CUDA enabled devices one must understand the special data execution model, which is in turn intimately connected to the hardware. As a result of the large number of computational cores, and the capability to launch and handle a very large amount of concurrent threads, programs are often designed such that one thread is responsible for only one small fragment of the total problem. This fragment is usually one grid node for grid based methods, and the LBM is no exception. Moreover, in order to map all data elements to threads in a scalable manner, CUDA builds on the concept of *thread blocks* (Figure 15). A thread block is a group of threads with some inherent dimensional properties. It can be 1D, 2D and 3D, depending on what suits the problem best, and as thus each thread come with a spatial index as a unique identifier. The entire problem is then divided up into a number of thread blocks, also with a spatial structure, collectively forming what is called a *grid*, to cover all data elements. By structuring the threads in this manner, one does not need to know the specifics of the GPU hardware, i.e. the number of cores, and the problem should map equally well to a budget, high end, or future device. Nvidia GPU devices come with different types of memory

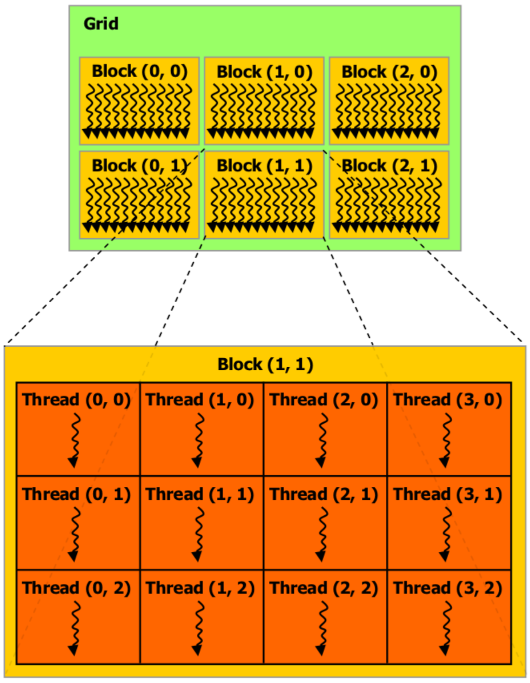

Figure 15: Schematics of the CUDA threading model. The entities labeled 'Block(x,y)' are thread blocks. Each thread block has access to a small low latency memory (shared memory) where data can be shared with the thread block only. The shared memory can be seen as a kind of user managed cache.

which can be divided up into two broad categories, global and shared. Global memory is the large main memory, where access is relatively slow (high latency), while the shared memory is local to each thread block, small in size but with very fast access (low latency). Key in obtaining peak performance is to use these types of memory efficiently. A common strategy is to copy memory from global memory to shared memory before using it. However, global memory must be accessed in a specific way to maintain high performance. Just as in the case of the CPU and its cache-structure, data should preferably be read from global memory in a sequential manner. This is even more important in the case of GPU computing. Without going into too much technical detail, data should be read in data packets of 128 bits (4 single precision floating point values, or 2 double precision), and should be read more or less strictly sequentially (small deviations are allowed, given it is done within certain restrictions). Such memory access is called *coalesced*. Once data resides in shared memory, it is available for use with more random access without performance loss. One can see the coalesced global memory access together with shared memory utilization as a form of user managed cache. For more details on the CUDA model, see [2].

The strategy used in this CUDA port is to let one thread compute and stream the populations of one lattice node. All populations corresponding to the thread block are first read to shared memory, whereby the new populations are computed. In order to achieve coalesced reads, the memory model is changed to *direction ordered*, as in Figure 8 (bottom). Using the node ordered memory model does not amount to coalesced access, and thus resulted in performance only on par with the CPU version. When using the temporary subgrid domain for storing the post-collision populations, as shown in Figure 16, along with the revised data storage pattern, the performance obtained was a speedup of a factor ~ 8 over the equivalent CPU implementation. However, examining the reads and writes one quickly notices that this uses 2 memory

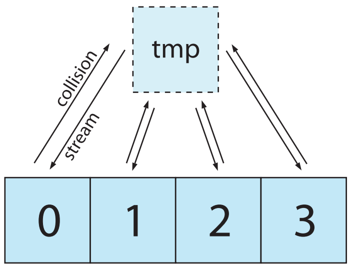

Figure 16: Subgrid memory model for the collision and streaming step, as used in the CPU code. The post-collision values are first stored in the temporary subgrid. When all these have been computed, they are subsequently 'streamed' back to the originating subgrid in the regular grid, thus resolving the data dependencies.

transfers per population: 1 where the post-collision value is stored in the temporary domain, and 1 when storing in the regular grid in the streaming step. Since memory bandwidth is usually the limiting factor in GPU computing, it is worth spending some time trying to optimize access patterns. By using a second grid, thus requiring twice the amount of memory, one can fuse the collision and streaming step by storing the post-collision value directly at the streaming destination in the second grid. For the next time step, the post-collision values are transferred back to the first grid, and so on. This way each data element only need to be transferred once per time step, cutting memory bandwidth in half. Indeed, the result using this memory model was almost twice the previous performance, yielding an impressive ∼14x speedup over the CPU version using the less optimized memory access pattern. However, the drawback is using twice the amount of memory. This is touching on a weak spot for GPU computing, since these devices normally carry much less memory than what is available to a CPU, thereby greatly limiting the size of the problem. But we shall see that the memory usage can be improved upon, by considering the property where the memory is contiguous in each subgrid. For a 2d problem, the subgrids that make up the total grid can be viewed as the left illustration in Figure 17. Looking at how these map to memory, the subgrids might as well be laid out in a straight line, as depicted in the right illustration in Figure 17. With this in mind a new updating

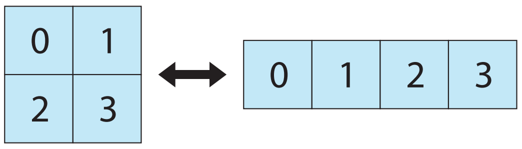

Figure 17: Equivalence of memory layout with subgrids of internally contiguous memory.

scheme can be devised, utilizing the fused collision and streaming step, but where only one extra subgrid is used. By computing one subgrid at a time, there will always be one subgrid available for storage, albeit the memory location for this is shifting. If subgrid 3 is first collide+streamed to the spare subgrid, its old location will become obsolete and can be used for storage. Thus, subgrid 2 is subsequently collide+streamed to the location for the old subgrid 3, and so on. This algorithm is repeated until all subgrids are computed for time $t + 1$ (Figure 18, top). Now, in commencing the computation for time step $t + 2$, there will be a spare storage where subgrid 0 resided for time t. Thus, in order to compute time step $t + 2$, one repeats the process but in reversed direction (Figure 18, bottom). This application

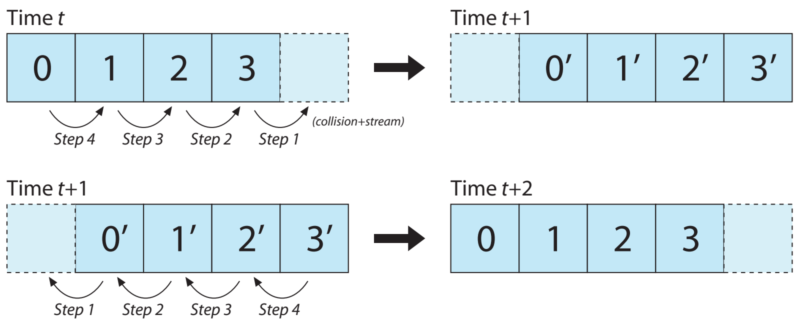

Figure 18: Schematics of revised updating model, being bandwidth *and* memory efficient.

of reduced memory usage with subgrids, in conjunction with bandwidth efficiency in ”shuffling” the subgrids back and forth, the performance is unimpaired compared to that of the full dual grid model, while only using the extra memory of one subgrid. This highlights the importance of careful implementation for performance; while the first naive CUDA implementation produced the correct result, it had less than 1/10th of the computational performance of the refined ’shuffle’ model.

Only a few LBM models were tested due to time constraints. For multicomponent flows, the CUDA approach was only taken for the D2Q9 model, since the memory requirements for a 3D multicomponent model greatly exceed that of the GPU devices available to the author. A brief summary of the models tested is listed in Table 4.

7.5 Computational performance

In selecting a model with a set of desired properties, speed could be a determining factor. Also, in order to compare results with other implementations and algorithms, a good measure of computational performance is necessary. A very commonly used measure in the context of LBM is the number of lattice sites updated per second. The conventional unit used is MLUPS (Million Lattice Updates Per Second), and this has been measured for the most relevant models - single component BGK and MRT (2d and 3d), and multicomponent BGK and MRT (3d). Also, a few GPU models have been measured. The hardware used was a quad-core i5 2500k consumer desktop CPU (Intel 'Sandy Bridge' architecture) running at 4.0GHz, accompanied by 8Gb of system memory. For the GPU simulations, an Nvidia GTX580 with 512 CUDA cores and 3Gb of onboard 'device' memory was used. Both CPU and GPU versions use the 'shuffle' algorithm described in section 7.4. The CPU results are listed in tables 1, 2, and 3. The GPU results are listed in table 4. Best performance for the 2d models was found with a subgrid size of approximately (200, 200) lattice units, and for 3d models (40,40,40) was a good choice. For GPU models, domain sizes should be chosen as a power of 2 to maintain performance.

	BGK D2Q9	MRT D2Q9	BGK D3Q19	MRT D3Q19
$N = 1$	170	80	45	28

Table 1: Performance measure in MLUPS of single component LBM models. All models use single precision floating point.

	BGK D3Q19 (4th)	BGK D3Q19 (6th)	BGK D3Q19 (8th)
$N = 2$	10.4	8.3	4.6
$N = 3$	6.0	4.5	2.3

Table 2: Performance measure in MLUPS of multicomponent BGK LBM models using the He forcing scheme. N represents the number of fluid components, and the number in parenthesis indicate the order of isotropy for the midrange pseudopotential force.

	MRT D3Q19 (4th)	MRT D3Q19 (6th)	MRT D3Q19 (8th)
$N = 2$	7.3	6.4	4.2
$N = 3$	4.6	3.9	2.3

Table 3: Performance measure in MLUPS of multicomponent MRT LBM models using the He forcing scheme.

Model	CPU MLUPS	**GPU MLUPS**	Speedup
BGK D2Q9	170	1900	11
MRT D2Q9	80	1000	12.5
BGK D3Q19	45	725	16
Multicomponent $N = 3$:			
BGK D2Q9 (8th)	9.0	180	20
BGK D2Q9 (10th)	6.9	145	21
MRT D2Q9 (10th) ($\Psi = \rho$)	5.5	125	23
MRT D2Q9 (10th) ($\Psi = 1 - e^{-\rho}$)	3.3	115	35

Table 4: Performance results and comparisons for GPU implementations. All measures using single precision floating point.

One obvious conclusion is that 3d models are expensive, and multicomponent ones even more so. Two major costs come with the multicomponent models. First there is one additional set of population functions per additional fluid component that have to be calculated. Second, the interaction potential model requires a force to be calculated where the sum of the density is taken over a set of neighbors. The set of neighbors grows with increased order of isotropy, and so does the computation time. Interestingly, the BGK and MRT models are nearly equally costly for the 8th order interaction force. This demonstrates that the computation of this force is much larger than the extra overhead required by the MRT collision term. As such, there really is no reason to choose the BGK model in this scenario, unless one favors a simple implementation.

Looking at the GPU results in table 4, it is observed that the more computationally demanding the model is, the greater the benefit over the CPU version. This can be explained by reasoning that the GPU memory bandwidth becomes less of a bottleneck as the computational burden increases.

As for comparing with other implementations in literature, this proved to be difficult. There were no clear cases found where similar hardware was used, a necessity for a meaningful comparison. Mostly large cluster implementations are benchmarked, and the type of CPU's used were high-end workstation models, usually from an older generation, whereas the results in this work are based on a single modern consumer desktop CPU. However, one independent means of measuring the efficiency of the algorithm is to calculate a theoretical performance limit on the basis of available data bandwidth. Considering only the data quantity that needs to be moved, disregarding any calculations, one can get an upper bound on the performance. Here, each population on every lattice site will be read once and stored once, resulting in two memory accesses per population. However, because of the CPU architecture, when writing a value to memory, the corresponding address must reside in the cache. This results in two memory accesses (since the store location is not previously accessed and thus uncached): one for fetching the address to the cache and one for subsequently storing the value. Thus requiring a total of 3 memory accesses. Using the single precision floating point datatype, occupying 4 bytes, this gives that $3 \cdot 4 \cdot q$ bytes will have to be transferred to update one lattice cell for any DdQq model. With knowledge of maximum bandwidth BW MBytes/s for the hardware architecture in use, the maximum MLUPS measure can be calculated as

$$\text{MLUPS}_{\text{max,CPU}} = \frac{BW}{3 \cdot 4 \cdot q} \tag{7.6}$$

The GPU architecture does not come with the requirement that the write address resides in cache, and thus the maximum performance is given by

$$\text{MLUPS}_{\text{max,GPU}} = \frac{BW}{2 \cdot 4 \cdot q} \tag{7.7}$$

To find the maximum bandwidth for the CPU, a STREAM-like benchmark was performed. This test consists of sequentially copying large amounts of data from one array to another, a large number of times (to even out fluctuations), while recording the elapsed time. From this a reliable value of the actual bandwidth can be deduced. Using all CPU cores, this test amounted to an attainable bandwidth of $\sim 23 \cdot 10^9$ Bytes/s for the system at hand. The peak memory bandwidth of the GPU device used is given by manufacturer specifications as 192 GBytes/s. The performance of the D2Q9 and D3Q19 implementations can now be evaluated against this measure. The results are listed in Table 5.

Model	MLUPS measured	MLUPS max	Efficiency
CPU:			
BGK D2Q9	170	213	80%
BGK D3Q19	45	101	45%
GPU:			
BGK D2Q9	1900	2667	71%
BGK D3Q19	725	1263	57%

Table 5: Efficiency of single component models, as measured against maximum bandwidth performance in which only data traffic is considered.

Considering this measure is for data moving only, the achieved results indicate an efficient implementation for at least the D2Q9 versions. The 3d models involve a larger number of arithmetic operations per site, due to the 3 components of the velocity, so are expected to come with a lower efficiency compared to the 2d case. However, measured values are still unexpectedly low and indicate some further optimization can likely be done for the 3d models.

It is not so straight forward to estimate the maximum bandwidth performance for the multicomponent models, due to the much greater extent of memory operations. However, some conclusions can be drawn from the straight performance measures. Due to the fact that the computation time for the interaction force is dominating, any optimizations should be aimed at this step. Currently this is evaluated per component, per lattice site. One possible improvement might be to pre-compute these forces, which is essentially a convolution process. This approach will incur a relatively small memory cost. Further, by using the fact that the force that fluid A feels from a neighboring fluid B is the same as the reverse situation, but with a switch in sign, the number of computations can be reduced by half.

Regarding GPU performance, the results clearly show that the LBM lends itself well to massively parallel computing. Even if the memory constraint is a limiting factor, the GPU approach can be an excellent low cost option for interactive feedback of low resolution simulations during the modeling process. Moreover, using GPUs is also an interesting option for cluster computing, since the problem can then be broken down into subdomains small enough to fit inside GPU memory.

Part IV

Simulation and results

8 Benchmarking of proposed model

To assess the physical accuracy and numerical stability in the proposed model, a series of benchmark tests will be performed. In some cases, numerical solutions will be compared to analytical solutions, where these exist. Other LBM models than the proposed one will also be subject to benchmarking, in order to quantitatively evaluate the difference between them. These will mainly be used to compare numerical artifacts, stability limits and computational performance. Results which may be of use when selecting models for other projects.

There are many standard benchmark tests available, some with analytical and some with experimental solutions. Because of the application in question, mainly multicomponent benchmarks will be treated.

Unless otherwise stated, dimensionless lattice units will be used throughout the section.

8.1 Poiseuille flow

One of the most widely used benchmarks regarding fluid flow is flow in a straight channel, also known as Poiseuille flow. This makes for a good benchmark, since it is easy to set up and there exists a simple analytical solution. For a 2-dimensional flow along the x-direction, the velocity profile across the channel (y-direction) is given in lattice units by

$$u_x(y) = \frac{4u_{max}}{D^2}(Dy - y^2) \tag{8.1}$$

where D is the diameter of the channel, and u_{max} is the prescribed velocity in the center of the channel.

To achieve steady state Poiseuille flow with the LBM, boundary conditions are set according to Zou-He. Along the inlet, a velocity profile is prescribed according to (8.1), with $u_{max} = 0.15$. At the outlet the pressure is set to the constant value $\rho = \rho_0 = 1.0$. Viscosity is set to $\nu = 0.05$. Walls are introduced by defining the grid elements along the top and bottom as solid. The initial velocity through the channel is set to zero, and the simulation is allowed to run until equilibrium has been reached. Results for the single component BGK and MRT models are plotted in Figure 19. In this simple regime, the BGK and MRT models perform essentially identically, with a 0.52% overshoot in velocity at the center of the pipe, where the profile is taken along the cross-section at the middle of the simulation domain.

8.2 Von Karman vortex street

By placing an obstacle in a stationary fluid flow, usually a sphere, cylinder, or cube, a periodic disturbance develops behind the obstacle, given that the Reynolds number is above a certain threshold value. This disturbance is called a *von Karman vortex street*, and is a common benchmark to validate that a solver captures some degree of the physics involved. Formulated differently; if a solver is unable to reproduce the von Karman vortex street, it is probably not very accurate at all. Here, this benchmark is only carried out for the proposed model, i.e. with the MRT collision term. Using a similar setup as with the Poiseuille flow test in section 8.1, a cylinder with radius $r \approx 14$ is inserted near the center of the channel. The viscosity is

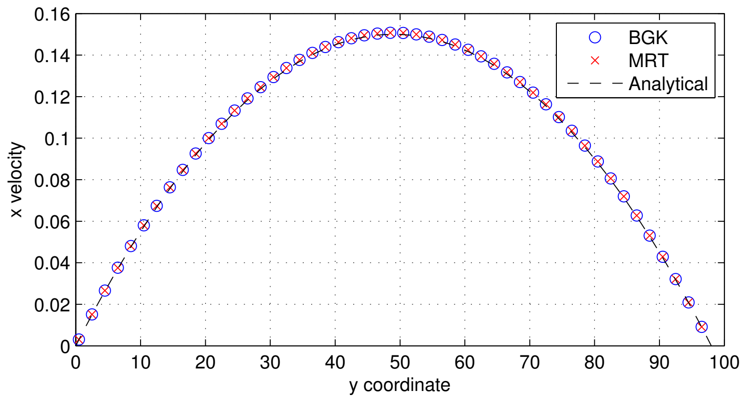

Figure 19: Measured velocity profile along 2-dimensional channel, compared to analytical poiseuille profile. The boundary conditions are prescribed velocity at inlet (left), and constant density at outlet. Here, the BGK and MRT models perform almost identically, and differ 0.52% from the analytical profile at the center of the channel. The grid dimensions are $(N_x, N_y) = (400{,}100)$.

changed to $\nu = 0.064$, and the velocity profile to $u_{max} = 0.2$ in order to acquire $Re = 100$. The simulation is initiated as previously in section 8.1, and propagated for 50,000 time steps, after which a snapshot is taken. The result is plotted in Figures 20,21, and are clearly displaying the characteristic pattern of a (confined) vortex street.

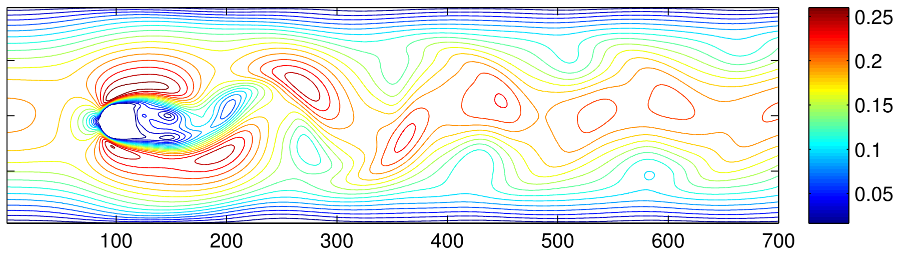

Figure 20: Contour lines of the velocity magnitude of a von Karman vortex street flow at $Re = 100$. The grid dimensions are $(N_x, N_y) = (700{,}200)$.

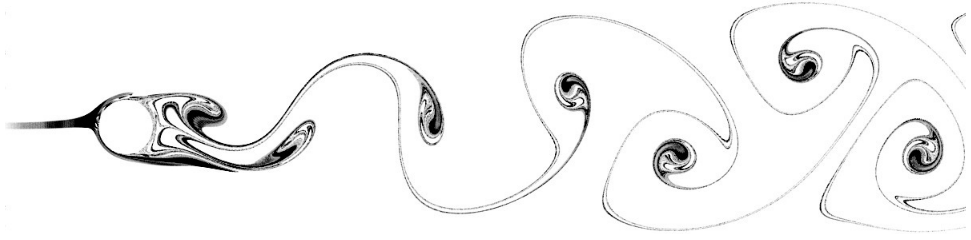

Figure 21: Streamed particles in a von Karman vortex street flow at $Re = 100$ from benchmark simulation.

8.3 Spinodal decomposition phase diagram

As an initial test, the phase diagram for a two-component mixture will be retrieved. To find the transition point to spinodal decomposition, the *spinodal*, the liquid will be initialized with a certain volume fraction ϕ_A of 'red' component. This measure is of course normalized such that the volume fraction of 'blue' component is simply $\phi_B = 1 - \phi_A$. The interaction parameter G_{12} is then successively increased until phase separation is observed. Same component interaction is kept constant at $G_{11} = G_{22} = 1.0$. This is repeated for many different volume fractions ϕ_A until a clear pattern is seen. The result is plotted in Figure 22. Qualitatively, the measured

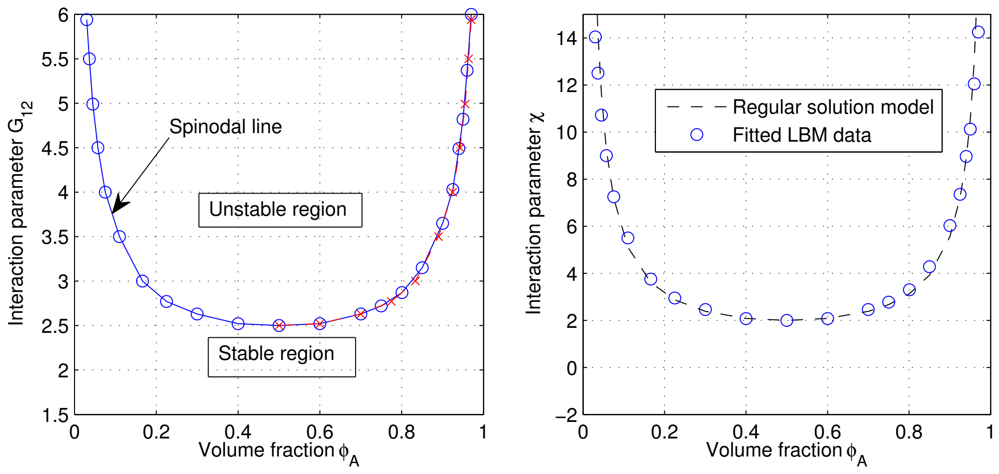

Figure 22: The spinodal, according the to LBM multicomponent model (left). The red dashed line with crosses are the measured values for $\phi_A \leq 0.5$, mirrored across $\phi_A = 0.5$ to demonstrate symmetry. Also plotted is the analytical solution for the spinodal line, with the LBM data fitted (right).

LBM data show similar appearance of the spinodal line as for the regular solution model. The interaction parameter G_{12} here is not exactly the same as χ in the regular solution model, for which the spinodal point at $\phi_A = 0.5$ assumes the value of $\chi = 2$, although they do have the same function. To compare the qualitative behavior, the LBM data can be fitted to the analytical solution for the spinodal line according to the regular solution model. This model predicts the free energy F for a binary liquid to be

$$\frac{F_{mix}}{k_B T} = \phi_A \ln \phi_A + \phi_B \ln \phi_B + \chi \phi_A \phi_B \tag{8.2}$$

The spinodal for a particular value of the interaction parameter χ is where

$$\frac{d^2 F_{mix}(\phi,\chi)}{d\phi^2} = 0$$

holds. Taking the second derivative of (8.2), and using that $\phi_B = 1 - \phi_A$, gives the interaction parameter at the spinodal as a function of ϕ

$$\chi_{sp}(\phi) = \frac{1}{2}\left(\frac{1}{\phi} + \frac{1}{1-\phi}\right) \tag{8.3}$$

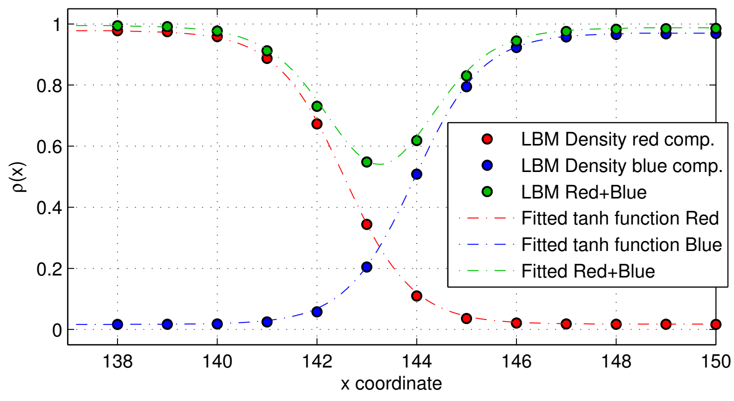

Figure 23: The density profile across the interface of a suspended droplet, simulated with the MRT model using 10th order isotropy interaction potential. The profiles correspond very well to the analytical solution of a tanh profile. The values used are $G_{12} = 2.5$, and $\nu = 0.1$.

which is plotted in Figure 22 together with the fitted LBM data (right). They show excellent qualitative agreement, which is probably to expect since both the interaction potential model and the regular solution model have in common the use of a mean field approximation.

8.4 Suspended droplet

Further testing the multicomponent model, an immiscible binary fluid shall be investigated. The case will be a droplet consisting of one component, immersed in a surrounding fluid of the other component. The simulation will be allowed to run until equilibrium has been reached, i.e. when then velocity and pressure fluctuations that arise as a result of a slight off-equilibrium initial condition have attenuated to an insignificant magnitude. The grid resolution for all tests is $(N_x,N_y) = (200,200)$, using periodic boundary conditions.

8.4.1 Interface profile

In the interaction potential model, the force responsible for separating two immiscible components originates from a pseudo-molecular interaction. Macroscopically, the separation process and the properties of the resulting segregated domains is governed by the Cahn-Hilliard (C-H) equation. Since there exists no obvious mathematical connection between the two approaches, and neither has there been any attempt in proving one, what one can do is measure the results from simulation and compare with experiment and analytical solutions. Solving the C-H equation for a spherical droplet gives that the resulting density distribution across the interface should follow a tanh function. In Figure 23 is a plot of the density distribution of a droplet with radius $r = 40$, and the corresponding distribution for the surrounding fluid. The fit to a tanh profile is in great agreement, giving evidence of physical correctness. By increasing the interaction parameters G_{ij} in equation (6.7), the interface width can be adjusted. However, narrower interface widths come at the cost of increased spurious velocities and reduced stability.

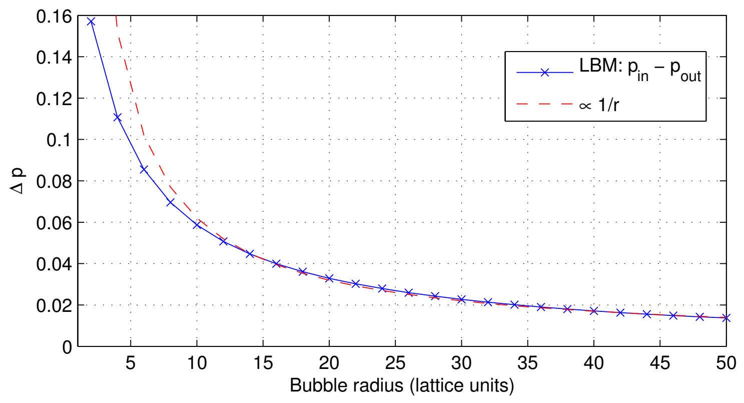

Figure 24: Comparison of LBM with analytical result of pressure difference across the interface of a circular droplet for $\nu = 0.1$.

8.4.2 Laplace law

Due to surface tension effects at the droplet interface, the pressure p_{in} inside the droplet will be higher than the pressure outside, p_{out}. Moreover, the pressure difference $\Delta p = p_{in} - p_{out}$ will be inversely proportional to the radius r, as given by the *Laplace law*

$$\Delta p = \frac{\sigma_0}{r} \tag{8.4}$$

where σ_0 is the surface tension coefficient. The pressure is directly proportional to the density in equilibrium, and can be easily acquired in lattice units by $p = c_s^2 \rho$. Simulations were run for droplet radii in the range $2 \leq r \leq 50$ with the MRT model using 10th order isotropy. A plot of the pressure difference compared to a $1/r$ function can be seen in Figure 24. There is good agreement for larger droplets, although for smaller drops the internal pressure is not high enough to match the Laplace law with $1/r$ appearance. This might in part originate from the way the interaction force is incorporated, considering discrete lattice effects. In this case 3 layers of neighboring lattice nodes are used in calculating the separating force, thus spanning a total of 7 lattice nodes. For small droplets, this would span across the interface in every direction, thus smearing out the force and creating an unphysical result. Although, as the representation of any curved shape degrades significantly when the shape is approaching a similar size as that of the grid element itself, inherent discrete lattice effects is probably the biggest contributor to unphysical behavior at this scale.

8.4.3 Spurious velocities

A side effect of the discretization and of the truncation of higher order terms in the expansions involved, is the emergence of spurious velocities. These entirely unphysical velocities are particularly prominent at the interface between fluid components, as a result of the strong forces that apply in these regions, and can reach speeds fast enough to destabilize the entire simulation. Even while inside the stability region they might still cause significant inaccuracies, as these velocity currents can spread out far from the interface and affect surrounding regions. Any attempt in

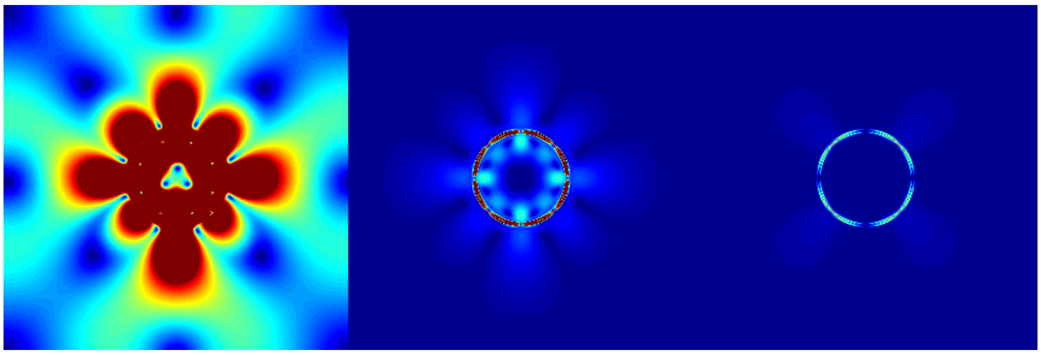
Figure 25: Visual guide to extent of spurious velocities for a droplet of 80 lattice units across, simulated with Shan-Chen BGK (left), He force model BGK (middle), and MRT model (right). All three images are plotted to the same scale. For corresponding quantitative comparison, see Figure 26.

modeling multicomponent flows with phase separation should see to reduce these artifacts as far as possible. Thankfully, there are a number of ways of achieving this. It has been demonstrated [19, 39, 51] that the use of the MRT collision term, the He forcing scheme, and the extended mean-field interaction potential all serve to reduce these currents. Carried out here is a series of test which measure and compare the spurious velocities for different models and forcing schemes. All tests have been performed by examining a droplet with $r = 40$ on a (320,320) grid, using the interaction potential model with $G_{12} = 2.8$, $G_{11} = G_{22} = 0.5$, and with viscosity set to $\nu = 0.1$. The densities of the two components are initialized to be identical, with $\rho_1 = \rho_2 = 1.0$. For the MRT D2Q9 model, the following relaxation times have been used for the freely tunable values: $s_2 = 0.5, s_3 = 0.7, s_5 = s_7 = 1.414$. The value of s_2 controls the bulk viscosity, and was set to 0.5 (corresponding to a very high viscosity) in order to dampen the transient fluctuations as quickly as possible. Noteworthy as well is that when setting s_7 very near $\sqrt{2}$, the spurious velocities reduce significantly. This seems to indicate a connection to the symmetry of the lattice.

First, a study is made of the magnitude of the spurious velocity for the different multicomponent models: Shan-Chen, He forcing, and MRT. The interaction potential forcing is calculated to the 10th order for all models. After equilibrating for 25,000 time steps, the velocities were measured. The result is portrayed in Figure 25 and 26. From these images it is clear that the MRT model is superior in regards to spurious velocities at and around the interface. The magnitude (in lattice units) of the non-local spurious velocities are $u_s^{SC} \approx 0.0015$, $u_s^{He} \approx 0.00012$, and $u_s^{MRT} \approx 0.00003$ for the three models, respectively. These spurious velocities will spread to and, if large enough, possibly affect the neighboring region. In this respect, the MRT model is more than an order of magnitude better than the Shan-Chen model. The spurious velocities inside the interface are $u_s^{SC} \approx 0.022$, $u_s^{He} \approx 0.0016$, and $u_s^{MRT} \approx 0.0003$. The interfacial spurious velocities are much larger in magnitude than their non-local counterpart, and of greater concern since they tend to destabilize the entire simulation if large enough. Here the spurious

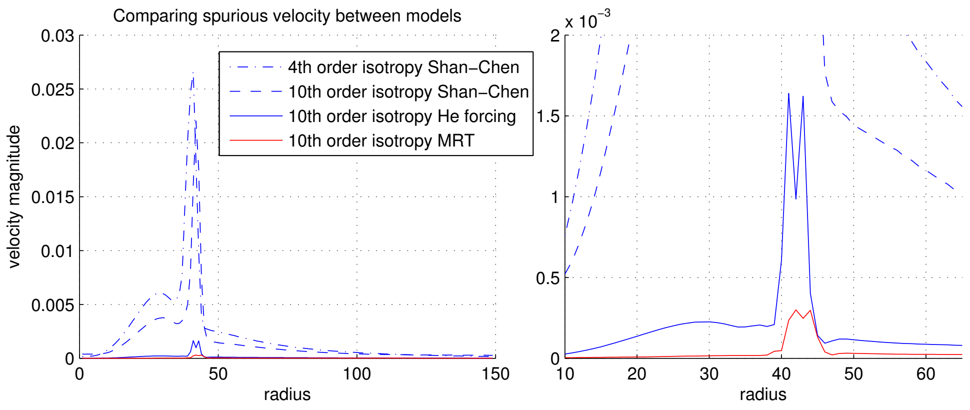

Figure 26: Comparison of spurious velocities between different models. Values are radially averaged to account for the radial asymmetry of spurious velocities seen in Figure 25.

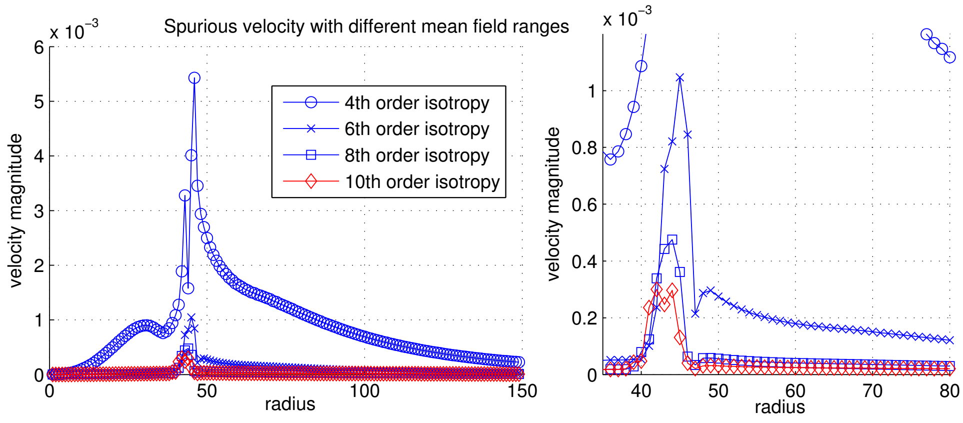

Figure 27: Spurious velocity (radially averaged) across a droplet interface for the MRT model for different orders of isotropy in calculating the mean field force. Right image is a close up of 6th, 8th, and 10th order.

velocities are almost two orders of magnitude lower for the MRT model, when compared to the Shan-Chen model. This gives strong indication to a much increased stability envelope for the MRT model. To further examine the MRT model, a test is set up to evaluate how the order of isotropy for the interaction potential affects the spurious velocities. A droplet was set up, similar to previous examples with $r = 40$, and the velocity profile across the interface was plotted for 4th, 6th, 8th and 10th order of isotropy (Figure 27). First, it can be seen that the 10th order has the best properties. However, it is not by a significant margin when compared to 8th order. Though some three times larger than the 10th order, the 6th order force is still a much better choice than the 4th order, but with little extra computational cost. This can be a good choice when performing repeated tests while iterating through parameters, when simulation speed is of concern. The 8th and 10th order are both good choices for accurate simulations, and the 4th is probably best avoided if stability is of concern. To give an idea of the anisotropic nature of the spurious currents, a vector plot is provided in Figure 28.

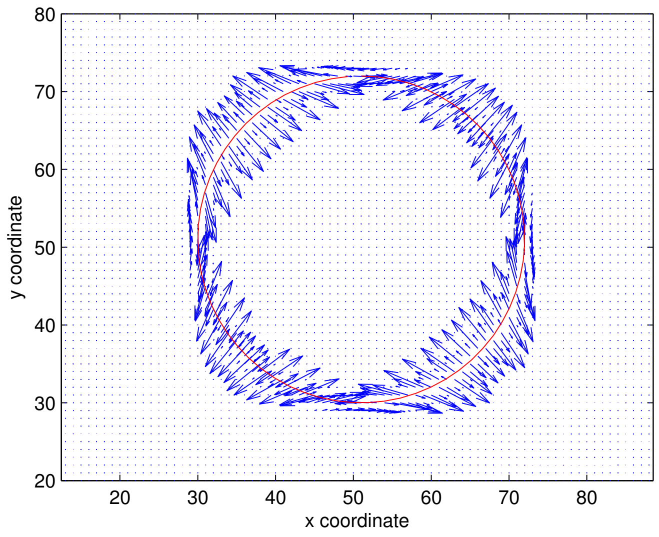

Figure 28: Spurious velocity vectors (exaggerated), showing distinct anisotropy.

8.5 Conclusion of benchmarks

It has been demonstrated that, expectedly, the MRT model is physically sound as it can accurately reproduce some basic fluid phenomena such as Poiseuille flow and the von Karman vortex street. Furthermore, tests involving multicomponent fluids show that relevant physical properties are reflected in this regime as well, as shown by the phase diagram, Laplace law and the interface density profile. The discrepancy in the Laplace law for the smallest of droplets means some inaccuracy in resolving the smaller structures. However, since these are transient, it is not believed to have any significant impact on the larger structures formed over time in the spinodal decomposition process.

The comparison of spurious velocities with other multicomponent models show clear evidence that the MRT model is a very good choice, which is in agreement with research literature. Although the He forcing scheme alone is a large leap in stability compared to the Shan-Chen forcing, even if just used with the BGK collision term. This might be something to consider for cases where one is not near the stability limit. Lastly, the increased degree of isotropy in calculating the inter-particle force provides an extra means of achieving better accuracy and stability. Put together, the MRT collision term with the He forcing scheme, and using the interaction potential according to Sbragaglia *et al.* provide a very competitive model.

9 Modeling and simulation

In order to model the case described in section 6, a number of different methods have to be employed (section 6.1) to capture the necessary physics. Some of the phenomena that needs modeling are evaporation, solidification and partial solidification, and phase separation through spinodal decomposition. Putting them all together at once would introduce a large number of unknown parameters, resulting in a very complex setup. Therefore, each phenomenon will first be modeled separately to find a working methodology and a suitable set of parameter values. When each subcase has been successfully modeled, they will all be combined in an attempt to simulate the full picture. In all cases, the proposed model described in section 6.2 is used.

In the sections that follow, the three components of the mixture will be visualized as red, blue and green when depicted. These three components are sometimes referred as to polymers (red and blue) and ethanol (green). It should be pointed out that this naming acts mainly to connect to the problem case and to aid in discussion. The different fluid components in the simulations do not possess the *full* physical and thermodynamical properties of their real counterparts. Rather they are meant to reflect the essence of the physics we are trying to capture, as detailed in section 6.

Note: It should be emphasized here that the denotion $\rho(\boldsymbol{x})$ in the LBM context refers to the *number density* at a particular lattice site $\boldsymbol{x}$, and is different to the *molar weight*. Even though two number densities $\rho_A \neq \rho_B$ are not equal, they can still have the same weight per molecule. To obtain the effect of different molar weights, either entries relating to self-interactions in the matrix G are modified, or the molar weight is incorporated into the forcing term.

9.1 Spinodal decomposition

Two cases of spinodal decomposition will be covered. First for a binary immiscible liquid to reflect the composition of the polymer mixture. Then also with a ternary liquid where the third component is a solvent, corresponding to the ethanol, completely miscible with both of the phase segregating solutes.

9.1.1 Binary spinodal decomposition

As a first test, it seems natural to examine the spinodal decomposition in the most simple setting possible. A rectangular domain of dimensions $(N_x,N_y,N_z) = (280,160,160)$ lattice units is initialized with a completely mixed liquid in the ratio $\rho_{A_0} : \rho_{B_0} = 0.3 : 0.7$, with zero velocity everywhere. To initiate the decomposition, a random fluctuation to the composition have to be added to each lattice site. This fluctuation need only be small and is here set to 1%, such that $\rho_A(\boldsymbol{x}) = \rho_{A_0}(1 + \epsilon)$ where $-0.01 \leq \epsilon \leq 0.01$. Subsequently, to avoid pressure fluctuations, the total number density of each lattice site is renormalized so that $\rho_{tot}(\boldsymbol{x}) = \rho_A(\boldsymbol{x}) + \rho_B(\boldsymbol{x}) = 1.0 \ \forall \ \boldsymbol{x}$. Further, the interaction matrix for the two components are set to

$$G = \begin{bmatrix} 1.0 & 3.5 \\ 3.5 & 1.0 \end{bmatrix} \tag{9.1}$$

The relaxation rates are set to

$$s_1 = 1.19, \quad s_2 = 1.4, \quad s_4 = 1.2, \quad s_9 = 1.25, \quad s_{10} = s_2, \quad s_{13} = s_9, \quad s_{16} = 1.5 \tag{9.2}$$

which are then used in the following way for both fluid components

$$\hat{\Lambda} = \mathrm{diag}(1, s_1, s_2, 1, s_4, 1, s_4, 1, s_4, s_9, s_{10}, s_9, s_{10}, s_{13}, s_{13}, s_{13}, s_{16}, s_{16}, s_{16}) \tag{9.3}$$

The viscosity can be retrieved from $s_9 = s_{13} = 1.25$, which translates to $\nu = \frac{1}{3}(\frac{1}{s_9} - 0.5) = 0.1$. The particular choice of the other relaxation times are for reasons of symmetry and from linear stability analysis [22]. Finally, boundary conditions are periodic. The simulation results of the decomposition progress can be observed over different points in time as a series of 2D cross-sections (Figure 29) and as 3D structures (Figure 30).

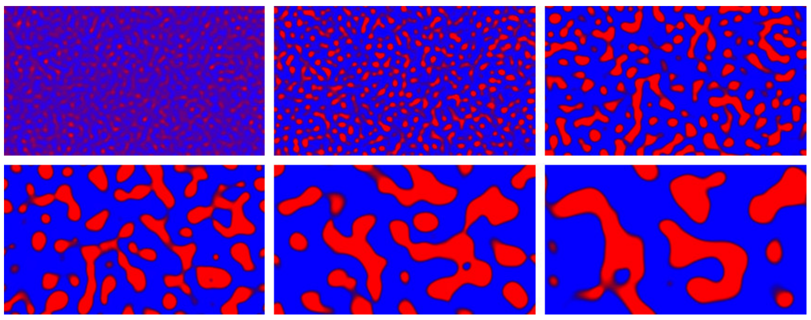

Figure 29: Cross-section ($z = N_z/2$) for spinodal decomposition of binary fluid, taken at the following time steps: 150, 200, 300, 450, 650, 900, from top left to bottom right.

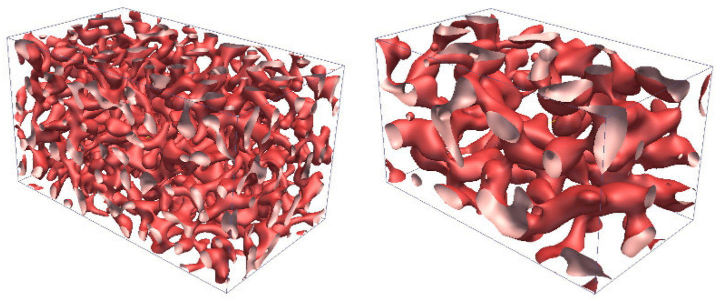

Figure 30: Representation of density iso-surface at $\rho_A = 1.0$ for spinodal decomposition of binary fluid, taken at time steps 450 and 900.

At a certain threshold value for G_{12}, the (pseudo) inter-molecular forces will become large enough to cause phase segregation. While above this threshold, the magnitude of this interaction parameter can be used to control the interface thickness. However, this changes the physical behavior, and set high enough the simulation will start becoming unstable. An alternative way of controlling the interface thickness, without altering the full magnitude of the force, is to carefully adjust the weights of the different neighbor layers when computing the interaction potential. How to determine these weights is described in detail in [39].

The phase segregation process had not reached equilibrium by time step 900, and by then the structures formed are relatively large. Clearly, in order to produce the shapes involved in the real world case, a mechanism to halt the segregation process is needed.

9.1.2 Binary spinodal decomposition in ternary mixture

Similar to the process just described above, a third miscible component will here be added to reproduce spinodal decomposition with a solvent present. The ratio between solutes will be same in respect to each other, with the ratio between solvent and solute being 1 : 1. Normalized to 1, the volume fraction ratio between all components is thus $\rho_A : \rho_B : \rho_C = 0.15 : 0.35 : 0.5$, where A and B refers to the solute (polymers), and C the solvent (ethanol). The interaction matrix G now becomes a 3-by-3 matrix, which in the first attempt is set as similar to the binary case as possible

$$G = \begin{bmatrix} 1.0 & 3.5 & 1.0 \\ 3.5 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix} \tag{9.4}$$

Interestingly, running the simulation with these values, no phase segregation is

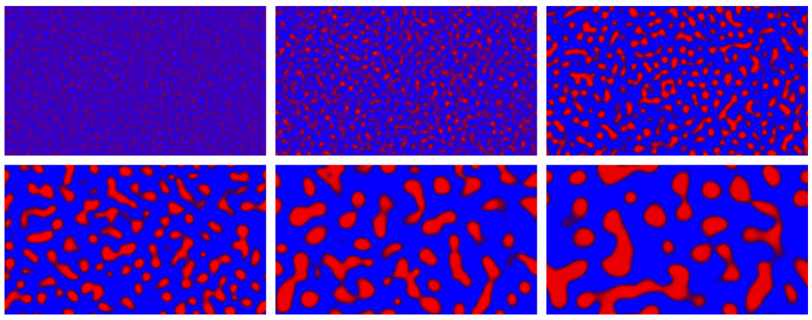

Figure 31: Cross-section ($z = N_z/2$) for spinodal decomposition of two immiscible components immersed in a third, miscible, component (not depicted here), taken at the following time steps: 150, 200, 300, 450, 650, 900, from top left to bottom right.

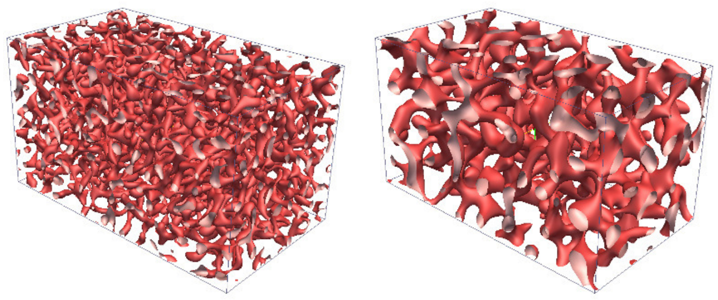

Figure 32: Representation of density iso-surface at $\rho_A = 0.5$ for spinodal decomposition in ternary fluid, at time steps 450 and 900.

present. The A-B interactions have to be increased to $G_{12} = G_{21} = 4.5$ to observe a similar result as with the binary fluid. This means that the presence of a third 'neutral' component naturally influences the miscibility condition of the other two. Cross-sections of the density distribution of component A can be seen in Figure 31. Furthermore, judging from looking at the plot, the segregated domains seems to have a different structure and interconnections. This will most likely affect the permeability of the structure, should it assume a solid form. As a final observation, looking at the concentration of the solvent only (Figure 33), it is clear that this fluid component, though completely miscible, is not evenly distributed in the mixture, but has a slightly lower volume fraction inside the 'red' domains, and forms a denser film around them.

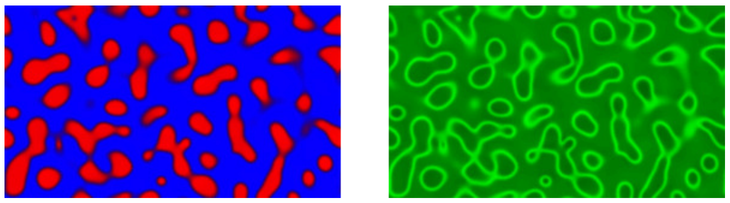
Figure 33: Cross-section ($z = N_z/2$) for ternary mixture. Rightmost image shows the concentration (contrast exaggerated) for the solvent only.

The former phenomenon can be explained by the higher pressure inside the domains, as explained by the Laplace law, making it more difficult for fluid particles to enter this region. The latter is understood by looking at the density distribution across the interface between two liquids (Figure 23 in section 8.4.1). This drop in density (and pressure) will attract a neutral liquid, thus coating the domain.

9.1.3 Ternary spinodal decomposition

Although not needed for the case of the pill coating mixture, a few examples of ternary spinodal decomposition is demonstrated. The purpose is to show that this is easily realized and how it materializes with the proposed model. Also, it should be emphasized that by varying parameters only a small amount, a very different outcome might be the end result. Experimenting within the parameter space might be a useful tool in gaining an intuitive understanding of how properties of the individual components affect the resulting structures. In the large leftmost image in Figure 34, forming a continuous network, the domain was initialized with a completely mixed liquid of component ratios $\rho_A : \rho_B : \rho_C = 0.3 : 0.7 : 0.3$. Viscosity was set to $\nu = 0.5$, and the interaction matrix set to

$$G = \begin{bmatrix} 1.0 & 4.2 & 4.9 \\ 4.2 & 1.0 & 4.2 \\ 4.9 & 4.2 & 1.0 \end{bmatrix} \begin{array}{l} \rho_A = \text{Red} \\ \rho_B = \text{Blue} \\ \rho_C = \text{Green} \end{array} \tag{9.5}$$

For the top right image, where one can see more disconnected structures, the mixture was initialized with the ratios $\rho_A : \rho_B : \rho_C = 0.15 : 0.7 : 0.15$. Viscosity was $\nu = 0.15$, and the interaction matrix

$$G = \begin{bmatrix} 0.3 & 3.8 & 3.8 \\ 3.8 & 0.3 & 3.8 \\ 3.8 & 3.8 & 0.3 \end{bmatrix} \tag{9.6}$$

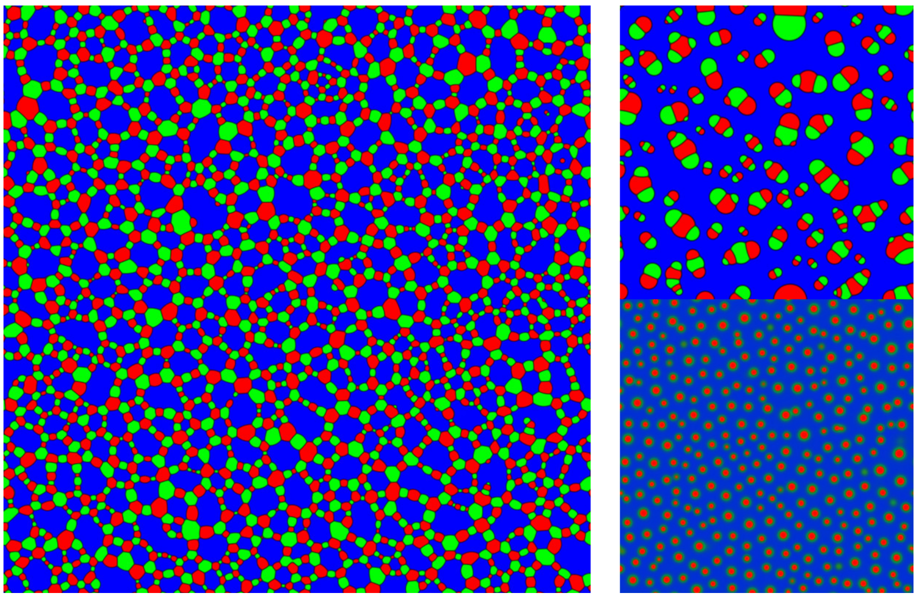
Figure 34: Some examples of ternary spinodal decomposition in 2d. Here ρ_A,ρ_B and ρ_C corresponds to red, blue and green.

Finally, for the bottom right image we see that individual droplets, or cells, have formed, where each cell has a thin coating of another component. To achieve this, the interaction matrix had to be set somewhat more component asymmetric. The mixture was initialized with $\rho_A : \rho_B : \rho_C = 0.15 : 0.7 : 0.3$, the viscosity set to $\nu = 0.5$, with the interaction matrix

$$G = \begin{bmatrix} 0.3 & 3.6 & 2.6 \\ 3.6 & 1.0 & 2.5 \\ 2.6 & 2.5 & 1.0 \end{bmatrix} \tag{9.7}$$

9.2 Solidification

To emulate the gradual solidification of the solute, the Brinkman force term will be employed, following the principles described in section 6.2.3. Effectively, this force term can simply be expressed as $F_b = -c_b \rho \boldsymbol{u}$, where c_b is a factor determining the strength of the force. This force is applied to each component, where the total fluid velocity is used, and is thus written as

$$\boldsymbol{F}_b^\sigma = -c_b \rho_\sigma \boldsymbol{u} \tag{9.8}$$

where $\boldsymbol{u}$ is defined in equation (6.5). In testing this solidification implementation, a Brinkman force is added which is gradually increasing throughout the domain. This scenario is close to the full physical model, where a concentration gradient of ethanol will determine the degree of solidification. Also, the effect of the force is easier to see this way. At the bottom om the domain, where $0 \leq y \leq 0.3N_y$, the force is set to zero, that is $c_b = 0$. The Brinkman force then increases quadratically to reach $c_b = 3.5$ at the top. Walls are introduced at the top and bottom, and periodic boundary conditions are applied at the other domain boundaries. The grid resolution is chosen relatively large at $(N_x,N_y,N_z) = (240, 480, 120)$ as to be able

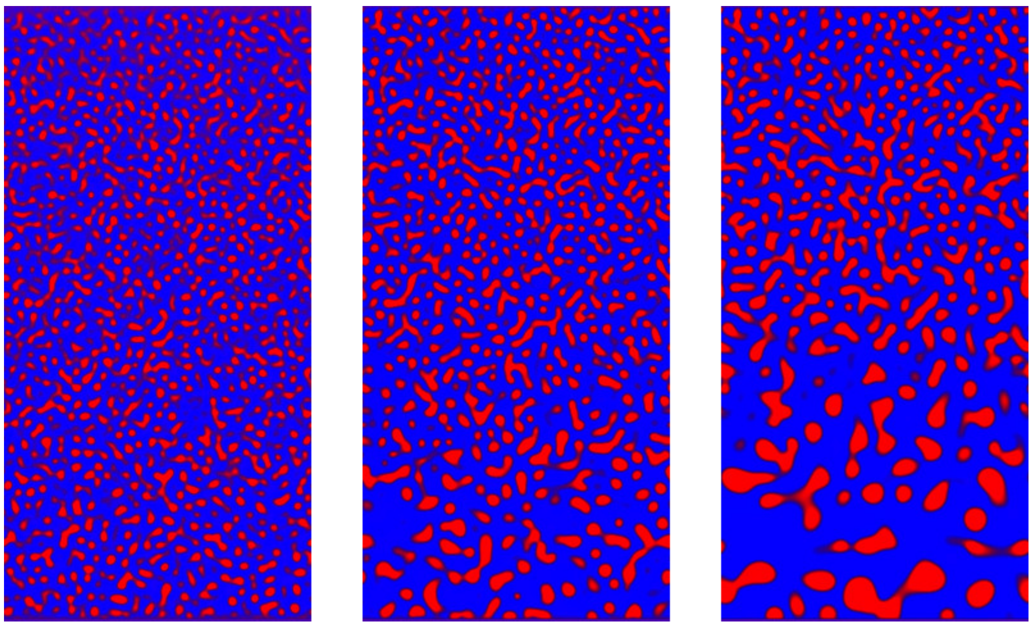

Figure 35: Cross-section of segregation process, subject to a spatially variable Brinkman force. Time steps 300, 400, and 700 are shown.

resolve a large span in segregated domain sizes. All remaining parameters are set as in section 9.1.1. Evident from the results in Figures 35 and 36, the introduced

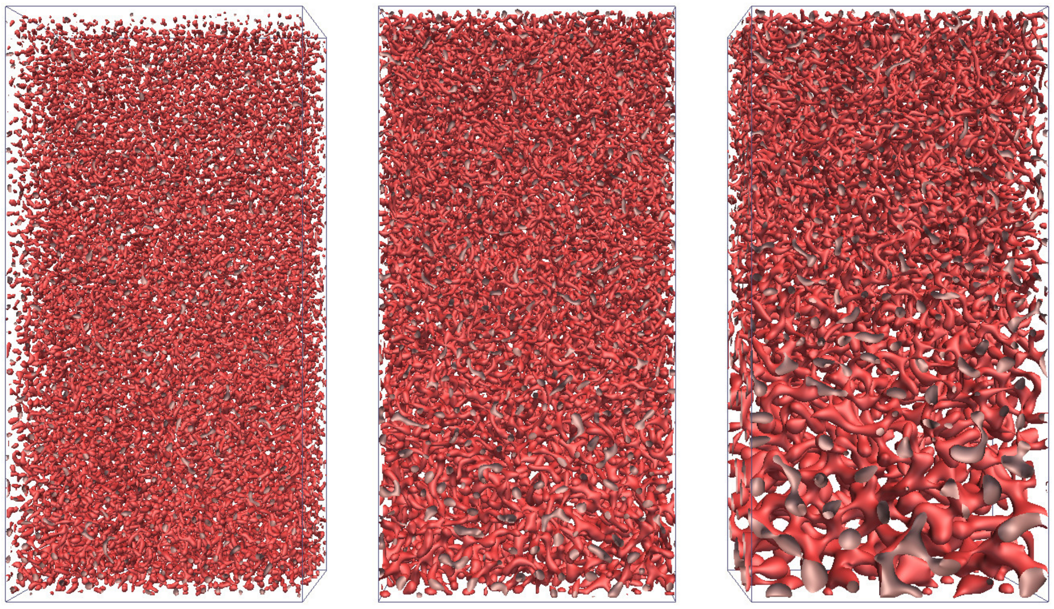

Figure 36: Isosurfaces of $\rho_A = 1.0$ for the same structures as in Figure 35.

force does indeed manage to slow down the flow, and is able to reproduce an effect resembling a variable degree of solidification. The effect is quite strong even for small values of c_b, as can be seen by the sudden transition from larger structures to smaller as we move along y. However, while nearly completely still, it is unable to stop the flow fully. Setting c_b much higher than 3.5 with these parameter settings risks destabilizing the simulation. Instead, to achieve full solidification, when the ethanol concentration drops below a certain threshold value, corresponding to a high value of c_b, the node should be defined as a solid node in the LBM sense. It will now be taken out of the propagation loop and treated with the bounce back boundary condition. Apart from expanding the array of solid nodes, no extra programmatical treatment should be needed. The distributions will remain present and constant in memory, despite residing on a lattice node being redefined as solid.

As a final comment, judging by the resulting structures, there is probably no point in using a higher value than $c_b \approx 2$ for the Brinkman force (for similar parameter settings). There is very little difference in resulting structure above this value, but the stability decreases significantly for $c_b \gtrsim 3$.

9.3 Evaporation and shrinking

In this section, the problem of removing one component by means of evaporation, and the subsequent reduction in size of the remaining liquid, will be examined. While the partial removal of a fluid component is easy in principle, simply removing some volume fraction in a container full with liquid or gas will not do much. There will be a temporary drop in number density, and thus pressure, where the particle populations were removed, and this disturbance will soon be evened out as the system relaxes to its new equilibrium state. But obviously no reduction in volume will take place since there is no accounting physical mechanism. Moreover, the removal of fluid species must be done in a manner consistent with that of evaporation through a given boundary.

One solution is to set up a system with a gas phase and a liquid phase. The particles removed from the liquid by evaporation will be added to the gas phase in such a way as to maintain constant pressure. This will however introduce a fourth component (gas phase) and the need for buoyancy forces, thus resulting in a computationally demanding model.

A more computationally efficient and potentially less complex setup is to implement the gas-fluid interface as a solid wall at the top. As some liquid volume is evaporated, the wall is moved correspondingly in such a way as to maintain constant pressure inside the fluid domain. This alleviates the need for a fourth gas-phase component and buoyancy forces. Furthermore, as the wall closes in on the liquid, there is no need to compute the domain above the wall, hence the computational burden per time step will decrease as the liquid domain shrinks. However, as this wall will move slowly and at a varying rate, functionality need to be added for a more general solid-fluid interface which does not necessarily coincide with the lattice nodes. Moreover, the moving wall will have to exert a force on the fluid to maintain momentum balance and mass conservation. Both these issues have been treated by Lallemand & Luo, and have been implemented here according to their work in [24], as outlined in section 6.2.1.

9.3.1 Evaporation mechanisms

One way to model the evaporation is to set the boundary condition for the ethanol component alone to a fixed velocity, pointing out of the simulation domain. This will cause a material outflow across the boundary and naturally form the concentration gradient observed in the real world case. The shape of the gradient will depend on the magnitude of the prescribed velocity at the boundary. The velocity magnitude is in turn related to the rate of evaporation. Zou & He have derived boundary conditions for the D2Q9 and D3Q15 lattices [52], while an extension to the D3Q19 lattice on the basis of the same principles is provided by Hecht & Harting in [16].

A somewhat less physically accurate but flexible solution is to initialize the ethanol with a prescribed concentration gradient. The ethanol is then removed throughout the entire domain in such a way as to maintain the concentration gradient. This can be done by simply multiplying the populations of each lattice site with a factor less than unity. As the evaporation process can take a relatively large number of time steps, this provides an efficient means of scaling the evaporation time. By using this approach, one adopts the following two assumptions: first, the currents produced by the evaporation and diffusion process are small enough to be rendered negligible. Second, the rate of evaporation is related to the diffusion process in such a way as to produce the prescribed concentration gradient.

Due to time constraints the chosen evaporation process is that of the latter. The concentration gradient is chosen to be linear. To see the effects of the moving wall on the segregation process, phase separating forces have been applied. However, no solidification is performed here, and the simulations are only in 2d in order to work out a methodology. The mixture is initialized with component ratios $\rho_A : \rho_B : \rho_C = 0.3 : 0.7 : 1.0$, i.e. 50% polymer mixture and 50% ethanol, where the green component indicates the ethanol. The result of the simulation at various time steps can be seen in Figure 37. In Figure 38 is a plot of total mass for the different

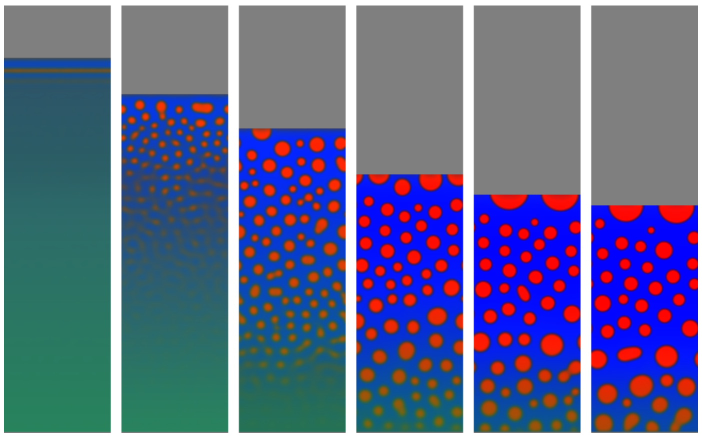

Figure 37: Time evolution of shrinking the liquid through evaporation and a moving boundary (top). The wall exerts a force on the liquid to maintain momentum balance and to preserve mass. The time steps displayed are 600, 1,200, 2,000, 4,000, 6,000 and 8,000.

components. There is a slight increase in mass for the red and blue components when these should be conserved exactly. This issue is related to the moving wall, but how is unclear. The mass balance can however be adjusted by modifying the magnitude of the force exerted by the wall on the fluid.

Even though there is a small inconsistency in mass conservation, the 3-component

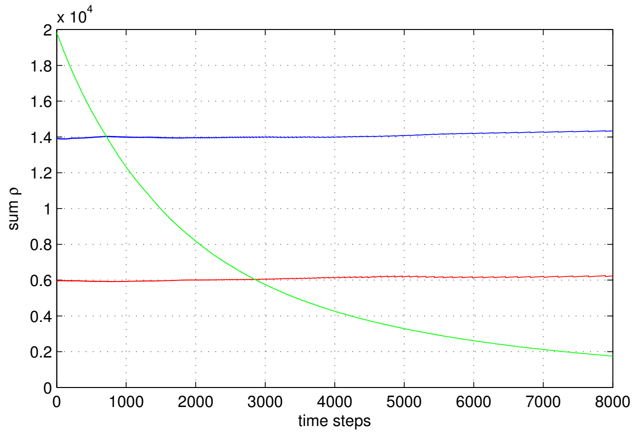

Figure 38: Total mass as a function of time for the process displayed in Figure 37. Mass conservation is not completely consistent for the red and blue components.

approach with a moving solid boundary will be chosen in modeling the full case. It is less complex than the 4-component model, and dealing with only 3 liquids instead of 4 reduces computational time greatly, as the algorithmical complexity scales as $O(N^3)$ with the number of components N. Further, there will not be any need for buoyancy forces, and finally, as an added bonus, the domain above the solid-fluid interface does not have to be computed, further easing the computational burden to some extent. For the evaporation process, the freely scalable model is chosen as to cut down on simulation times to allow for a faster turnaround in experimenting. The fixed velocity boundary condition of course remains the more physically correct of the two.

9.4 Full case

Finally, as all separate methods have been verified to a reasonable working order and a methodology for each have been found, it is time to combine all the different phenomena in an attempt to reproduce the process of the drying pill coating mixture.

To recapitulate, this mixture is initialized with a certain volume fraction of ethanol and a completely mixed polymer blend. The ratio between the two polymers are $\rho_A : \rho_B = 0.3 : 0.7$ with a small random fluctuation in order to initiate spinodal decomposition. Ethanol is evaporated by removing populations throughout the entire container in order to reduce simulation time. However, it is initialized with a linear concentration gradient, and removed in such a way that this gradient remains linear. This approach assumes that the currents induced by the ethanol transport to the outflow boundary are negligible. Simultaneously, a wall at the top of the container is pushing down on the liquid at the same rate as the ethanol is evaporated, such that the pressure is maintained in the mixture. The purpose of this wall is to mimic a free surface without having to introduce a fourth fluid component in gas phase, with all the complications that this would bring. Furthermore, the concentration of the ethanol is coupled to a force which has the effect of slowing down the flow. As the ethanol evaporates, this force becomes stronger until eventually the fluid assumes the appearance of a near solid. In all figures, the ethanol is represented with green, the 30% polymer with red, and the 70% polymer with blue.

Although it is known that the mixture is started with ~85% ethanol, it is assumed here that the mixture enters the unstable region of spinodal decomposition at a much lower concentration. As such, the simulated mixture will be initialized with a concentration near the spinodal in order to avoid unnecessary simulation time. This concentration is at present unknown, but a few different values will be tested. In general, it proved quite difficult to find values which reproduced the sought behavior. Following is a discussion of a few simulation runs.

In the first case (*case A*), it was assumed that the decomposition process starts at an ethanol concentration of ~45%, and the mixture was thus initialized accordingly. The 'solidification force' is introduced when the ethanol concentration reaches below 15% and is linearly increased with decreasing concentration all the way down to 0% ethanol when $c_{b,max} = 0.5$. A series of cross-section snapshots from the simulation is shown in Figure 39, and the final time step represented in 3d can be seen in Figure 42. While it can be concluded that the mixture does very nearly solidify at

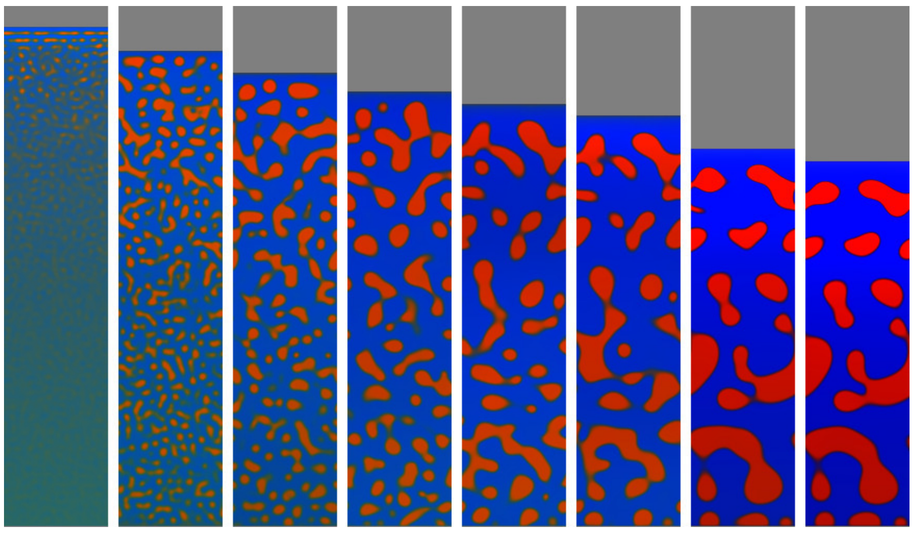

Figure 39: Cross-sections of the drying process for *case A*, with the time steps shown: 250, 450, 650, 850, 1050, 1250, 2250, and 3250. Grid resolution is (N_x,N_y,N_z) = (100,500,100) lattice units.

the end, as there is not much difference between the two last images with 1000 time steps in between, it can easily be seen that the resulting structures are much too large. The segregating process is allowed to run for quite long until the solidification process starts, resulting in a strong coarsening of the domains. However there is an interesting effect as a result of the shrinking; the topmost segment has become compressed with somewhat smaller domains as a result. Conversely, the separation has continued for longer in the bottom part, resulting in somewhat larger domain structures in this region. These effects are much more apparent when watching the process in motion.

Also worth noting is that the top part starts separating first, as a result of the lower concentration of ethanol. Thus these domains initially grow larger than the ones on the bottom. As the process continues, the upper part solidifies while the domains can continue to grow in the bottom half. At the end in the near solid structure, the domains are larger in the bottom than on the top, showing evidence there is some interesting dynamics involved in domain size growth, governed by the distribution and evaporation of the ethanol.

Based on the results in case A, the mixture in the next simulation (*case B*) is initiated with only 20% ethanol. The motivation being that this should give the liquid less time to segregate before solidifying, thus forming a finer structure. Naturally, this also assumes the segregating process starts at the prescribed initial ethanol concentration. Moreover, the maximum solidifying force is set higher at $c_{b,max} = 1.5$. The results can be seen in Figures 40 and 42. This time a somewhat

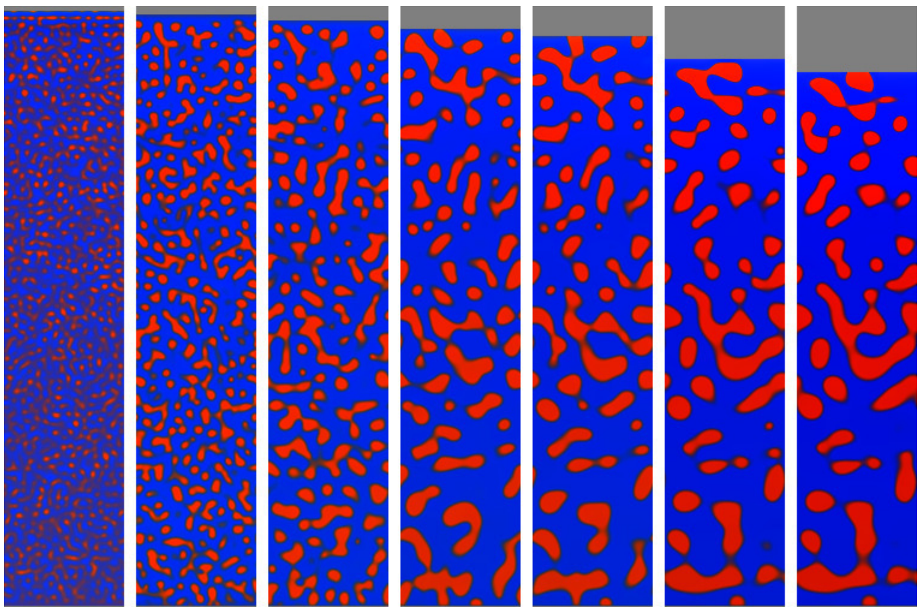

Figure 40: Cross-sections of the drying process for *case B*, with the time steps shown: 150, 250, 350, 550, 750, 1500, and 2150. Grid resolution is (N_x,N_y,N_z) = (100,500,100) lattice units.

finer structure is obtained. Since the total ethanol content was less in this simulation, the amount of shrinking is naturally less. Although more detailed, similar characteristics can be seen here as in case A. It is less obvious in the cross-sections, but a more obvious pattern emerges in the 3d representation in Figure 42. Here it is clear that a varying composition in domain sizes has been achieved, with larger structures in the bottom, and gradually becoming finer when moving towards the top. Still, a somewhat finer structure overall is desired. Since these simulations are so far dimensionless, it could be a matter of upscaling the process onto a larger grid, allowing for a finer detail. However, in order to keep computational resources to a minimum, attempts should be made to gain the most detail out of any given resolution.

In an effort to produce a more fine grained domain structure, the spinodal decomposition process is slowed down by applying the solidifying Brinkman force across the whole simulation grid. An additional solidifying force is then applied on top to achieve full solidification. The idea is that the structures should now have less effective time to coarsen, or do so at a slower rate. For this *case C*, the global Brinkman force applied has a magnitude of $c_{b,global} = 0.5$. Also, the resolution is somewhat increased to reduce the effect of the periodic boundary conditions, and to retrieve finer detail. The remaining parameters are unchanged from case B. A few images from the resulting simulation is shown in Figure 41. This approach manages

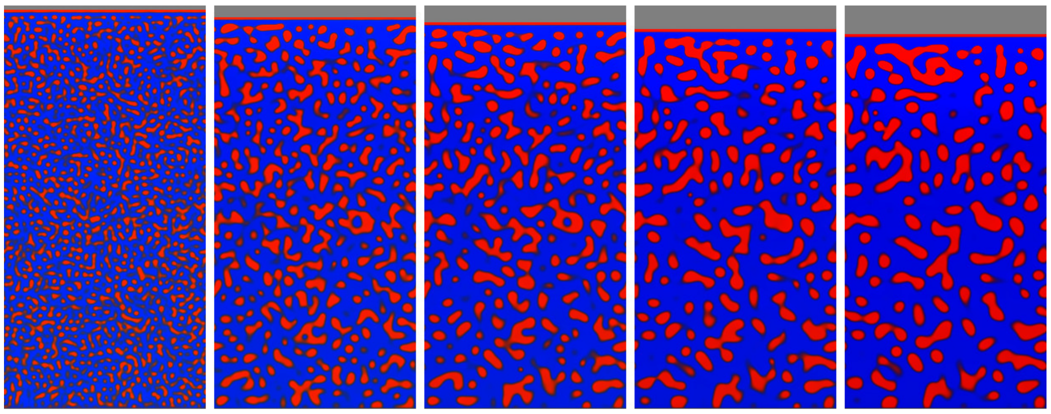

Figure 41: Cross-sections of the drying process for *case C*, with the time steps shown: 250, 750, 1250, 2250, and 3750. Grid resolution is $(N_x,N_y,N_z) = (240,480,120)$ lattice units.

to produce finer structures overall, as is most evident from the 3d-representation in Figure 42. In this image, a subsection of the full simulation domain is taken as to make a fair comparison with the previous simulations. However, there is not much variation in the resulting structures, indicating the dynamics previously seen as a result of the ethanol has less of an impact.

As a concluding remark of this section, it seems as if complex dynamics can indeed be incorporated into the model, with resulting rich behavior. Finding the right parameters to achieve the desired result, however, can be difficult and time consuming. Each simulation run with a decent resolution takes hours on a normal desktop machine. To gain the most understanding of the dynamics for each simulation run, a large portion of the time steps should be saved and watched in real time playback once the simulation is completed.

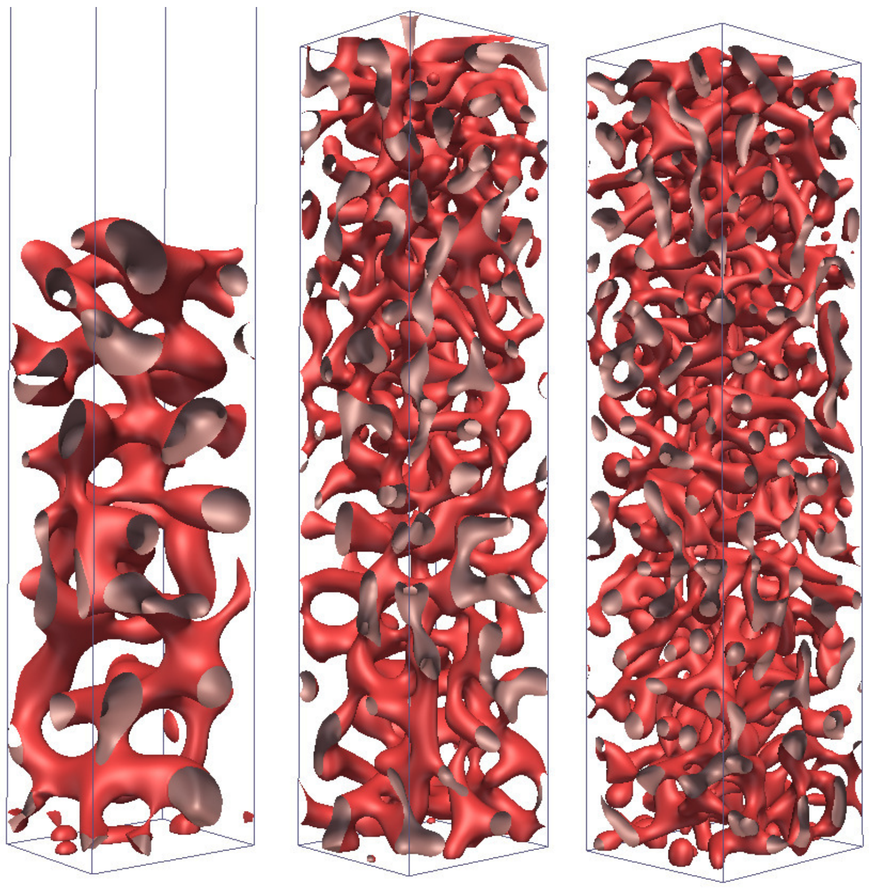

Figure 42: Iso-surface of the final solidified 3d-structure of the 'red' component for *case A* (left), *case B* (middle), and *case C* (right).

10 Discussion

In analyzing and evaluating the work done up to this point, there are a lot of aspects to consider. In beginning is a closer look at the more technical aspects of the implementation. Subsequently, discussion will turn to the proposed LBM model and simulation results.

Programming

The software written for this thesis was not meant to be a complete package by any means. Rather, the main purpose of doing it from the ground up was the great flexibility to modify the functionality that comes with such an approach. Improvements and additions could readily be added without being subject to any possible limitations to an API (Application Programmer Interface), making it very valuable and efficient in trying many different models. But still, in order to produce results within a reasonable time frame, performance must be fairly optimized. Efforts were made to develop a memory model which would be efficient regarding cache locality, memory locality and multithreading. The subgrid model with contiguous memory layout satisfies all the aspects mentioned above, while also allowing for convenient distribution of the workload for cluster computing. Moreover, it provides for linear scaling with problem size, meaning a larger problem will not lead to a degraded specific performance. Memory usage with this model is also greatly reduced, limited to one extra subgrid, when compared to the dual grid method. Furthermore, this particular memory layout makes possible the devised 'shuffle' time stepping model, which cuts the memory accesses in half while retaining efficient use of memory. When applied to the GPU implementation, performance almost doubled. However, the CPU with its more sophisticated cache is not as sensitive to memory access as a GPU device. For the single component D2Q9 model, the performance measured at 170 MLUPS, up from the previous 130 MLUPS with 30%. A downside with the particular choice of memory model is that the communication between subgrids in the streaming step is somewhat more convoluted, as the mapping from global coordinates to memory index is not as straight forward as with a regular memory model. This can however be easily be circumvented with a wrapper function to compute the destination address. Overall, the advantages with this memory model seem to clearly outweigh the disadvantages.

GPU results

The GPU implementation clearly demonstrated that the LBM does very well on these types of devices, owing to its inherent parallelism. With the full implementation achieving some 70% of peak theoretical performance of data transfers alone, coupled with efficient memory usage, the implemented algorithm can be considered efficient. Although GPU devices are generally limited in memory, this approach could be very interesting for interactive feedback simulation during initial stages of modeling, using low resolution models. A GPU implementation of a multicomponent model would certainly be beneficial, considering these are much slower than their single component counterpart. Also worth highlighting is that, generally, the results indicate that the more computationally intensive the model is, the greater the speedup over the CPU implementation.

Object oriented programming

The object oriented approach has worked fairly well. Although the use of difficult to read code and structures was sometimes needed in order to maintain performance. Sadly, adding functionality the most user friendly and intuitive way will most certainly lead to notable performance degradations. Improvements are needed in adding the force into the collision model. At the moment a new collision routine have to be defined in order to add the force in an efficient way. Generally, a more efficient way of adding functionality without sacrificing performance can likely be added through the use of C++ templates and metaprogramming (code generating code).

Overall, the choice to write the code from scratch has been valuable in many aspects. Most notably the possibility to add any kind of functionality, but also in gaining a deeper understanding of the LBM and its inherent issues, and finally the ease to port the results to other hardware solutions such as GPU devices.

On the proposed model

The suggested multicomponent model is based on the popular Shan-Chen model. Its appeal lies in the ease of which the intercomponent interaction is achieved. Rather than solving the Cahn-Hilliard equation in addition to the regular fluid, as in the free energy methods, phase separation occurs naturally as a result of an effective repulsive potential, in turn constructed by intuitive 'inter-molecular' forces. However, as of yet there seems to be little or no mathematical treatment to show the correctness of this approach. Instead one has to rely on empirical proof through validation of benchmarks from theory and experiment. The initial version of the Shan-Chen model was associated with some thermodynamically inconsistent behavior and relatively large numerical instabilities. In the model suggested here, these problems should be substantially reduced. Much of the problems with the Shan-Chen model seem to stem from the special incorporation of the force into the model, an approach which is abandoned here. Further, the lack of isotropy in the interaction force is addressed through an increased stencil. Finally, added numerical stability is added through the use of the MRT collision term. The motivation for the inclusion of these particular improvements came from evidence in literature [28,39,51], and the benchmark tests performed here seem to generally validate these findings. Evidence of physical correctness can be witnessed in several benchmarks. The interface profile shows great qualitative agreement (Figure 23), the pressure drop across the interface follows the Laplace law for all but the smallest drops where some deviation can be seen (Figure 24). The latter issue is most likely due to increased discrete lattice effects as resolution breaks down, rather than a problem with the model. Moreover, the measured phase diagram show excellent qualitative behavior with the regular solution model (Figure 22). So far only qualitative results have been shown, as no physical units have been applied. Since no thermal effects occur in the model, meta-stability and nucleation are not reproduced.

Regarding the spurious velocities, comparing between the different benchmarked models, it is clear that the He forcing scheme is preferential over the Shan-Chen forcing, and it is equally clear that the MRT collision term provides a further significant improvement. Using identical interaction parameters, the MRT model with 10th order isotropy interaction force offers a reduction of the spurious velocities with two orders of magnitude, when compared to the original Shan-Chen model. Also, considering that the additional computation time for the MRT model becomes almost insignificant when compared with a large inter-component force stencil, this collision term is obviously preferential over the BGK model. The only reason to choose the BGK collision term would be for an more straight forward implementation.

While intuitive to use, a drawback with the proposed model is the computational burden. As a result of the additional sets of populations for each fluid together with the interaction matrix, the algorithmic complexity is of order $O(N^3)$ with the number of fluid components N. Here the free energy model instead uses a reduced set of additional populations (usually D3Q7) to solve the Cahn-Hilliard equation. While the LBM is in general regarded as computationally efficient, other multicomponent models, such as the free energy models, might offer a further reduced computational burden when compared to the proposed model.

Simulation results

Working with the model, as in adding functionality and performing subsequent tests, was generally a very positive experience. Introducing additional liquids, and defining the miscibility relations, is a simple matter of changing a variable and extending the N by N interaction matrix. Adding the solidifying force as a function of ethanol concentration was a few lines of code. Since all the concentrations (number densities) are precomputed for the whole grid, introducing additional complex effects depending on these is simple. Consequently, getting the desired results when independently testing the necessary physical phenomena was straight forward. Although, admittedly, the 4 component evaporation setup was problematic in the way that large pressure fluctuation arose as a result of an ever-so-slight departure from equilibrium in the initial condition. The difficulty lies in setting the precise distribution across the interfaces between the components, as a function of density, interface curvature, interaction parameters, etc. This is near impossible, and the resulting pressure waves takes long time ($O(10^4)$ time steps in this case) to subdue, affect the results considerably, and pose a real problem. One solution could be to start the simulation with the liquids slightly mixed, while gradually increasing interaction parameters to let the liquid assume equilibrium smoothly. However, by introducing a moving wall, which could be arbitrarily placed between lattice nodes, acting as the free surface, this scenario could be circumvented altogether. This approach worked well and was therefore adopted, although it did show a small inconsistency in mass conservation. This might originate from the forcing term from the wall not being incorporated according to the He scheme.

One fundamental difficulty is the slow nature of evaporation, combined with the small time step for the LBM. A solution employed here in order to cut down simulation times is to artificially remove ethanol throughout the entire volume in a consistent manner. This works relatively well and effectively scales the process in time. However, some detail might be lost in the process, such as fluid transport effects. One must be careful not to scale this process too much, as pressure fluctuations will arise. Letting the evaporation occur over more than 1000 time steps seems to work well. However, the more physically and dynamically correct treatment of a fixed velocity boundary condition to cause material outflow is preferable, given that time constraints is not an issue.

Combining all contributing phenomena in an *attempt* to model the full case was an easy task, while getting the desired end result was not. Quite simply, the parameter space is huge and finding the right window is difficult. A task made even more challenging as many of the parameters are unknown even in the real world case. Experimenting in any decent resolution in 3D is time-consuming on a simple desktop computer, and it is difficult getting a 'feel' for how the simulation responds to different input. Key in getting a physically correct result seems to be finding the right connection between solidification and ethanol concentration. At the moment this is a linear map, and solidification is a simple effect of slowing down the flow through a force opposing the flow direction, while the actual solidification process is no doubt much more intricate than that. A non-newtonian collision model would be a good place to start, in an attempt to capture a more reptation like behavior in the semi-solid state. Such models have been devised, for example by Yoshino *et al.* [47], but was not implemented here due to time restrictions. Apart from this, the problems encountered was less connected to the LBM model (or any numerical model), but more to the uncertainty of the physical processes and parameters involved.

These complications aside, some interesting dynamics were captured. Reflecting the real world case, varying sizes in domain structure was accomplished. Also, because of the gradient of the miscible component ('green') and its successive evaporation, there was an observed shift, or reversal, in domain growth patterns. Initially, structures grew larger in places of low concentrations of the miscible component (the top), however as these solidified soon after, the structures at the bottom were allowed to continue their growth until surpassing the domains at the top in size. Moreover, since the top was nearly solid as the miscible component continued to evaporate, it tended to almost collapse in on itself, resulting in a somewhat compressed elongated domain appearance.

11 Conclusions

After having worked extensively with the LBM, both in terms of technical implementation and model studies, it becomes obvious that this is still a model under development, especially with multicomponent flows and issues related to thermodynamical behavior. However, plenty of options exist and new research is frequently being published. Creating a basic implementation capable of relatively complex behavior is very straight forward. Incorporating more sophisticated functionality, necessary for many real world cases where greater demands on accuracy and stability need to be satisfied, requires more initial work, but the obtained improvements are often significant.

The proposed model is showing much improved behavior over a 'naive' version, although requiring an increased implementation effort and comes with a higher computational cost, especially for multicomponent flows. This calls for a highly optimized implementation, both algorithmically and programmatically. The calculation of the interaction potential bears the largest computational cost, so optimization efforts are extra important here. A speedup of a factor $\sim$2 should be attainable by reusing force calculations between components. The devised memory model, domain decomposition with contiguous memory, seems to work well with a multithreaded approach and scales well with problem size, as well as bringing other advantages. It is difficult to evaluate the implementation when comparing to results found in literature, since no cases where similar hardware is used have been found. Although one can use a maximum bandwidth measure for a good estimate on efficiency for single component models. Both CPU and GPU results indicates an efficient implementation with this measure, especially for 2d models, while 3d models can likely be improved some. Porting the LBM to a GPU device gives significant performance increases, here with a factor of $\sim 10 - 15$ for the single component models, and $\sim 20 - 35$ for 2D multicomponent models, and should be considered if interactivity is desired. A GPU version could prove to be very advantageous for multicomponent flows, albeit limited to smaller resolutions.

Implementing a multicomponent solver with the LBM through the interaction potential model, and the use thereof, was intuitive and straight forward. The MRT collision term gives excellent numerical stability and reduced spurious velocities when compared to the BGK collision term. The He forcing scheme definitely seems to be an improvement over a basic implementation, when comparing spurious velocities with the Shan-Chen forcing. The extended stencil, i.e. the mid-range interaction potential, for computing the interaction force reduces spurious velocities and stability further, but is very expensive. Careful consideration should be taken when employing this. The largest stencil might not always be necessary.

Since there is little theoretical grounds established for the interaction potential model, proof of correctness have to come from empirical studies. The benchmarks performed in this work all show good evidence of physical validity, as shown through the phase diagram, interface profile, and the Laplace law. Moreover, adding new functionality to account for various physical phenomena proved to be relatively easy. Although satisfactory to some extent, the solidification process is currently limited to a semi-solid state and does not reproduce the sought after non-newtonian behavior most likely exhibited in the pill coating mixture. Here a more sophisticated model is needed, such as the one developed by Yoshino *et al.* [47]. The emulation of a gradually receding free surface by a moving wall worked satisfactorily, but a slight inconsistency in mass conservation was observed.

Combining all physical phenomena was technically easy, but controlling the end result was difficult. This is mainly a result of a very large parameter space, where many aspects are unknown even in the real world case. Consequently, the final modeling process required a lot of guess work, and feedback was limited due to long simulation times. Hence, mainly as a result of the many unknowns, the reproduction of the pill coating drying process was not completely achieved to the desired degree. However, some interesting dynamics was displayed, representative with observed actual processes, and the outlook to model this case with the LBM by improving e.g. the solidification process remains positive. Although, importantly, much more information regarding the physical process needs to be known. Finally, the free energy model should be considered for any future attempts, although a 3-component model might have to be developed as no examples of this have been found in literature.

12 Future work

As discussed throughout the text, a number of aspects could be improved upon or extended. Following is a summary of the most relevant issues that should be looked at, if attempting to carry this work further.

Since the solidification process is such a central aspect, other models should be considered in favor of the Brinkman force, such as a non-newtonian model. The current solution does produce an effect similar to that of solidification, but as the semi-solid state is reached the governing physics of a reptation like phenomenon is not captured.

On a more fundamental level, the class of free energy models should be evaluated as a possible alternative to the interaction potential model. An interesting prospect is to develop a 3-component or even a general n-component free energy model. This will likely cut computation time to some extent.

If staying with the present model, some performance tuning is needed. Improvements can be made by optimizing the mean field inter-component force, a very expensive calculation at the moment. Although, most interesting from a performance point of view would be a multi-GPU implementation of the 3D multicomponent model, interaction potential model or other. With a multi-GPU setup, enough memory storage would be available to contain a high resolution 3D simulation. Further, the 2D multicomponent model on a single GPU gave enough performance increase to provide for an interactive experience, so naturally a similar speedup is expected for 3D models. Though an interactive response would be much to ask for in the case of a 3D simulation, drastically cut simulation times would be very beneficial for the time-consuming process of experimenting with parameters.

Of a more theoretical nature, there would likely be interest from the LBM community on a more rigorous treatment of the interaction potential model. So far reliance rests mainly on phenomenological grounds. Showing analytically how this model satisfies the governing equations, and how it connects to the regular solution model, would be beneficial.

13 Summary

A comprehensive study of the lattice Boltzmann method has been made, focusing on multicomponent flows and spinodal decomposition. The interaction potential model, first introduced by Shan and Chen, was employed as the mechanism responsible for phase separation. This particular choice was motivation by its ease of use and intuitive approach, as well as straight forward extension to any number of fluid components. As a response to known inconsistencies with the original Shan-Chen model, several improvements were made based mainly on recent research. The most significant modifications were a more accurate implementation of body forces, increased potential range, and the use of the Multiple-Relaxation-Time (MRT) collision term. This model was implemented from the ground up in multithreaded C++ and a few preliminary attempts were made with GPU computing, resulting in significant performance gains. Benchmarks demonstrated good signs of physical correctness, as evidenced by the phase diagram, interface profile, and Laplace law. Unphysical side effects were notably reduced as a result of mentioned improvements. In order to model a real world case of a ternary mixture, simultaneously shrinking, drying and phase-segregating, some physical phenomena such as solidification and evaporation had to be introduced. Due to many unknown factors regarding the physical process, a large number of tunable parameters, and lack of feedback because of long simulation times, the formation of the pill-coating structure was not fully replicated. However, the resulting simulations did show some spontaneously occurring complex dynamics in agreement with observations of the real process. The interaction potential approach, combined with the extended interaction range, turned out to be computationally demanding, in context of the LBM. Therefore, other mechanisms of phase separation could potentially prove a better choice. All things considered, with careful choice of properties the lattice Boltzmann method remains a very interesting and promising approach for multicomponent flows.

Acknowledgements

I would like to extend my sincere gratitude to my very knowledgeable advisor and examiner Alexey Geynts, who, together with Astra-Zeneca, provided me with the opportunity to work with the lattice Boltzmann method for this thesis. Also very much appreciated are the, sometimes very long, discussions about how to tackle all sorts of problems, and expert guidance in the jungle that research literature sometimes can be. At times when feeling generally lost and out of ideas, I was always put back on track again with some well put advice.

Appendix

A MRT details

Here some detail regarding the MRT collision term is given. The collision/relaxation process occur in moment space. To transform the distribution functions to moment space, form a column vector f and multiply with the moment matrix according to $m = Mf$. Prior to the streaming step, the post-collision moments are transformed to velocity space by $f = M^{-1}m$.

A.1 D2Q9

The moment matrix for the D2Q9 model is defined as

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{A.1}$$

The equilibrium functions in moment space are given by

$$m^{eq} = \begin{bmatrix} m_0^{eq} \\ m_1^{eq} \\ m_2^{eq} \\ m_3^{eq} \\ m_4^{eq} \\ m_5^{eq} \\ m_6^{eq} \\ m_7^{eq} \\ m_8^{eq} \end{bmatrix} = \begin{bmatrix} \rho \\ -2\rho + 3(\boldsymbol{j} \cdot \boldsymbol{j})/\rho_0 \\ \rho - 3(\boldsymbol{j} \cdot \boldsymbol{j})/\rho_0 \\ j_x \\ -j_x \\ j_y \\ -j_y \\ (j_x^2 - j_y^2)/\rho_0 \\ j_x j_y/\rho_0 \end{bmatrix} \tag{A.2}$$

A.2 D3Q19

The moment matrix for the D3Q19 model is defined as $M =$

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-30 & -11 & -11 & -11 & -11 & -11 & -11 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\
12 & -4 & -4 & -4 & -4 & -4 & -4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & -4 & 4 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & 0 & 0 & -4 & 4 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & -4 & 4 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
0 & 2 & 2 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2 \\
0 & -4 & -4 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2 \\
0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & -2 & 2 & 2 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1
\end{bmatrix} \tag{A.3}$$

The equilibrium functions in moment space are given by

$$m^{eq} = \begin{bmatrix} m_0^{eq} \\ m_1^{eq} \\ m_2^{eq} \\ m_3^{eq} \\ m_4^{eq} \\ m_5^{eq} \\ m_6^{eq} \\ m_7^{eq} \\ m_8^{eq} \\ m_9^{eq} \\ m_{10}^{eq} \\ m_{11}^{eq} \\ m_{12}^{eq} \\ m_{13}^{eq} \\ m_{14}^{eq} \\ m_{15}^{eq} \\ m_{16}^{eq} \\ m_{17}^{eq} \\ m_{18}^{eq} \end{bmatrix} = \begin{bmatrix} \rho \\ -11\rho + 19(\boldsymbol{j}\cdot\boldsymbol{j})/\rho_0 \\ w_\varepsilon \rho + w_{\varepsilon j}(\boldsymbol{j}\cdot\boldsymbol{j})/\rho_0 \\ j_x \\ -\frac{2}{3}j_x \\ j_y \\ -\frac{2}{3}j_y \\ j_z \\ -\frac{2}{3}j_z \\ (2j_x^2 - j_y^2 - j_z^2)/\rho_0 \\ w_{xx}(2j_x^2 - j_y^2 - j_z^2)/\rho_0 \\ (j_y^2 - j_z^2)/\rho_0 \\ w_{xx}(j_y^2 - j_z^2)/\rho_0 \\ j_x j_y/\rho_0 \\ j_y j_z/\rho_0 \\ j_x j_z/\rho_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.4}$$

where $w_\varepsilon, w_{\varepsilon j}$ and w_{xx} are free parameters to tune. Linear stability analysis has shown [22] that good choices for these are $w_{\varepsilon j} = -475/63$, $w_\varepsilon = w_{xx} = 0$. The freely tunable relaxation rates used in this work are set according to

$$s = (1.0, s_1, s_2, 1.0, s_4, 1.0, s_4, 1.0, s_4, s_9, s_{10}, s_9, s_{10}, s_{13}, s_{13}, s_{13}, s_{16}, s_{16}, s_{16}) \tag{A.5}$$

where

$$s_1 = s_2 = 0.7, \quad s_4 = 1.2, \quad s_{10} = s_2, \quad s_{11} = s_9, \quad s_{13} = s_9, \quad s_{16} = 1.5 \tag{A.6}$$

where s_9 controls the shear viscosity.

References

[1] Ahrenholz, B. Massively parallel simulations of multiphase- and multicomponent flows using lattice boltzmann methods. *Science*, April (2009), 148.

[2] Balfour, J. Introduction to cuda. Available online: http://mc.stanford.edu/cgi-bin/images/f/f7/Darve_cme343_cuda_1.pdf.

[3] Bhatnagar, P. L., Gross, E. P., and Krook, M. A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems. *Phys. Rev. 94* (May 1954), 511–525.

[4] Cao, N., Chen, S., Jin, S., and Martínez, D. Physical symmetry and lattice symmetry in the lattice boltzmann method. *Phys. Rev. E 55* (Jan 1997), R21–R24.

[5] Chen, S., and Doolen, G. D. Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics 30*, 1 (1998), 329–364.

[6] D'Humières, D. Generalized lattice boltzmann equations. *Rarefied Gas Dynamics: Theory and Simulations 159*, 1792 (1992), 450–458.

[7] D'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P., and Luo, L.-S. Multiple-relaxation-time lattice boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society - Series A: Mathematical, Physical and Engineering Sciences 360*, 1792 (2002), 437–451.

[8] Feichtinger, C., Götz, J., Donath, S., Iglberger, K., and Rüde, U. Walberla : Exploiting massively parallel systems for lattice boltzmann simulations. *6th International Conference for Mesoscopic Methods in Engineering and Science* (2009), 241–260.

[9] Frisch, U., Hasslacher, B., and Pomeau, Y. Lattice-gas automata for the navier-stokes equation. *Phys. Rev. Lett. 56* (Apr 1986), 1505–1508.

[10] Ginzburg, I., Verhaeghe, F., and d'Humières, D. Two-relaxation-time lattice boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions. *Communications In Computational Physics 3* (Feb 2008), 427–478.

[11] González-Segredo, N., Nekovee, M., and Coveney, P. V. Three-dimensional lattice-boltzmann simulations of critical spinodal decomposition in binary immiscible fluids. *Phys. Rev. E 67* (Apr 2003), 046304.

[12] Habich, J., Feichtinger, C., Köstler, H., Hager, G., and Wellein, G. Performance engineering for the lattice boltzmann method on gpgpus: Architectural requirements and performance results. *Computers & Fluids* (2012).

[13] He, X., and Luo, L.-S. Lattice boltzmann model for the incompressible navies-stokes equation. *Journal of Statistical Physics 88* (1997), 927–944.

[14] He, X., and Luo, L.-S. A priori derivation of the lattice boltzmann equation. *Phys. Rev. E 55* (Jun 1997), R6333–R6336.

[15] He, X., Shan, X., and Doolen, G. D. Discrete boltzmann equation model for nonideal gases. *Phys. Rev. E 57* (Jan 1998), R13–R16.

[16] Hecht, M., and Harting, J. Implementation of on-site velocity boundary conditions for d3q19 lattice boltzmann simulations. *Journal of Statistical Mechanics: Theory and Experiment 2010*, 01 (2010), P01018.

[17] Higuera, F. J., and Jiménez, J. Boltzmann approach to lattice gas simulations. *EPL (Europhysics Letters) 9*, 7 (1989), 663.

[18] Hou, S., Sterling, J., Chen, S., and Doolen, G. D. A Lattice Boltzmann Subgrid Model for High Reynolds Number Flows. *Contributions to Mineralogy and Petrology* (Jan. 1994), 1004.

[19] Huang, H., Krafczyk, M., and Lu, X. Forcing term in single-phase and shan-chen-type multiphase lattice boltzmann models. *Phys. Rev. E 84* (Oct 2011), 046710.

[20] Huang, H., Zheng, H., Lu, X.-y., and Shu, C. An evaluation of a 3d free-energy-based lattice boltzmann model for multiphase flows with large density ratio. *International Journal for Numerical Methods in Fluids 63*, 10 (2010), 1193–1207.

[21] Junk, M., Klar, A., and Luo, L.-S. Asymptotic analysis of the lattice boltzmann equation. *J. Comput. Phys. 210*, 2 (Dec. 2005), 676–704.

[22] Lallemand, P., and Luo, L.-S. Theory of the lattice boltzmann method: Dispersion, dissipation, isotropy, galilean invariance, and stability. *Phys. Rev. E 61* (Jun 2000), 6546–6562.

[23] Lallemand, P., and Luo, L.-S. Theory of the lattice boltzmann method: Dispersion, dissipation, isotropy, galilean invariance, and stability. *Phys. Rev. E 61* (Jun 2000), 6546–6562.

[24] Lallemand, P., and Luo, L.-S. Lattice boltzmann method for moving boundaries. *Journal of Computational Physics 184*, 2 (2003), 406 – 421.

[25] Lapitski, D., and Dellar, P. J. Convergence of a three-dimensional quantum lattice boltzmann scheme towards solutions of the dirac equation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 369*, 1944 (2011), 2155–2163.

[26] Luo, L.-S. Theory of the lattice boltzmann method: Lattice boltzmann models for nonideal gases. *Phys. Rev. E 62* (Oct 2000), 4982–4996.

[27] Martys, N. S. Improved approximation of the brinkman equation using a lattice boltzmann method. *Physics of Fluids 13*, 6 (2001), 1807–1810.

[28] McCracken, M. E., and Abraham, J. Multiple-relaxation-time lattice-boltzmann model for multiphase flow. *Phys. Rev. E 71* (Mar 2005), 036701.

[29] McNamara, G. R., and Zanetti, G. Use of the boltzmann equation to simulate lattice-gas automata. *Phys. Rev. Lett. 61* (Nov 1988), 2332–2335.

[30] Mei, R., Luo, L.-S., Lallemand, P., and d'Humières, D. Consistent initial conditions for lattice boltzmann simulations. *Computers & Fluids 35*, 8 - 9 (2006), 855 – 862. <ce:title>Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science</ce:title>.

[31] Mendoza, M., and Muñoz, J. D. Three-dimensional lattice boltzmann model for electrodynamics. *Phys. Rev. E 82* (Nov 2010), 056708.

[32] Pan, C., Luo, L.-S., and Miller, C. T. An evaluation of lattice boltzmann schemes for porous medium flow simulation. *Computers & Fluids 35*, 8 - 9 (2006), 898 – 909. Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science.

[33] Pattison, M., Premnath, K., Morley, N., and Abdou, M. Progress in lattice boltzmann methods for magnetohydrodynamic flows relevant to fusion applications. *Fusion Engineering and Design 83*, 4 (2008), 557 – 572.

[34] Pohl, T., Deserno, F., Thurey, N., Rude, U., Lammers, P., Wellein, G., and Zeiser, T. Performance evaluation of parallel large-scale lattice boltzmann applications on three supercomputing architectures. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing* (Washington, DC, USA, 2004), SC '04, IEEE Computer Society, pp. 21–.

[35] Premnath, K. N., Pattison, M. J., and Banerjee, S. Generalized lattice boltzmann equation with forcing term for computation of wall-bounded turbulent flows. *Phys. Rev. E 79* (Feb 2009), 026703.

[36] Reider, M. B., and Sterling, J. D. Accuracy of discrete-velocity bgk models for the simulation of the incompressible navier-stokes equations. *Computers & Fluids 24*, 4 (1995), 459 – 467.

[37] Ren, X., Tang, Y., Wang, G., Tang, T., and Fang, X. Optimization and implementation of lbm benchmark on multithreaded gpu. In *Proceedings of the 2010 International Conference on Data Storage and Data Engineering* (Washington, DC, USA, 2010), DSDE '10, IEEE Computer Society, pp. 116–122.

[38] Ryoo, S., Rodrigues, C. I., Baghsorkhi, S. S., Stone, S. S., Kirk, D. B., and Hwu, W.-m. W. Optimization principles and application performance evaluation of a multithreaded gpu using cuda. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming* (New York, NY, USA, 2008), PPoPP '08, ACM, pp. 73–82.

[39] Sbragaglia, M., Benzi, R., Biferale, L., Succi, S., Sugiyama, K., and Toschi, F. Generalized lattice boltzmann method with multirange pseudopotential. *Physical Review E: Statistical, nonlinear, and soft matter physics 75*, 2 (2007), 026702.

[40] Shan, X., and Chen, H. Lattice boltzmann model for simulating flows with multiple phases and components. *Phys. Rev. E 47* (Mar 1993), 1815–1819.

[41] Smagorinsky, J. General circulation experiments with the primitive equations. *Monthly Weather Review 91* (1963), 99 – 164.

[42] Sterling, J. D., and Chen, S. Stability analysis of lattice boltzmann methods. *Journal of Computational Physics 123*, 1 (1996), 196 – 206.

[43] Succi, S., and Benzi, R. Lattice boltzmann equation for quantum mechanics. *Physica D: Nonlinear Phenomena 69*, 3 - 4 (1993), 327 – 332.

[44] Swift, M. R., Orlandini, E., Osborn, W. R., and Yeomans, J. M. Lattice boltzmann simulations of liquid-gas and binary fluid systems. *Phys. Rev. E 54* (Nov 1996), 5041–5052.

[45] Swift, M. R., Osborn, W. R., and Yeomans, J. M. Lattice boltzmann simulation of nonideal fluids. *Phys. Rev. Lett. 75* (Jul 1995), 830–833.

[46] Tolke, J., and Krafczyk, M. Teraflop computing on a desktop pc with gpus for 3d cfd. *Int. J. Comput. Fluid Dyn. 22*, 7 (Aug. 2008), 443–456.

[47] Yoshino, M., Hotta, Y., Hirozane, T., and Endo, M. A numerical method for incompressible non-newtonian fluid flows based on the lattice boltzmann method. *Journal of Non-Newtonian Fluid Mechanics 147*, 1-2 (2007), 69 – 78.

[48] Yu, H., and Girimaji, S. S. Near-field turbulent simulations of rectangular jets using lattice boltzmann method. *Physics of Fluids 17*, 12 (2005), 125106.

[49] Yu, H., Luo, L.-S., and Girimaji, S. S. Les of turbulent square jet flow using an mrt lattice boltzmann model. *Computers & Fluids 35*, 8 - 9 (2006), 957 – 965. <ce:title>Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science</ce:title>.

[50] Yu, Z., and Fan, L.-S. An interaction potential based lattice boltzmann method with adaptive mesh refinement (amr) for two-phase flow simulation. *J. Comput. Phys. 228*, 17 (Sept. 2009), 6456–6478.

[51] Yu, Z., and Fan, L.-S. Multirelaxation-time interaction-potential-based lattice boltzmann model for two-phase flow. *Phys. Rev. E 82* (Oct 2010), 046708.

[52] Zou, Q., and He, X. On pressure and velocity boundary conditions for the lattice boltzmann bgk model. *Physics of Fluids 9*, 6 (1997), 1591–1598.

Spinodal decomposition with the lattice Boltzmann method
A generalized multicomponent model with high order isotropy component interaction

Mikael Håkansson

Technical report no xxxx:xx
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden

Telephone +46 31 772 1000

TRYCKERIETS NAMN
Gothenburg, Sweden 2012