Delft University of Technology
Faculty of Applied Sciences

Bachelor Thesis

Investigating the effect of microchannel junction geometry on two-phase flow using the Lattice Boltzmann Method

Author:
Feline de Groot
4584678

December 14, 2021

Committee:
Dr. ir. M. Rohde (Supervisor)
Dr. ir. R. M. De Kruijff
A. Sudha
Department: Radiation Science and Technology (RST)
section: Reactor Physics and Nuclear Materials (RPNM)
Group: Transport Phenomena in Nuclear Applications

TU Delft Delft University of Technology

Abstract

Over the past years, interest has grown on the use of microfluidic systems in chemical reactors. The small scale on which a microfluidic system operates results in better controllability of the system and it allows higher surface to volume ratios, which improves the heat and mass transfer in the system. One particular application in which microfluidic channels are used is liquid-liquid extraction, where two immiscible fluids are brought together in a microchannel where particles are transferred from one fluid to the other before the fluids are separated again at the end of the channel. Inside the main channel, the two fluids create different flow patterns dependent on different fluid properties and channel properties. Parameters like fluid density, viscosity, flow rate and channel geometry have proven to affect the flow pattern formed in the channel. Since the formed flow pattern in the microchannel is of great influence on for example the efficiency of the phase separation at the end of a microchannel or the mass and heat transfer between the fluids, many researches have already focussed on the effect of different parameters on the flow pattern. This research investigates the effect of junction geometry on the flow pattern by simulating and analyzing flow patterns of water and n-heptane flowing through a T-shaped microchannel, using the Lattice Boltzmann Method. The results have been compared to the results found by Zheng Liu for a Y-shaped channel to determine how the junction geometry affect the observed flow patterns.

The Lattice Boltzmann Method (LBM) is a Computational Fluid Dynamics method for simulating fluid flow. The LBM is based on the Boltzmann equation for gases, from where the Navier Stokes equations can be derived. In short, the LBM solves the Boltzmann equation by simulating the dynamics of groups of particles at different lattice nodes in space. By plotting the simulated density contours, flow pattern behaviour in the simulated channel can be analyzed. Using a numerical method to investigate flow patterns instead of performing experimental research saves a lot of time and money needed to fabricate an experimental setup and perform the experiments.

In this research, two-phase flow of water and n-heptane has been simulated in a T-shaped microchannel with a width of $100\mu m$. The simulations showed that slug flow was formed at low Capillary numbers. Increasing the inlet flow rates of the fluids and, in doing so, increasing the Capillary numbers, resulted initially in a decrease of slug length before the flow transitioned to parallel flow. The transition from slug to parallel flow was observed in the simulations at Capillary numbers of about 10^{-4} for n-heptane and about $4 \cdot 10^{-4}$ for water. This observation indicates that a T-shaped channel shows a greater tendency for parallel flow compared to a Y-shaped channel, as Zheng Liu observed the transition from slug to parallel flow at Capillary numbers of about $2 \cdot 10^{-4}$ for n-heptane and 10^{-3} for water. The results for the T-channel have also been compared to the experimental results of Yagodnitsyna et al for three different liquid-liquid systems inside a T-channel. They observed transition from slug to parallel flow at Weber numbers of the same order of magnitude as the results from the simulations in this research.

However, the numerical model doesn't simulate the microfluidic two-phase flow flawless. For example, a flow pattern has been observed in the simulations named *transition flow* that doesn't have been observed in physical experiments. The occurrence of this flow pattern in the simulations could be due to a numerical instability or an error in the algorithm.

In further research, it would be beneficial to focus on simulating the entire microchannel, complete with T-junction inlet and outlet. By analyzing not only the effect of junction geometry on the formed flow pattern at the inlet but also on the phase separation at the outlet, a lot of useful information could be collected. Thereby it is recommended to dedicate more research to the effect of inlet junction geometry on the flow pattern by simulating more shapes of microchannels in order to gain more detailed information on the effect of junction geometry on the flow pattern. Furthermore, the Lattice Boltzmann model could be improved by carrying out more research on the transition flow pattern to find out what numerical instability or error is causing this flow pattern to occur.

Contents

1 Introduction

1.1 Microfluidic flow

Microfluidic systems have become increasingly interesting over the past couple of years for applications in chemical reactors. This is due to the many advantages that come with the use of microchannel reactors compared to the standard reactor. For example, microfluidic systems allow very high surface to volume ratios which cause higher heat and mass transfer rates in the system. [38] In addition, the smaller volumes that flow through microfluidic systems make them better controllable and more safe to work with when dangerous chemicals are involved. [40] Due to these advantages, microfluidic systems are very commonly used in applications like metal extraction, organic synthesis, purification of bio-molecules and pesticide analysis. [17]

One specificly interesting application of microfluidic systems is in the separation of particles using so-called liquid-liquid extraction (LLE). LLE is applied in many differend fields, it is used for example for the purification of enzymes [33], the extraction of metals [13] [32], the extraction of essential oils in the food industry [1] and the purification of radioisotopes in the field of Nuclear Medicine [35]. For LLE, two non-mixing fluids are brought together in a microchannel through two inlets, one fluid containing the particles that are to be extracted. While the fluids flow together through the main channel, particles are transferred from one fluid to the other. At the end of the channel, the fluids are separated from each other through for example a membrane or by an outlet junction. In figure 1, a schematic overview of the LLE process is shown.

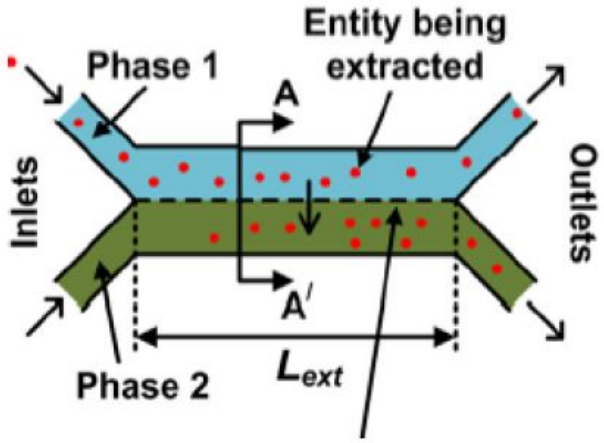
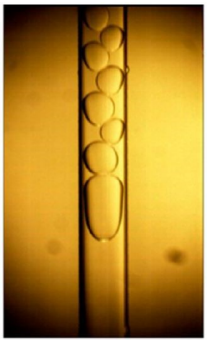


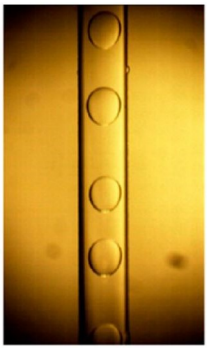
Figure 1: Schematic overview of Liquid-Liquid Extraction in a microchannel

The fluids that are used for LLE must be immiscible. For example, one aqueous fluid can be used where particles are dissolved in, with an organic fluid where the particles will be transferred to during the process. The surface area between the two fluids is of great influence on the particle transfer, a greater surface-to-volume ratio means a higher particle transfer rate. [54] Considering this, the flow pattern will be of great importance to the efficiency of the particle transfer, as it determines the contact area between the fluids. This is why many studies have already been performed on the effect of different parameters on the created flow pattern in microchannels. A selection of observed flow patterns is shown in figure 2 below.

(a) Slug Flow by Darekar et al [12]

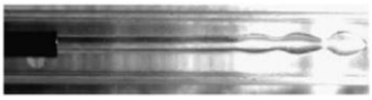


(b) Slug Droplet Flow by Darekar et al [12]

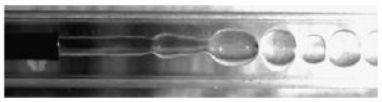


(c) Droplet Flow by Darekar et al [12]

(d) Parallel Flow by Darekar et al [12]

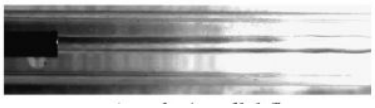


(e) Deformed Interface by Kashid et. al. [27]



(f) Slug Droplet flow by Kashid et al [27]

(g) Droplet flow by Kashid et al [27]



(h) Parallel flow by Kashid et al [27]

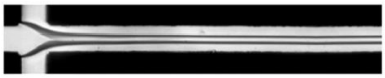
(i) Slug flow by Zhao [53]

(j) Droplet flow by Zhao [53]

(k) Deformed Interface flow by Zhao [53]

(l) Parallel flow by Zhao [53]

(m) Slug flow by Cubaud et al [45]



(n) Thread flow by Cubaud et al [45]

(o) Jet flow by Cubaud et al [45]

(p) Droplet flow by Tsaoulidis et al [11]

(q) Quasi Annular flow by Tsaoulidis et al [11]

(r) Intermitted flow by Tsaoulidis et al [11]

Figure 2: Different types of flow patterns observed in microfluidic channels

In for example the application of radioisotope transfer, it is beneficial for the process efficiency to decrease the time needed for the LLE process as much as possible, since radioisotopes often have short half lives. Parallel flow would be the most suitable pattern for reducing the time needed for the process. This is because stable parallel flow with no leakage at the outlet doesn't need, for example, a membrane to separate the two fluids at the end of the channel, which would be needed in the case of droplet flow or slug flow. [25] The separation of the fluids flowing parallel can be done by placing another junction at the end of the microchannel. This saves time which is essential when working with radioactive isotopes that possibly have short half lives. [17]

In addition to the flow pattern, the flow rate of the fluids is of great importance to the effectiveness of the LLE process. The extraction efficiency in LLE using an aqueous and an organic phase is given by the following equation [36]:

$$EE = \frac{C_{org,out} \times Q_{org}}{C_{aq,in} \times Q_{aq}} \tag{1}$$

Where Q represents the volumetric flow rate of the fluids and $C_{org,out}$ and $C_{aq,in}$ refer to the concentration of the organic phase at the outlet of the channel and the aqueous phase at the inlet respectively. In the application of LLE, it is important that the contact time of the two fluids is sufficient for the dissolved parts to transfer between the fluids. Therefore, flow rates as low as possible are favourable for LLE in order to increase the contact time of the fluids allowing the particle transfer more time to happen properly.

Since the flow pattern is of great influence on the efficiency of LLE, many studies have already been performed on flow pattern behaviour in two-phase microfluidic systems. They focussed on determining the influence of different kinds of parameters on the obtained flow pattern. Parameters that were found to affect the flow pattern are for example inlet flow rate, interfacial tension between the fluids, channel cross-section and wall wettability. The latter is a parameter which indicates the amount of adhesive forces between the solid walls and the fluids. [7] A wettable (hydrophilic) wall means that the wall shows great affinity for the fluid in question, where a non-wettable (hydrophobic) wall doesn't show affinity for the fluid. In the latter case, the fluid will try to minimize the contact area with the wall. Section 3.3.4 will elaborate more on this property. In addition to the aforementioned, the geometry of the inlet junction can affect the flow pattern created in the microchannel. There are multiple possibilities for the geometry of the inlet junctions, however the two most commonly used shapes of microchannels are T-junction microchannels and Y-junction microchannels. [27] A schematic overview of these two geometries is given below in figure 3.

(a) Microchannel with a T-junction inlet

(b) Microchannel with a Y-junction inlet

Figure 3: There are two frequently applied inlet junction geometries for two-phase flow microchannel systems, T-junctions and Y-junctions. [27]

Darekar et al has investigated different types of flow patterns in microchannels with a Y-shaped junction applying varying flow parameters like microchannel diameter, flow rate, interfacial tension between the fluids and the wettability of the channel walls. At high inlet flow rates, they found parallel flow while at low inlet flow rates, slug flow was formed. Also, they found that slug and droplet flow dominated the flow regimes when the interfacial tension between the fluids or the hydrophobicity of the walls was increased. [12]

Furthermore, Dessimoz et al studied the relationship between two dimensionless quantities and the created two-phase flow pattern in a microchannel. They calculated for different systems the Reynoldsnumber, which is the ratio of inertial forces to viscous forces in a fluid, and the Capillary number, which is the ratio between viscous forces and interfacial tension beween the two fluids. [14] These experiments were performed using a system of toluene and water. The same measurements were performed on both T-junction and Y-junction channels, in order to compare the flow patterns for these two geometries. From the experiments, Dessimoz et al found that an increase of interfacial tension is favourable for slug flow and a decrease of interfacial tension leads to parallel flow. Furthermore, an increase of linear velocity eventually leads to parallel flow. From the created flow maps based on Capillary and Reynolds number can be seen that the T-shaped microchannel has a greater tendency to form slug flow, while a Y-shaped microchannel is more likely to form parallel flow.

A comparable but numerical research has been performed by Raj et al, who compared the slug length formed in a microchannel with three different inlet junction geometries. Also here, the effect of interfacial tension on the flow pattern has been studied. Raj et al just like Darekar et al studied in addition the effect of the wettability of the channel walls. They concluded that the geometry of the inlet was of no interest for the length of the slugs. Furthermore, they came to the same conclusion as Dessimoz et al on the effect of interfacial tension, which was found to be of significant weight to the flow pattern in the channels. Furthermore, Raj et al found that an increase of wall wettability would result in a transition from parallel to slug flow.[41]

In addition, many other interesting studies have been performed on the topic. Qian et al (2006) for ex-

ample numerically studied the behaviour of slugs in a two-phase gas-liquid flow in a T-shaped microchannel for different cross-sections and varying flow rates of the gas and liquid phase. They concluded the length of formed slugs in a T-shaped channel is dependent on the width of the inlet channels. They found that wider inlet channels lead to longer slugs. [38] Sang et al also studied the variation in length of slugs in a T-shaped microchannel by varying the viscosity of the fluids and comparing different inlet cross sections, finding the slug length decreases when the viscosity increases. They also concluded that the inlet cross section influences the slug length.[30] Qian et al (2019) studied the effect of the dispersed phase channel width and the channel depth on the length of formed slugs, finding that an increase of these parameters leads to greater slug lengths.[39] Furthermore, Kashid et al studied the effect of contacting geometry on two-phase liquid-liquid flow patterns, building forth on the research of Dessimoz et al for gas-liquid systems. They found similar looking flow pattern maps for Y-channels compared to T-channels, however the flow velocities at which slug flow transformed to parallel flow are slightly different for both geometries.[27] Finally, experiments have been performed in a Y-channel by Zheng Liu, comparing flow patterns in a toluene/water and a n-heptane/water system at different Capillary numbers. Increase of Capillary number resulted for both systems eventually in parallel flow, starting at slug flow.

1.2 Computational Fluid Dynamics

A frequently used way to analyze the effect of physical parameters on microfluidic flow is by simulating the flow numerically. In order to find the numerical solutions to these simulation, the field of Computational Fluid Dynamics (CFD) is used. Some of the researches named above in the subsection about microfluidic flow are based on numerical simulations. For example, Qian et al (2006) used the CFD package FLUENT [38], and Raj et al implemented the Volume of Fluid (VOF) method to simulate the slug formation in a microchannel. [41] CFD has already been used frequently in research, due to its many advantages over experimental research. It is first of all very cost-effective because no experimental setup needs to be designed and produced. In addition, CFD is able to provide much more detailed and comprehensive information on the flow compared to experimental fluid dynamics. [23]

There are countless CFD methods available to simulate flows. However, not every method is as effective for simulating multiphase flows. [26] Simulation methods that are often used herefor are for example the Volume of Fluid method (VOF) [6][24], the Phase Field method (PF) [48], the Level-Set method (LS) [44][31], the Density Functional method (DFM) [2] and the Lattice Boltzmann method (LB) [9][50]. VOF tracks the volume of both fluids in cells that belong to the interface between the fluids. [8] PF describes the interface as a transition layer, which should be relatively thin in order to make the model adequately accurate. [48] LS is used very frequently due to the simplicity of the model and uses a level-set function to define the interface of the flow. [43] Lastly, LB methods use particle distribution functions based on the Boltzmann equation to model fluid flow. Interface tracking is not necessarily required for this method. [49] LB methods are often used due to their simplicity and their high computational efficiency.[22] There are multiple LB models available that are suitable for multiphase flow. Examples of this are the pseudo-potential model [4], the free energy model [9], the mean field model [49] and the color gradient model [47][42]. The latter is very beneficial due to the fact that parameters like interfacial tension, density ratio and wettability can be separately adjusted. Therefore, in order to determine the effect of these quantities on the flow pattern, the color gradient model will be a sensible choice. That is why in this work, the Lattice Boltzmann color gradient Method will be used to simulate the flow of two immiscible fluids through a microchannel.

1.3 Motivation

As stated in section 1.1, a lot of research already has been done to two-phase microfluidic flow. Flow patterns in T-junction and Y-junction microchannels have been experimentally compared by Dessimoz et al and the relationship with the Capillary number and Reynolds number of the flows have been determined. Raj et al also compared different geometries of the microchannel numerically by analyzing slug lengths. However, the conclusions of Raj et al deviated at some points from the experimental results obtained by Dessimoz et al. This makes it interesting to investigate whether other simulation methods like the LB method will be able to simulate the experimental observations more accurate. This gives rise to a couple of research questions:

How does the inlet junction geometry of a microchannel affect the created flow pattern of two immiscible fluids?

1. Does a correlation exist between the flow pattern and the Capillary numbers of the fluids?
2. How do the flow patterns created in a T-shaped microchannel differ from the flow patterns created in a Y-shaped microchannel?
3. Do the computational results of this research match the experimental results found in literature?

1.4 Thesis Outline

This research will be focused on analyzing the flow of two immiscible fluids in a T-shaped microchannel compared to that in a Y-shaped microchannel, using LBM. In order to answer the research questions formulated in section 1.3, a Lattice Boltzmann model will be made for two-phase flow in a T-channel. The flow patterns formed in the T-shaped channel will be analyzed for different Capillary and Reynolds numbers by varying the linear velocity of the fluids. Those results will be compared to results that already have been obtained by Zheng Liu using LBM for a Y-shaped microchannel, in order to find any affect of inlet junction geometry on the flow pattern. Finally, the found results and correlations will be compared to experimental results from literature in order to determine the accuracy of the numerical model.

In chapter 2 of this thesis, some essential background theory to support this research will be discussed. First, the theory needed to understand concerning fluid dynamics is discussed and secondly, the concept of Lattice Boltzmann is briefly discussed. In chapter 3, more in-depth information will be given on the Lattice Boltzmann method used in this research. Chapter 4 will give an overview of the results obtained by the numerical simulations, together with a discussion of the results. Chapter 5 then discusses the conclusions that can be drawn from the results. Finally, chapter 6 will give an overview of recommendations for further research based on this research.

2 Theoretical Background

2.1 Fluid Dynamics

Fluid dynamics is a field of physics and it is a subdiscipline of fluid mechanics together with fluid statics. Where fluid statics focuses on fluids at rest, fluid dynamics describes the flow of fluids. In this subsection, two important equations in fluid dynamics will be briefly discussed. [15]

The first important equation is called the continuity equation. This equation describes the flow of mass in and out of a fixed volume. It basically states that the rates at which mass flows into and out of the system should be equal to each other. The continuity equation is defined by equation 2, where ρ is the fluid density, t is the time and $\mathbf{u}$ is the fluid velocity.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2}$$

In the continuity equation, the last vector $\rho\mathbf{u}$ is called the momentum density.

The second thing that can be considered in fluid dynamics is the change of momentum in a system. The momentum in an ideal fluid can change due to fir example pressure differences and external body forces on the system. Considering an incompressible flow, i.e. flow with a constant density, the incompressible Navier-Stokes equation describes the momentum equation of the system. The Navier-Stokes Equation is given below in equation 3.

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \mu \Delta \mathbf{u} + \mathbf{F} \tag{3}$$

Here, μ is the dynamic viscosity of the fluid, which is a measure for how much a fluid resists deformation, p represents the pressure and $\mathbf{F}$ are the external body forces on the system. Furthermore, Δ is the Laplace operator defined by equation 4 and $\frac{D}{Dt}$ is called the material derivative, defined by equation 5.

$$\Delta = \nabla \cdot \nabla \tag{4}$$

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \tag{5}$$

2.1.1 Dimensionless numbers

In fluid dynamics, a couple of dimensionless numbers play an important role in order to analyze the behaviour of fluid flows. These numbers describe the relation between forces working on the system like viscous, interfacial and gravitational forces. Considering fluid dynamics on the microscale, gravity doesn't need to be taken into account. This is because the Bond number, that gives the ratio between gravitational and interfacial forces, is of the order of magnitude 10^{-5} in microfluidic systems. This means that the gravitational force is of such small influence on the flow in microfluidic systems, it's effect can be neglected. [28] System characteristics that are of significant importance in fluid dynamics on the microscale are inertial, interfacial and viscous forces. The ratios between these forces are given by the Reynolds and Capillary number, which is why these dimensionless numbers are the most useful for analyzing microfluidic flow. Furthermore, this research will use the Weber number, which represents the relation between drag force and interfacial force in a flow.

The Reynolds number gives the ratio between inertial forces and viscous forces in a flow. This number gives an indication of the turbulence of the flow. A low Reynolds number indicates a laminar flow, while a higher Reynolds number means the flow will be turbulent. The Reynolds number is defined according equation 6 given below.

$$Re = \frac{uL\rho}{\mu} = \frac{uL}{\nu} \tag{6}$$

Here, u is the flow speed of the fluid, L is the characteristic length of the flow. This could be the channel width, diameter or length, depending on the case. Furthermore, ρ is the fluid density, μ is the dynamic viscosity of the fluid and ν is the kinematic viscosity of the fluid, which is defined as $\nu = \frac{\mu}{\rho}$. In microfluidic systems, the diameter of the channel is very small, resulting in small Reynolds numbers. For this reason, only laminar flow is expected in microfluidic channels.

The Capillary number represents the ratio of viscous forces and interfacial forces in a two-phase flow. The Capillary number is defined according to equation 7.

$$Ca = \frac{\mu u}{\sigma} = \frac{\nu \rho u}{\sigma} \tag{7}$$

In equation 7, σ is the interfacial tension between the two phases. In microfluidics, the flow is largely dominated by viscous and interfacial forces. [5] The Capillary number is therefore very suitable to describe fluid phenomena like flow patterns in microchannels.

Finally, as mentioned before the Weber number represents the relation between drag forces and interfacial forces in a flow. The Weber number can be calculated according equation 8 below.

$$We = \frac{\rho L u^2}{\sigma} \tag{8}$$

2.1.2 Contact angle

The contact angle in fluid dynamics is an indicator of the adhesive between a fluid and a solid surface. In order to determine the contact angle, a situation is considered where a droplet of the fluid in question is placed on a solid surface and, in this example, it is surrounded by a gas. The droplet will spread out more or less on the solid surface. A schematic overview of this situation is given below in figure 4.

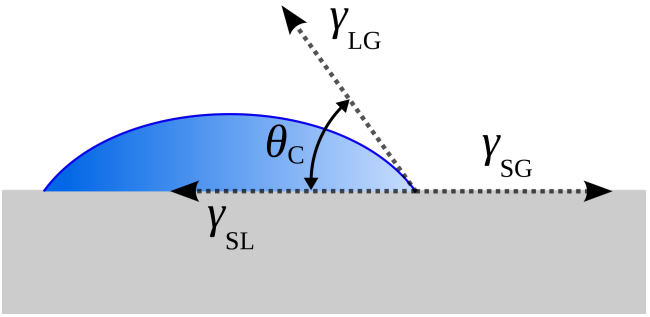


Figure 4: Schematic overview of the contact angle of a fluid on a solid surface.

The contact angle is defined as the angle between the fluid-gas interface and the fluid-solid interface. [16] In figure 4, the contact angle is denoted as θ_C. The formula for the contact angle is given below in equation 9. This definition has been developed by Thomas Young in 1805, and it is therefore usually referred to as Young's equation. [55]

$$cos(\theta_C) = \frac{\sigma_{SG} - \sigma_{SL}}{\sigma_{LG}} \tag{9}$$

In Young's equation, σ_{SG} is the interfacial tension between the solid surface and the gas surrounding the liquid, σ_{SL} is the interfacial tension between the solid and the liquid and σ_{LG} is the interfacial tension between the liquid and the gas.

As mentioned before, the contact angle indicates the affinity of the fluid with the solid interface. This is also referred to as the wettability of the liquid with the solid. A smaller contact angle indicates a greater affinity of the liquid with the solid, meaning the surface is more wettable. Vice versa, a greater contact angle means less affinity between the fluid and the solid, i.e. the surface is less wettable. In general, when the contact angle is less than 90°, the surface is called hydrophilic. At contact angles higer than 90°, the fluid tries to minimize the contact area with the surface. The surface is then called hydrophobic. [52] The wettability of a surface is dependent on different factors. Some big influences are for example the roughness of the solid surface and the surface tension of the fluid.

2.2 Lattice Boltzmann Method

This research uses the Lattice Boltzmann Method (LBM) for simulating the flow of two immiscible fluids in a microchannel. This section broadly outlines the general principle of the LBM for simulating a system containing

one single phase. Chapter 3 of this work will elaborate in more detail on the specific Lattice Boltzmann model (the color-gradient RK model) used in this research for simulating a two-phase flow. In short, the LBM simulates the dynamics of groups of particles at different lattice nodes by so-called streaming and colliding steps. Section 2.2.1 gives an introduction to the method in general, followed by section 2.2.2 which explains the streaming and collision step in more detail. Finally, in section 2.2.3, the conversion between physical and Lattice Boltzmann parameters is explained.

2.2.1 Introduction

As mentioned before in section 1.2, the Lattice Boltzmann method is used in this research in order to simulate the flow of immischible fluids in a microchannel numerically. Fluid dynamics can be simulated on the macroscopic scale by describing the motion of small volume elements in a much bigger volume, or it can be simulated in the microscopic scale by describing the dynamics of individual molecules. The Lattice Boltzmann method operates at the mesoscopic scale, which is a scale in between macro- and microscopic. On this scale, the dynamics of groups of particles in the fluid are simulated. These particle groups are described by the probabilistic particle distribution function $f(x, \xi, t)$, where ξ represents the lattice velocity. This particle distribution function is connected to macroscopic variables like the density and the fluid velocity. The mass density ρ and the momentum density $\rho\mathbf{u}$ by integrating f weighted with a function of ξ. These integrals are given below in equation 10. [15]

$$\begin{aligned} \rho(\mathbf{x}, t) &= \int f(\mathbf{x}, \boldsymbol{\xi}, t) d^3\xi \\ \rho\mathbf{u}(\mathbf{x}, t) &= \int \boldsymbol{\xi} f(\mathbf{x}, \boldsymbol{\xi}, t) d^3\xi \end{aligned} \tag{10}$$

In every system of a gas or fluid that is moving, when the system is left alone long enough, eventually it will settle down to an equilibrium situation with an equilibrium distribution function $f^{eq}(\mathbf{x}, \xi, t)$. The development of f over time is described by the Boltzmann equation for gases, given in equation 11. [19]

$$\frac{\partial f}{\partial t} + \xi \cdot \nabla f + \mathbf{F} \cdot \nabla_\xi f = \Omega(f) \tag{11}$$

From the In the Boltzmann Equation, the first two terms on the left side indicate the particle distribution function that is moved with lattice velocity ξ. The third term on the left stands for the forces that affect the particles and their velocities. Furthermore, the source term $\Omega(f)$ is called the *collision operator*, which indicates the local particle distribution due to collisions of particles. This collision operator originally is determined by a complicated integral. [15] However, in the Lattice Boltzmann Method, an approximation of the collision operator is used developed by Bhatnagar, Gross and Krook, named the *BGK collision operator*. This approximation is defined according equation 12, and is a lot easier to work with.

$$\Omega(f) = -\frac{1}{\tau}(f - f^{eq}) \tag{12}$$

Here, τ is a time constant indicating the *relaxation time* of the system, i.e. the time it takes for the system to reach the equilibrium distribution.

The Lattice Boltzmann Method in CFD is developed to numerically solve the Boltzmann equation for gases. The lattice Boltzmann equation can be used to simulate the Navier Stokes equations, describing the motions of the fluid. It describes the dynamic behaviour of a gas or fluid in a system on the mesoscale, from where it can be related to the macroscale by determining a conversion factor between the model parameters and the corresponding physical parameters. Section 2.2.3 will elaborate more on this conversion. The LBM divides space into a grid of different cells, each in its own state with an individual discrete particle distribution. The LBM updates these individual state each timestep for all cells.

The basic element of the LBM is the so-called *particle population* at each grid point. These particle populations are the Particle Distribution Function (PDF) per lattice node, denoted by f_i. The macroscopic fluid properties can be easily determined from the weigthed sum over all Particle Distribution Functions: [15]

$$\begin{aligned} \rho(\mathbf{x}, t) &= \sum_i f_i(\mathbf{x}, t) \\ \rho\mathbf{u}(\mathbf{x}, t) &= \sum_i e_i f_i(\mathbf{x}, t) \end{aligned} \tag{13}$$

In equation 13, i represents the direction of the discrete PDF with e_i being the particle velocity in the ith direction. There are multiple models possible that are used to model the discrete velocity, dependent on the number of dimensions and the number of possible directions for the particles to move. These discrete velocity models are indicated as D_nQ_m, n being the number of dimensions and m being the number of possible directions. The four most commonly used models are D2Q7, D2Q9, D3Q15 and D3Q19, which are shown in figure 5. In this research, the D2Q9 model has been used. The discrete velocities for this model are given by:

$$\begin{bmatrix} \boldsymbol{e}_0 & \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 & \boldsymbol{e}_4 & \boldsymbol{e}_5 & \boldsymbol{e}_6 & \boldsymbol{e}_7 & \boldsymbol{e}_8 \end{bmatrix} = c \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}$$

where c is the lattice speed defined as $c = \frac{\Delta x}{\Delta t}$, where Δx is 1 lattice unit (lu) and Δt is 1 time step (ts).

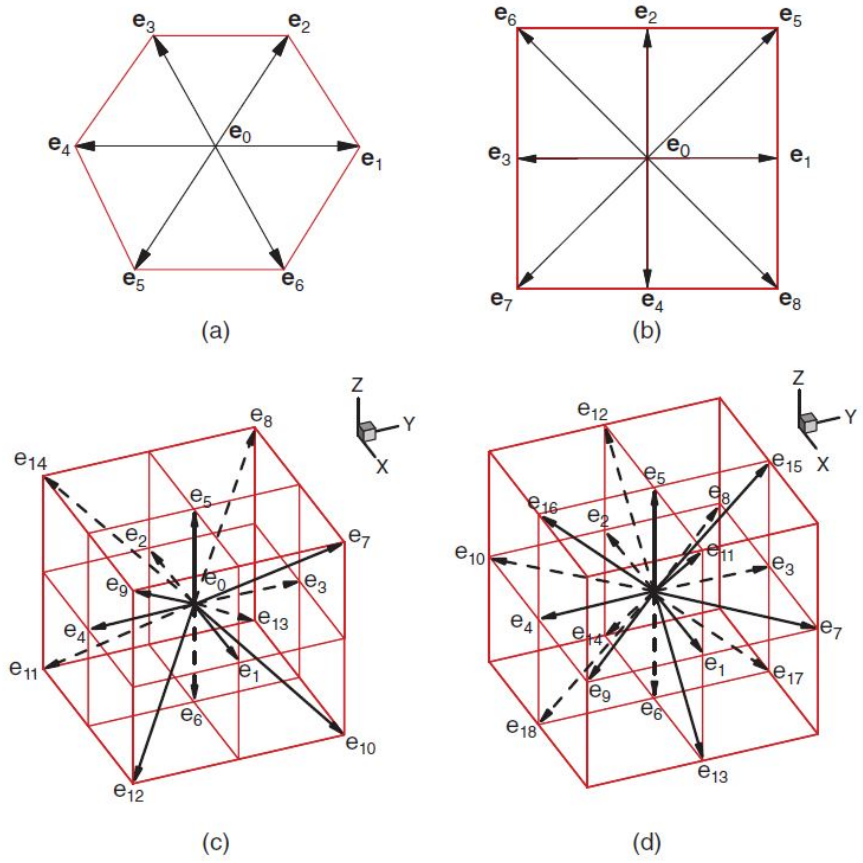


Figure 5: The four most commonly used discrete velocity models a) D2Q7 b) D2Q9 c) D3Q15 d) D3Q19 [19]

Just as the mass density and momentum density can be discretized according equation 13, the Boltzmann equation can be discretized to the Lattice Boltzmann Equation (LBE), given below in equation 14.

$$f_i(\mathbf{x} + \mathbf{e}_i\Delta t, t + \Delta t) = f_i(\mathbf{x}, t) - \frac{1}{\tau}\left(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right) + S_i(\mathbf{x}, t) \tag{14}$$

In the LBE, S_i is a source term added into the standard Lattice Boltzmann equation, which encompasses the forcing term. Furthermore, the relaxation time τ is related to the kinematic viscosity ν according equation 15, where c_s is the speed of sound in the fluid which depends on the lattice speed c by $c_s^2 = \frac{c^2}{3}$.

$$\nu = c_s^2(\tau - 0.5)\Delta t \tag{15}$$

The discretized equilibrium distribution function according the LBM is calculated according equation 16.

$$f_i^{eq}(\mathbf{x}, t) = w_i\rho\left[1 + \frac{e_i \cdot \mathbf{u}}{c_s^2} + \frac{(e_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2}\right] \tag{16}$$

Here, w_i are the weight belonging to each lattice direction. In the D2Q9 model, these weights are given by equation 17.

$$\mathbf{w} = \begin{cases} w_0 = \frac{4}{9} \\ w_{1,2,3,4} = \frac{1}{9} \\ w_{5,6,7,8} = \frac{1}{36} \end{cases} \tag{17}$$

In short, the LBM solves the Boltzmann equation numerically by simulating the dynamics of groups of particles at different lattice nodes. All nodes have a particle distribution described by f_i at each time step t. The distance in between lattice nodes is 1 lu and the time it takes a particle to travel this distance is 1 ts. The particle movement at each time step is simulated through four phases, which are repeated over a large number of time steps in order to simulate the macroscopic flow over time. In the next few sections, each step will be briefly summarized and explained.

2.2.2 Streaming and Colliding

The two phases to consider in the LBM are the streaming and the collision step. Prior to these phases, the macroscopic density ρ and velocity $\mathbf{u}$ are determined and used to calculate the equilibrium distribution function f_i^{eq} according equation 16. Next, the collision step follows, where all possible outcomes of two particles in the system colliding are considered. The post-collision distribution function, which remains at the end of the collision step, becomes the new particle distribution from where the streaming step will take place. During this streaming (or propagation) step, particles move from the lattice node they are located in after the collision step to a neighbouring lattice node, or they remain located at their original lattice node. During the streaming step, the particles move along a straight line in any direction during one time step Δt:

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \mathbf{e}_i(t)\Delta t \tag{18}$$

A visualization of the streaming and colliding between lattice nodes is shown in figure 6.

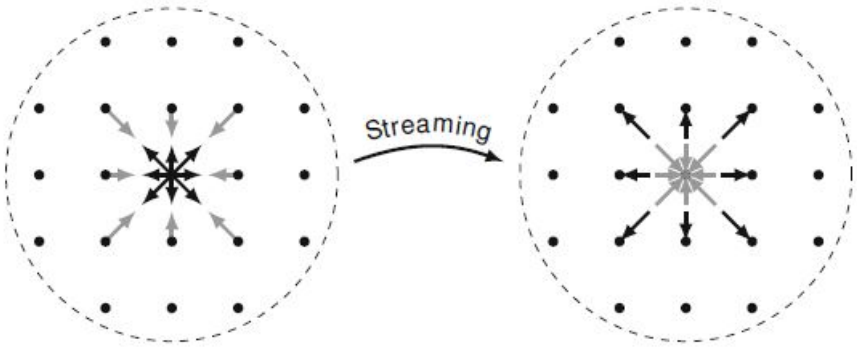


Figure 6: Visualization of the streaming and collision step, particles are streaming to neighbouring cells (black arrows) and vice-versa (grey arrows). Left shows the particle distributions *before* streaming, right shows the particle distributions *after* streaming.

2.2.3 Unit Conversion

In order to relate the results found by the Lattice Boltzmann simulation, a conversion factor needs to be determined between physical parameters and model parameters. [15] Conversion factors are denoted by C_x, x for a physical parameter x with corresponding Lattice Boltzmann (LB) parameter x^*. The common rule for notation of LB parameters is to add a * to the symbol, in order to separate the LB parameters from physical parameters. For example, can be written:

$$\ell^* = \frac{\ell}{C_\ell} \tag{19}$$

In this example, ℓ is the physical lenght and ℓ^* is the LB length. C_ℓ is the corresponding conversion factor. In order to calculate LB parameters from other LB parameters, the so called law of similarity is used. This law states that dimensionless numbers are the same when calculated from physical parameters and from LB parameters [15]. For example, when calculating the Capillary number applies the following relation:

$$Ca = \frac{\nu \rho u}{\sigma} = \frac{\nu^* \rho^* u^*}{\sigma^*} \tag{20}$$

The law of similarity can also be used to calculate conversion factors from other conversion factors. Combining relation 19 and 20, the following relation is found:

$$\frac{C_\nu C_\rho C_u}{C_\sigma} = 1 \tag{21}$$

This shows that for example, the conversion factor for the kinematic viscosity ν can be calculated from those for ρ, u and σ by rewriting equation 21.

3 Multiple-Relaxation-Time Color-Gradient Model Description

There are multiple LB models available for simulating fluid flow using the LBM. The model used for the numerical experiments done in this research is based on the Color-Gradient Model described by Ba et al, using a multiple-relaxation-time (MRT) collision operator. [51] This chapter discusses this model in more detail. Section 3.1 explains the implementation of the model in numerical simulations. Section 3.2 discusses the initial parameters chosen for the model and the fluids that are simulated. Next, section 3.3 discusses the boundary conditions applied in the model and section 3.4 closes this chapter with an overview of the Python algorithm used in this research to simulate flow.

3.1 Model Outline

In this work, the flow patterns of two immiscible fluids inside a microchannel with a T-shaped inlet junction are investigated. In the RK model, the two fluids are separated by two different colors, blue and red. The fluids both are represented by their own PDF, named f_i^1 and f_i^2. The total particle distribution is given by the sum over all PDF's:

$$f_i(\mathbf{x}, t) = \sum_{i,k} f_i^k(\mathbf{x}, t), \qquad k = 1, 2 \tag{22}$$

3.1.1 Streaming Step

The streaming step is similar to the general streaming step mentioned in section 2.2:

$$f_i^k(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) = f_i^{k+}(\mathbf{x}, t) \tag{23}$$

Here, f_i^{k+} is the PDF after the execution of the recoloring step. The particle velocities e_i for the D2Q9 model are used.

3.1.2 Bounceback Step

The next phase in the RK model is the Bounceback step. In the LBM, two types of lattice nodes are defined for the simulation. The first type is called a *fluid node*, which represents a part of space where the fluid can flow. The second type of lattice node is a *solid node*, representing a solid, e.g. a channel wall. At solid boundaries in fluid mechanics, a boundary condition applies called the *no slip boundary*. This boundary condition states that the velocity of the fluid at a solid wall must be the same as the velocity of the wall itself, meaning the velocity of a fluid at a channel wall is zero as the velocity of the wall is zero.

In the LBM, the boundary condition described above means specifically that a particle that flows from a fluid node to a solid node during the streaming step, is bounced back directly in the same time step to the fluid node it came from, making sure the effective velocity of the fluid at the wall is zero. Since this concerns a boundary condition, section 3.3.1 will elaborate in more detail on this bounceback step.

3.1.3 Collision Step

After the bounceback step, the collision step is executed, described according equation 24, where f_i^{k*} is the post-collision PDF.

$$f_i^{k*}(\mathbf{x}, t) = f_i^k(\mathbf{x}, t) + (\Omega_i^k)^1 + (\Omega_i^k)^2 \tag{24}$$

Here, $(\Omega_i^k)^1$ is the BGK operator, given by:

$$(\Omega_i^k)^1 = -\frac{1}{\tau}(f_i^k(\mathbf{x}, t) - f_i^{k,eq}(\mathbf{x}, t)) \tag{25}$$

The RK model adds an extra collision term to the BGK operator, which is called the *perturbation operator*, denoted as $(\Omega_i^k)^2$. Since the RK model simulates flow of two different fluids, collisions will not only take place between similar particles (of the same fluid), but also non-similar particles (of different fluids) will collide with each other. The perturbation operator has been added to the model in order to generate the interfacial tension between the fluids and take into account collisions between non-similar particles at the interface of the fluids. In literature, different variants of this more complicated collision term are found. In this research, the perturbation operator used by Ba et al [51] and Halliday et al [21] has been used:

$$(\Omega_i^k)^{(2)} = A^k W_i \left(1 - \frac{w^k}{2}\right) [3(\mathbf{e}_i - \mathbf{u}) + 9(\mathbf{e}_i \cdot \mathbf{u})\mathbf{e}_i] \cdot \mathbf{F}_s \tag{26}$$

In the upper equation, A_k is the part of interfacial tension that is contributed by fluid k, satisfying $\sum_k A^k = 1$. The final parameter $\mathbf{F}_s$ represents the interfacial tension, which can be expressed according equation 27:

$$\mathbf{F}_s = -\frac{1}{2}\sigma K \nabla \rho^N \tag{27}$$

Here, ρ^N is defined as $\rho^N = (\rho_1 - \rho_2)/(\rho_1 + \rho_2)$, and K represents the local curvature of the interface between the fluids. K is defined in 2D according equation 28, where n_x and n_y are parts of the outward pointing unit normal vector $\mathbf{n}$ of the interface.

$$K = n_x n_y \left(\frac{\partial}{\partial y} n_x + \frac{\partial}{\partial x} n_y\right) - n_x^2 \frac{\partial}{\partial y} n_y - n_y^2 \frac{\partial}{\partial x} n_x \tag{28}$$

The equilibrium distribution function $f_i^{k,eq}(\mathbf{x}, t)$ in the RK model is almost quite the same as the general LBM equilibrium distribution function. :

$$f_i^{eq}(\mathbf{x}, t) = \rho_k \left(C_i + w_i \left[1 + \frac{e_i \cdot \mathbf{u}}{c_s^2} + \frac{(e_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2}\right]\right) \tag{29}$$

The only difference here with the general LBM equilibrium distribution function is an added coefficient C_i. This parameter is added because the RK model simulates fluids with different densities, which needs to be taken into account. C_i is related to the ratio of the two fluid densities, and it differs for different directions. The different values for C_i are given below for each direction i,:

$$C_i = \begin{cases} \alpha_k, & i = 0 \\ \frac{1-\alpha_k}{5}, & i = 1, 2, 3, 4 \\ \frac{1-\alpha_k}{20}, & i = 5, 6, 7, 8 \end{cases} \tag{30}$$

Here, α_k is a parameter that adjusts the density of the fluids. Furthermore, the density of the kth component ρ_k, the total density ρ and the momentum $\rho\mathbf{u}$ are described according equation 30 given below.

$$\begin{aligned} \rho_k &= \sum_i f_i^k \\ \rho &= \sum_k \rho_k \\ \rho\mathbf{u} &= \sum_k \sum_i f_i^k \mathbf{e}_i \end{aligned} \tag{31}$$

The density ratio of the fluids is given by:

$$\kappa = \frac{\rho_1}{\rho_2} = \frac{1 - \alpha_2}{1 - \alpha_1} \tag{32}$$

3.1.4 Multiple-Relaxation-Time Method

In the model, the Multiple-Relaxation-Time (MRT) method is applied. The MRT method has been developed by Lallemand and Luo [34], and it allows different modes like energy, momentum and viscous stresses to relax independently. This provides the LBM more degrees of freedom, resulting in a greater stability. The relaxation term at the surface can be determined by first defining a parameter ψ dependent on the densities of the fluids, before applying an interpolation scheme. ψ is defined as follows:

$$\psi(\mathbf{x}) = \rho^N(\mathbf{x}) = \frac{\rho_1(\mathbf{x}) - \rho_2(\mathbf{x})}{\rho_1(\mathbf{x}) + \rho_2(\mathbf{x})} \tag{33}$$

The interpolation scheme that is applied using this definition of ψ is given below. This interpolation scheme ensures that the relaxation time changes smoothly over the interface between the two fluids.

$$\tau(\mathbf{x}) = \begin{cases} \tau_1 & \psi > \delta \\ g_1(\psi) & \delta \geq \psi > 0 \\ g_2(\psi) & 0 \geq \psi \geq -\delta \\ \tau_2 & \psi < -\delta \end{cases} \tag{34}$$

In the scheme given above in equation 34, $g_1(\psi)$ and $g_2(\psi)$ are given by:

$$\begin{cases} g_1(\psi) = s_1 + s_2\psi + s_3\psi^2 \\ g_2(\psi) = s_1 + s_2\psi + s_3\psi^2 \end{cases} \tag{35}$$

With s_{1-3} and t_{1-3} given by:

$$\begin{cases} s_1 = t_1 = 2 \cdot \frac{\tau_1\tau_2}{\tau_1+\tau_2} \\ s_2 = 2 \cdot \frac{\tau_1-\tau_2}{\delta} \\ s_3 = -\frac{s_2}{2\delta} \\ t_2 = 2 \cdot \frac{t_1-\tau_2}{\delta} \\ t_3 = \frac{t_2}{2\delta} \end{cases} \tag{36}$$

Here, $\delta \leq 1$ is a positive parameter by which the interface thickness can be set. The dynamic viscosities of the fluids are given by equation 37 below, with $c_s^2 = \frac{c^2}{3}$ where c is the lattice speed, normally set to 1 in this model.

$$\nu_k = c_s^2(\tau_k - 0.5)\Delta t \tag{37}$$

3.1.5 Redistribution Step

The last phase of an iteration in the model is the redistribution, also called the *recoloring step*. The immiscible character of the fluids investigated in this research needs to be taken into account in the numerical model. That is why the redistribution step is added into the model. This phase redistributes the particles at the interface of the two fluids in such a way that the fluids remain separated and do not mix with each other, by adding a so-called perturbation step. The recoloring step is given for both fluids according equations 38 and 39.

$$f_i^{1,+} = \frac{\rho_1}{\rho} f_i^* + \beta w_i \frac{\rho_1\rho_2}{\rho} \cos(\lambda_i)|\mathbf{e}_i| \tag{38}$$

$$f_i^{2,+} = \frac{\rho_2}{\rho} f_i^* + \beta w_i \frac{\rho_1\rho_2}{\rho} \cos(\lambda_i)|\mathbf{e}_i| \tag{39}$$

Here, β is a parameter that determines the interface thickness, which can take any value between 0 and 1. After the recoloring step, $f_i^{1,+}(\mathbf{x}, t)$ and $f_i^{2,+}(\mathbf{x}, t)$ become the new starting PDF's for a new streaming step. The procedure illustrated above will be iterated over many time steps, simulating a two-phase flow.

3.2 Model Parameters

The flow patterns created by the simulations of two-phase flow in a T-junction microchannel in this research, will be compared to the results already obtained by Zheng Liu for a Y-shaped channel. therefore, the same model parameters for the fluids and partly for the channel will be used as by Zheng Liu.

First of all, the physical channel width of the system is chosen to be $d = 100\mu m$. In the Lattice Boltzmann Model, a channel width of 10 lattice nodes has been chosen, i.e. $d^* = 10$ lu. The lenght of the main channel in the model is chosen as $L^*_{channel} = 200$ lu, and the inlet length is $L^*_{inlet} = 45$ lu.

The fluids that are simulated in this research are water and n-heptane. In the simulation, water flows through the blue inlet en n-heptane through the red inlet. The fluid properties that this research relies on are given below in table 1. In this table, the chosen values for α and β are also displayed. As mentioned before, the value of β should be between 0 and 1. A lower value for β results in a more stable model, however it also increases the interface between the two fluids, making it more diffusive which decreases the accuracy of the model. therefore, β has been chosen to be 0.7 in this model.

Fluid	Viscosity ν (m^2/s)	Density (g/cm^3)	Interfacial Tension Coefficient (mN/m)	Contact Angle (°)	Relaxation Time (t.s.)	α	β
Water	10^{-6}	1		43.4	0.5244	5/9	0.7
n-Heptane	$5.676 \cdot 10^{-7}$	0.68	50.1		0.5139	3/9	0.7

Table 1: Overview of fluid properties of the simulated fluids Water and n-Heptane

As can be seen from Table 1, the contact angle has been set to 43.4°, meaning the channel walls are more wetting for water compared to n-heptane. Furthermore, the parameter σ which sets the interfacial tenstion between the fluids is chosen to be a constant value of $\sigma = 0.05$.

3.3 Boundary Conditions

In the simulation of the fluids, four boundary conditions are applied. In this section, these boundary conditions are discussed and explained.

3.3.1 Walls

The first boundary condition (BC) has already been mentioned before in section 3.1.2. It is called the *no-slip boundary condition* and it is applied at the walls of the channel. The no-slip BC states that the fluid particles near the wall must have the same velocity as the wall itself. In order to measure up to this condition, particles that flow from a fluid node to a solid node are bounced back to the fluid node they came from. There are two possible ways this bounce-back step can be performed. The first option is called the *fullway bounce back* method and the second option is the *halfway bounce back* method.

The fullway bounce back method takes the most time of the two options, namely two time steps. When this method is applied, a particle first flows all the way from the fluid node to the solid node. From there, it will be returned to the fluid node it originally came from during the next collision step. [15]

When the halfway bounce back method is applied, a particle starts streaming from the fluid node to the solid node at time $t = 0$. At time $t = \Delta t/2$, when it is travelled halfway, it returns directly to the fluid node it came from. During this bounceback procedure, it takes the particle only one time step to be bounced back. This method is therefore more time efficient, which is why it is chosen over the fullway bounce back method to be used in this research. The streaming step at solid nodes is then calculated according equation 40

$$f_i(x_b, t + \Delta t) = f_i^*(x_b, t) \tag{40}$$

Here, x_b is the position of the fluid node involved in the bounce back step. In figure 7, a schematic overview of the two bounce back method is given. In the figure, X_N represents the fluid node from where the particle starts streaming, and x_{N+1} represent the solid node. The surface of the channel wall is located halfway between the two nodes at the dashed line.

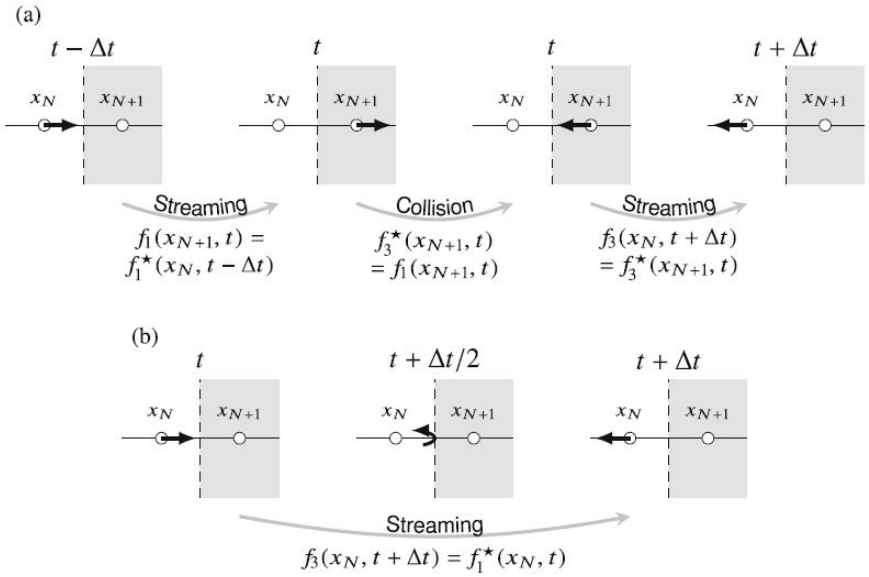


Figure 7: Different bounce back methods applicable to LBM: a) Fullway Bounce Back b) Halfway Bounce Back [15]

3.3.2 Inlet

At both inlets of the channel, the fluids flow into the channel at a velocity $\mathbf{u}$. In order to simulate this, the method created by Ladd is used. [29] This method is an extension on the bounce back method, with the difference that the walls do not have a velocity of zero, but move with velocity $\mathbf{u}_w$. This is why this method is called the *moving walls method*. This method likens the two fluid inlets to walls moving at the same velocity as the desired inlet flow velocity. This way, the moving walls 'push' the fluids in with the desired velocity as it were. The formula describing the moving walls method is given by equation 41 below.

$$f_i(x_b, t + \Delta t) = f_i^*(x_b, t) - w_i \rho_w \frac{\mathbf{e}_i \cdot \mathbf{u}_w}{c_s^2} \tag{41}$$

Here, ρ_w and $\mathbf{u}_w$ are the wall density and wall velocity respectively. Usually, ρ_w is set to be the same as the density at $f_i^*(x_b, t)$. Furthermore, $\mathbf{u}_w$ is set to the desired inlet flow velocity.

3.3.3 Outlet

At the outlet of the channel, the so called *convective boundary condition* (CBC) is applied. Lou et al investigated the performance of three different outlet BC's (the extrapolation, Neumann and convective boundary condition). [37] They found the convective BC to deliver the best results on accuracy and stability, which is why the CBC is chosen as outlet BC for this research. The CBC is given at $x = N$, with N being the location of the outlet, by the following equation:

$$\frac{\partial \chi}{\partial t} + U \frac{\partial \chi}{\partial x} = 0 \tag{42}$$

In equation 42, χ can refer to any variable and U is the velocity normal to the boundary. The value of U can be determined in three ways. First of all it can be the maximum velocity at the nodes directly near the boundary, secondly it can be the local velocity at the boundary nodes, or it can be the average velocity at the $(N-1)$th layer near the boundary. Lou et al compared these options in their research and found the most accurate results using the averaging scheme for U. [37] This is why this scheme for U is used in this research as well. U is therefore calculated according equation 43 below, where M+1 are the number of nodes in the $(N-1)$th layer.

$$U_{ave}(t) \equiv \frac{1}{M+1} \sum_j u(N-1, j, t) \tag{43}$$

The CBC in the mesoscopic scale is expressed according equation 44, where $\lambda = U(t + \delta t)\delta t / \delta x$:

$$f_i(N, j, t + \delta t) = \frac{f_i(N, j, t) + \lambda f_i(N - 1, j, t + \delta t)}{1 + \lambda} \tag{44}$$

3.3.4 Contact Angle

The fourth and final BC that is included in the model is the contact angle of the fluids with the walls. The contact angle θ is enforced on the walls throug equation 45, composed by Ding and Spelt [18]:

$$\mathbf{n}_w \cdot \nabla \rho^N = -\Theta_w \left| \mathbf{t}_w \cdot \nabla \rho^N \right| \tag{45}$$

Where Θ_w is defined as

$$\Theta_w = \tan\left(\frac{\pi}{2} - \theta\right) \tag{46}$$

In equation 45, $\mathbf{n}_w$ represents the normal vector to the wall, pointed toward the fluids, and $\mathbf{t}_w$ is the tangential unit vector to the wall. As an example, we consider the case of a bottom wall, located in between a solid node at $y = 0$ and a fluid node at $y = 1$. Discretization of equation 45 gives [20]:

$$\rho^N_{x,0} = \rho^N_{x,1} + \Theta_w \left| \mathbf{t}_w \cdot \nabla \rho^N \right| \delta_x \tag{47}$$

In order to determine the tangial component of $\nabla \rho^N$, the following extrapolation scheme is used:

$$\mathbf{t}_w \cdot \nabla \rho^N = 1.5\ \partial_x \rho^N \big|_{x,1} - 0.5\ \partial_x \rho^N \big|_{x,2} \tag{48}$$

The partial derivatives in the right side of equation 48 can be determined by the second-order central difference approximation. The steps of equations 47 and 48 are similar for all walls in a channel, the only thing that is to be adjusted is the node indications, dependent on the location of the wall.

3.4 Python Algorithm

In order to simulate the flow of water and n-heptane inside the T-shaped microchannel, Python is used to implement the LBM algorithm. Below, in figure 8, a flow chart is given to illustrate the different steps of the code. The full code used for the simulations can be found in appendix C.

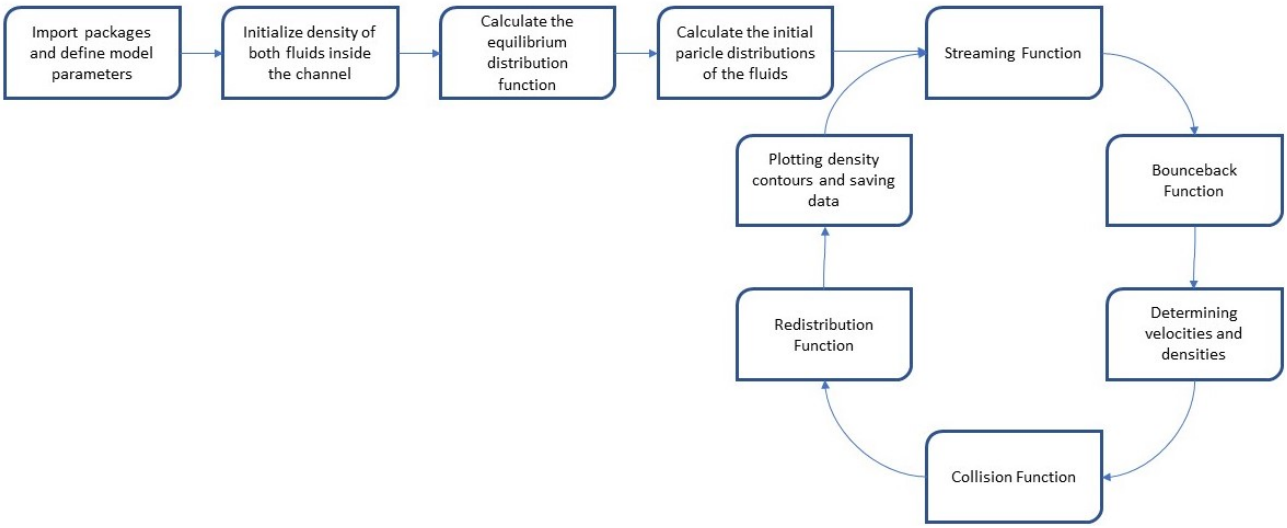


Figure 8: Flow chart illustrating the steps followed by the Python code used for the simulations

4 Results and Discussion

This chapter explains and discusses all results obtained from the simulations of two-phase flow in a T-shaped microchannel of water and n-heptane. Section 4.1 will show some validation results of the model, followed by 4.2 which will give an extensive overview of all results that have been obtained, together with an extensive discussion of the obtained results.

4.1 Poiseuille Flow

In order to validate the RK algorithm used to simulate two-phase flow, a simple example of two-phase Poiseuille flow is worked out using the RK algorithm. A schematic overview of a 2D channel with two immiscible fluids, one wetting and one non-wetting, showing Poiseuille flow is sketched in figure 9. The wetting fluid flows along the channel boundaries while the non-wetting fluid flows at the center of the channel.

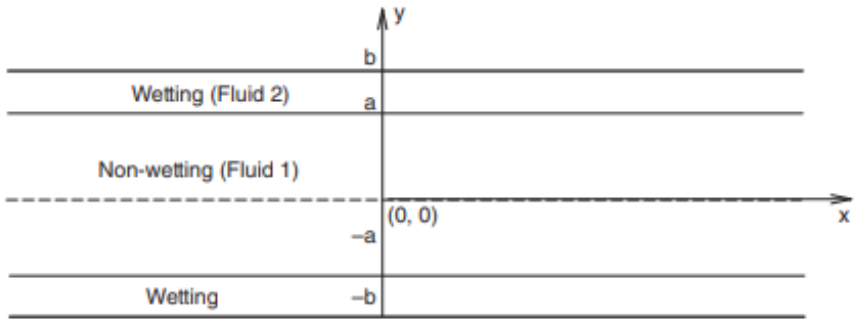


Figure 9: Two-Phase Poiseuille flow of two immiscible fluids in a 2D channel. The non-wetting fluid (fluid 1) flows at the center of the channel while the wetting fluid (fluid 2) flows along the upper and lower wall of the channel.[19]

In order to validate the algorithm, a velocity profile of two-phase Poiseuille flow has been simulated by the RK algorithm and plotted in figure 10. The density ratio and the kinematic viscosity ratio are both set to 3 in the simulation and the parameter β to set the thickness of the interface is set to $\beta = 0.5$. Furthermore, the channel dimensions are 4 lu (length) by 101 lu (width). In addition to the simulation, the velocity profile has been calculated analytically. The analytical solution has also been plotted in figure 10, in order to compare the results.

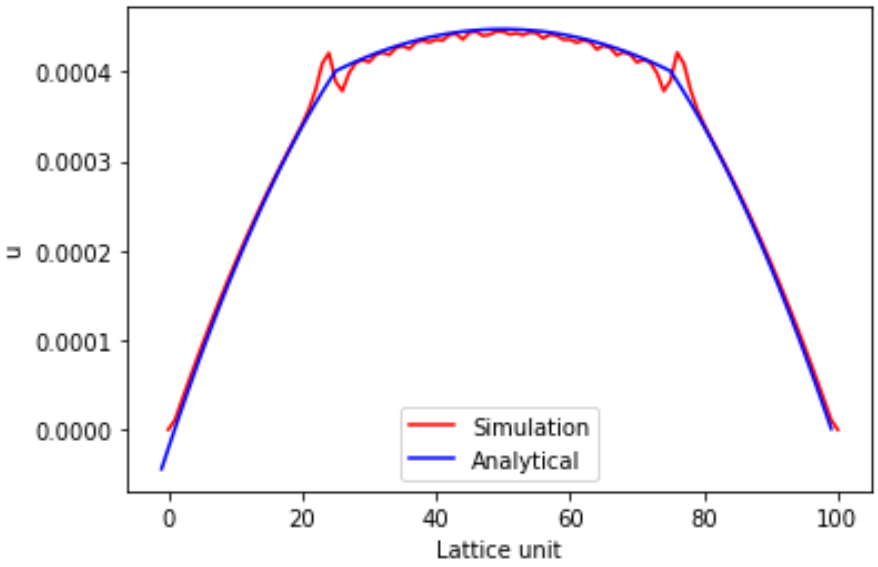


Figure 10: Plot of the analytical and the numerical solution for the velocity profile of two-phase poiseuille flow in a microchannel

As can be seen from the figure, the RK algorithm simulates the two-phase Poiseuille flow for the most part correctly compared to the analytical solution. However, at the interfaces between the fluids, the simulation shows an inaccuracy as the velocity profile shows an unexpected peak at the locations of the interfaces. These velocity 'bumps' are due to the difference in density of the two fluids. [19] This density difference causes a pressure difference at the interface of the fluids. Since the LBM doesn't explicitly track the interfaces of the fluids, these interfaces are diffusive in the simulation. Due to the sudden pressure difference at the interface, the velocity suddenly peaks at the interfaces before stabilizing again. The analytical solution calculates the velocity profile for a sharp interface, which is why that graph shows a fluent transition in the velocity profile at the interfaces.

4.2 Flow Patterns

In order to read out the results from the simulations of the two-phase flow, the density contours of water in the T-channel were plotted during the simulations every 5000 time steps, which is equal to 0.065 seconds in physical units. Prior to the iterations over time, at $t = 0$ the densities of water and n-heptane have been initialized in such a way that the blue inlet and the main channel are filled completely with water. The red inlet is filled completely with n-heptane. Figure 11 shows the density contour after this initialization.

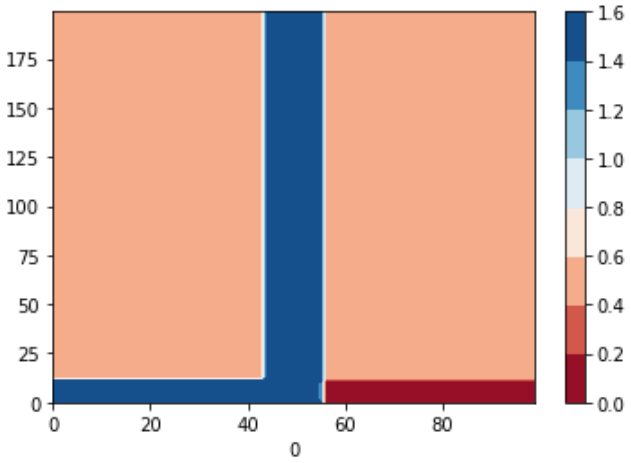
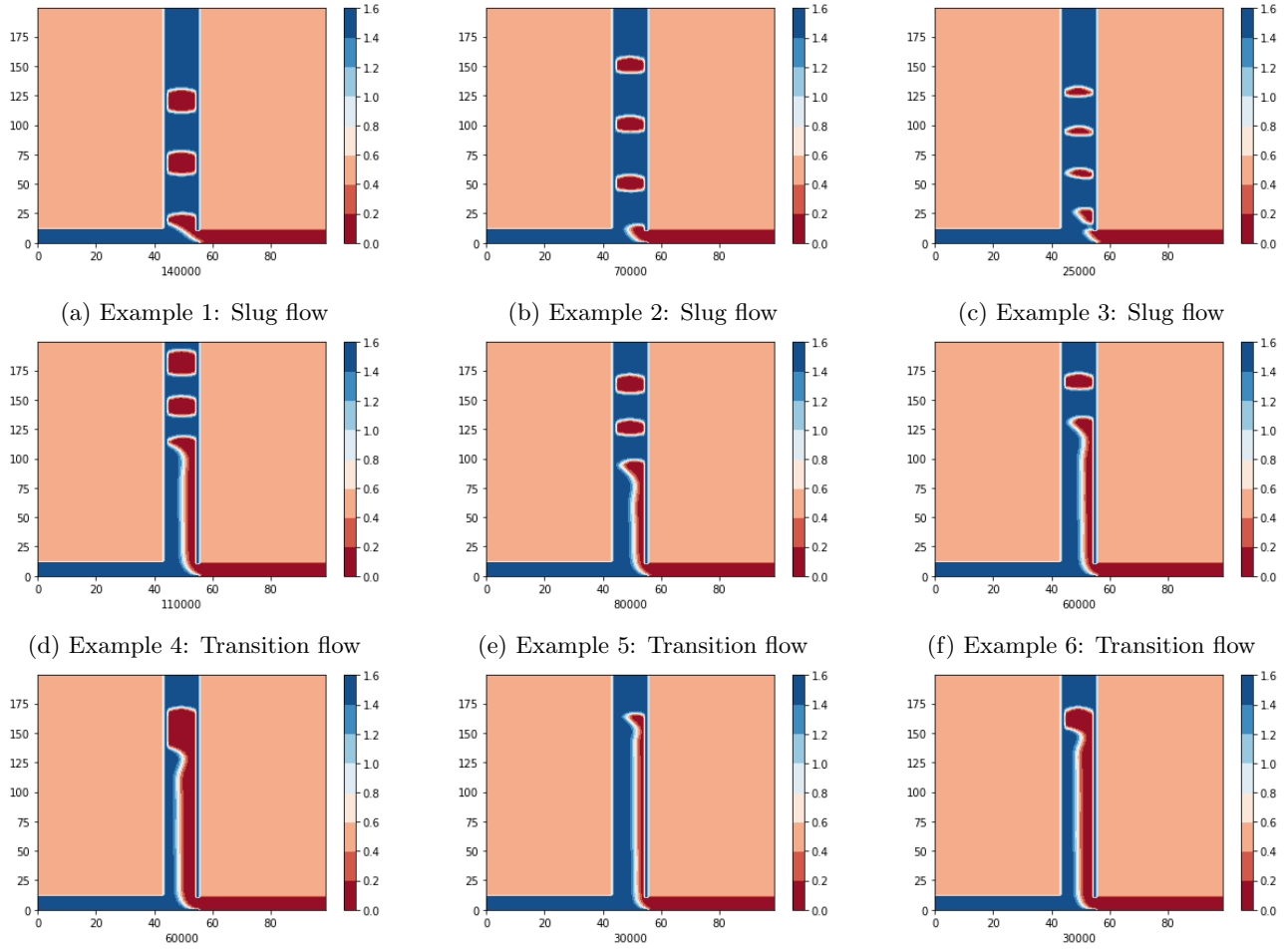


Figure 11: Initialization of the fluids in the microchannel at $t = 0$. The red color represents n-heptane and blue represents water.

4.2.1 Overall Flow Pattern Behaviour

At each simulation, the inlet velocities of water and n-heptane have been adjusted in the initialization step and the corresponding capillary numbers have been determined. Below, in figure 12, a couple of examples are given of the different flow patterns that have been observed during the simulations. The Capillary numbers of both fluids and the time passed corresponding to each example is given thereafter in table 2.



(a) Example 1: Slug flow (b) Example 2: Slug flow (c) Example 3: Slug flow

(d) Example 4: Transition flow (e) Example 5: Transition flow (f) Example 6: Transition flow

(g) Example 7: Parallel flow (h) Example 8: Parallel flow (i) Example 9: Parallel flow

Figure 12: Nine examples of flow patterns found during the simulations of two-phase flow of water and n-heptane in a T-shaped microchannel

Case	Ca_{water}	$Ca_{n-heptane}$	Time (s)	Time Steps (ts)
a	$3.69 \cdot 10^{-5}$	$1.71 \cdot 10^{-4}$	1.82	140 000
b	$6.45 \cdot 10^{-5}$	$4.88 \cdot 10^{-4}$	0.91	70 000
c	$1.20 \cdot 10^{-4}$	$1.22 \cdot 10^{-3}$	0.325	25 000
d	$8.30 \cdot 10^{-5}$	$2.44 \cdot 10^{-4}$	1.43	110 000
e	$8.30 \cdot 10^{-5}$	$3.66 \cdot 10^{-4}$	1.04	80 000
f	$1.11 \cdot 10^{-4}$	$4.88 \cdot 10^{-4}$	0.78	60 000
g	$1.84 \cdot 10^{-4}$	$2.44 \cdot 10^{-4}$	0.78	60 000
h	$1.84 \cdot 10^{-4}$	$1.22 \cdot 10^{-3}$	0.39	30 000
i	$3.23 \cdot 10^{-4}$	$6.10 \cdot 10^{-4}$	0.39	30 000

Table 2: Overview of fluid properties of the simulated fluids Water and n-Heptane corresponding to the examples given in figure 12, along with the time passed at the moment of plotting

As can be seen from the examples in figure 12, an increase of the Capillary numbers of the fluids in the system initially results in a decrease of slug length. Eventually, when both Capillary numbers are high enough, the flow eventually transitions to parallel flow. Figure 12d, 12e and 12f show examples of a flow pattern that occurred in the simulations that is sort of a combination of slug flow and parallel flow. The flow in these cases starts of behaving as slug flow, however when enough time passes the flow in the channel shifts more and more to parallel flow until all slugs are as it were 'pushed out' of the main channel and only parallel flow remains. This type of flow has been named *transition flow* in this research.

In order to analyze and observe the flow pattern behaviour observed in the simulations properly, a flow pattern map has been created of all observed flow patterns. This flow pattern map is shown in figure 13 in the form of a scatterplot, indicating the flow pattern at each combination of Capillary numbers for water and n-heptane ran in the simulations.

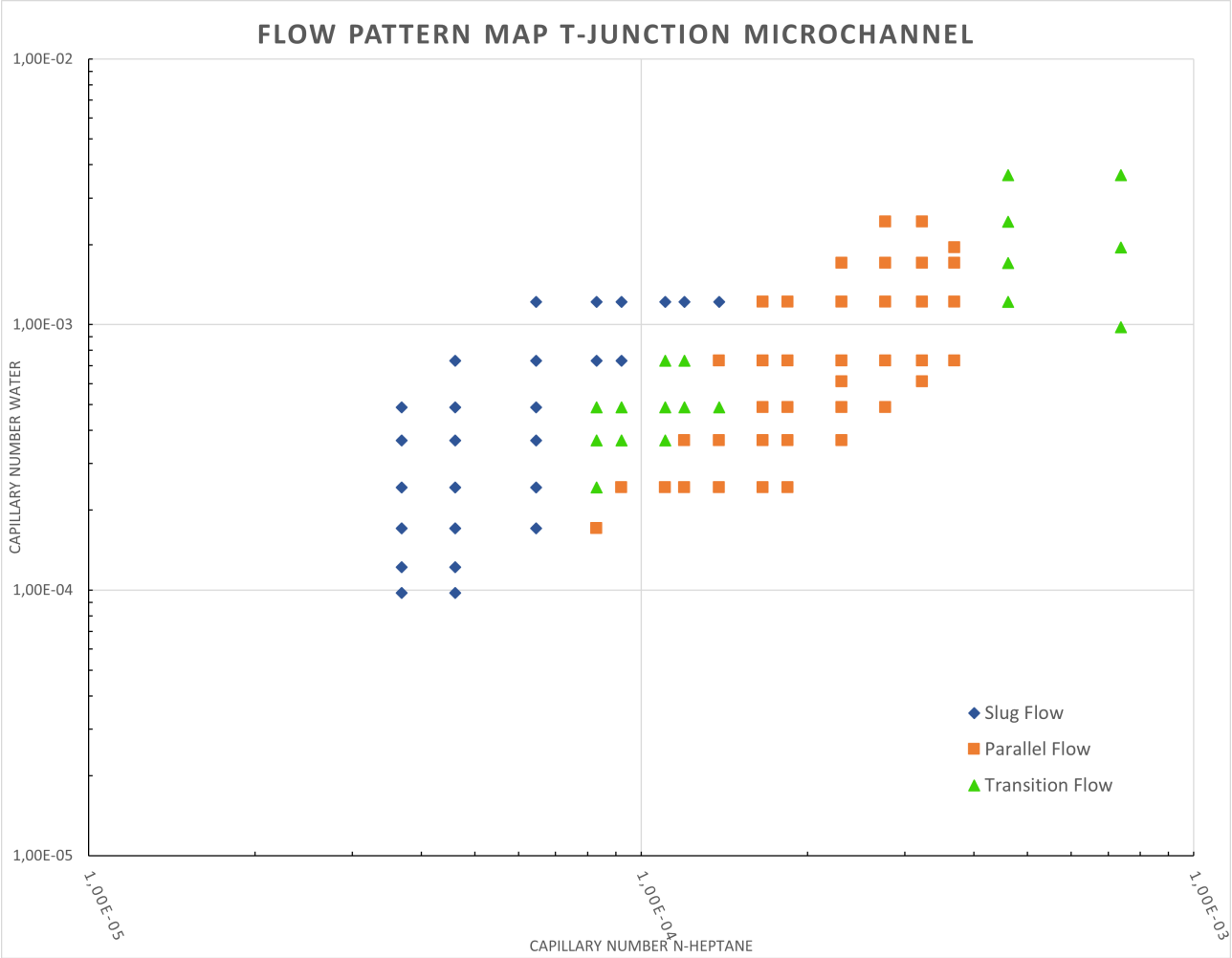


Figure 13: Flow pattern map of all flow patterns found during the flow simulations of water and heptane in a T-shaped microchannel

In order to evaluate the overall flow pattern behaviour of water and n-heptane inside the T-shaped microchannel, and to determine the influence of junction geometry on the flow, the created flow pattern map from figure 13 is compared to the results that have already been obtained by Zheng Liu for a Y-shaped microchannel. The flow pattern map of the simulations by Zheng is shown below in figure 14.

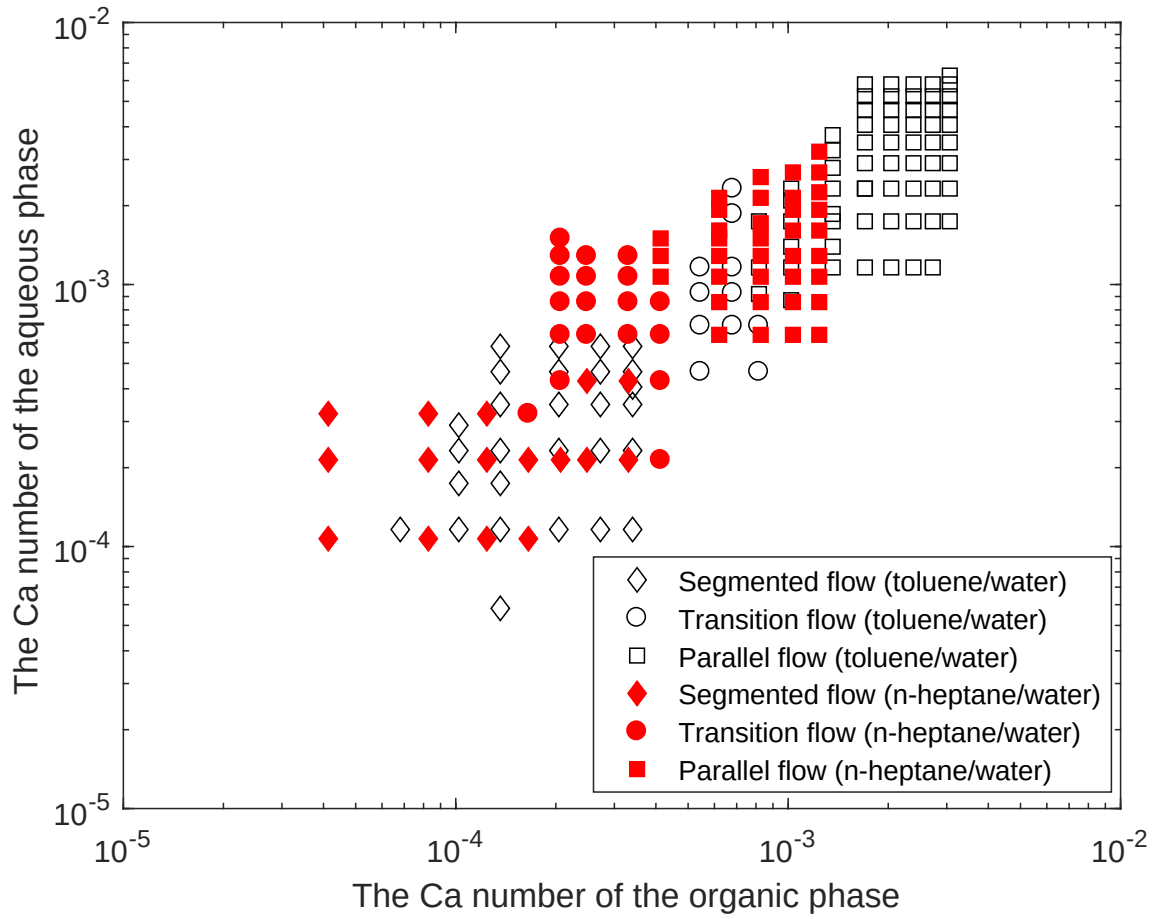


Figure 14: Flow pattern map for two-phase flow in a Y-shaped microchannel obtained by Zheng Liu.

As can be seen by comparing figure 13 and 14, the flow pattern map composed in this research for the T-shaped channel looks similar to the flow pattern map obtained by Zheng. It shows a similar trend concerning the transition from slug flow to parallel flow. However, from figure 13 can be read that the shift to parallel flow starts at lower Capillary numbers in the T-channel compared to the Y-channel. From figure 14 can be read that Zheng observed a transition from slug to parallel flow at Capillary numbers around $3 \cdot 10^{-4}$ for n-heptane and 10^{-3} for water. The simulations performed in this research however showed a transition from slug to parallel flow at Capillary numbers around 10^{-4} for n-heptane and $4 \cdot 10^{-4}$ for water. At first sight, this is a very remarkable result given that Dessimoz et al [14] concluded from their research that a Y-channel showed a greater tendency for parallel flow compared to a T-channel for a system of water and toluene. This could point to some numerical issue in the model causing parallel flow to occur at lower Capillary numbers than expected. However, Yagodnitsyna et al [3] also researched flow patterns inside a T-junction microchannel for three different combinations of immiscible liquids (kerosene-water, Paraffin oil-water and Castor oil-Paraffin oil). As the fluid properties like density, dynamic viscosity and interfacial tension with water of kerosene and n-heptane are all of the same magnitude -unlike those of castor oil and paraffin oil compared to n-heptane-, the flow pattern maps of kerosene-water and heptane-water would be expected to show similar looking results. Yagodnitsyna et al created flow pattern maps for the Weber numbers instead of the Capillary numbers of the fluids. Their results are shown in figure 15.

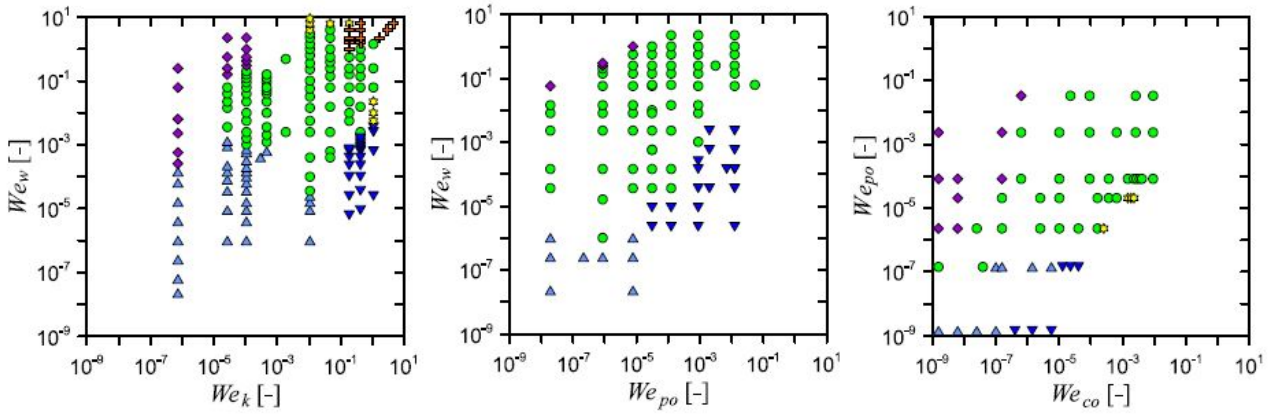


Figure 15: Flow pattern maps observed for three liquid-liquid systems by Yagodnitsyna et al, from left to right: kerosene-water, paraffin oil-water, paraffin oil-caster oil. The purple diamonds represent slug flow and the circles represent parallel flow. [3]

For comparison, the Weber numbers corresponding to the results displayed in figure 13 have been calculated. The flow pattern map from figure 13 has also been plotted for the corresponding Weber numbers, shown below in figure 16:

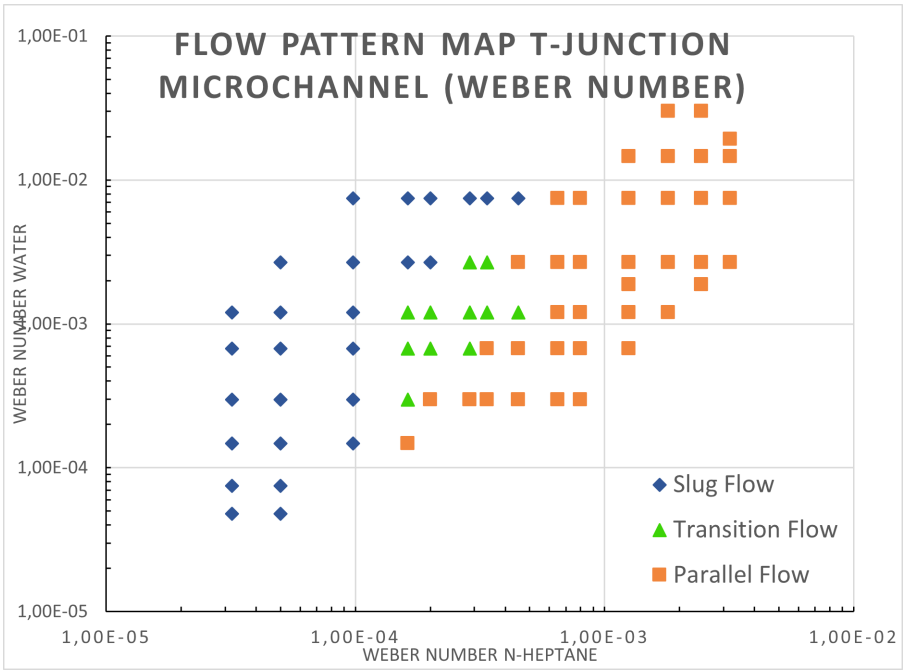


Figure 16: Flow pattern map of the flow patterns observed in the simulations according the corresponding Weber numbers

The flow maps of the experimental results obtained by Yagodnitsyna et al for the kerosene-water system and the simulated results from this research seem to look quite similar. The map is showing a transition from slug to parallel flow at approximately the same Weber numbers compared to this research. This supports the results for flow patterns in a T-channel found in this research. Since the experiments done by Yagodnitsyna et al are performed with different liquid-liquid systems and since they also haven't been performed in a Y-shaped microchannel as well, it is hard to fully verify the conclusion that a T-shaped channel shows greater tendency for parallel flow by its means. However, it does support the outcome of the simulations as the fluid and channel properties for both systems are of the same order of magnitude. Therefore it seems like the simulation technique doesn't show big errors.

4.2.2 Slug Flow

Furthermore, the simulations where slug flow formed in the microchannel have been analyzed to check whether there is any mass leakage occurring in the simulations. Therefore, the ratios $\frac{\text{slug length}}{\text{lenght space in between slugs}}$ have been determined and plotted against the corresponding ratios $\frac{\text{inlet velocity n-heptane}}{\text{inlet velocity water}}$. The plotted figure together with a brief discussion of the results can be found in appendix B of this report.

4.2.3 Transition Flow

The flow pattern named *transition flow* in this work is a very interesting result. Examples of this flow pattern can be seen in figure 12d, 12e and 12f, and it is indicated in the flow pattern map of figure 13 by the green triangles. This flow pattern shows a flow that starts off as slug flow, transforming over time to parallel flow. The parallel flow pushes out the slugs from the microchannel as it were, taking over the main flow pattern. This type of flow pattern draws attention because it doesn't seem to be found before in physical experiments regarding literature. In literature, flow patterns have been found that evolve reversed from the transition flow pattern found in this research. In these cases, the flow starts off to be parallel, breaking up into slugs after some time has passed. There are multiple examples of where this flow pattern occurred. For example, Taotao Fu et al investigated flow patterns of water and cyclohexane inside rectangular microchannels, and found so called *jet flow* as a flow pattern. [46] Cubaud and Mason also found this *jet flow* during their research to the formation of droplets in microfluidic two-phase systems [45] and Kashid et al found a flow pattern they called a *deformed interface* [27] which also looks similar to the jet flow obtained by Cubaud and Mason and by Fu. These jet flows and deformed interface flow are shown below in figure 17.

(a) Jet Flow observed by Fu et al [46]

(b) Jet Flow observed by Cubaud et al [45]

(c) Deformed Interface flow observed by Kashid et al [27]

Figure 17: Three types of flow patterns comparable to transition flow found during the simulations of this research.

However, this type of flow pattern seems to have been found in other numerical simulations of flow in a circular microchannel carried out by Dongliang Sun. [10] They simulated flow based on the volume-of-fluid method and the so called vapor-liquid phase change model. They found a flow pattern which looks quite the same as the transition flow found in this research, where the parallel flow also takes over as time passes. This flow pattern is shown in figure 18. Sun et al also approaches this simulated flow pattern as a kind of transitional flow between slug and parallel flow.

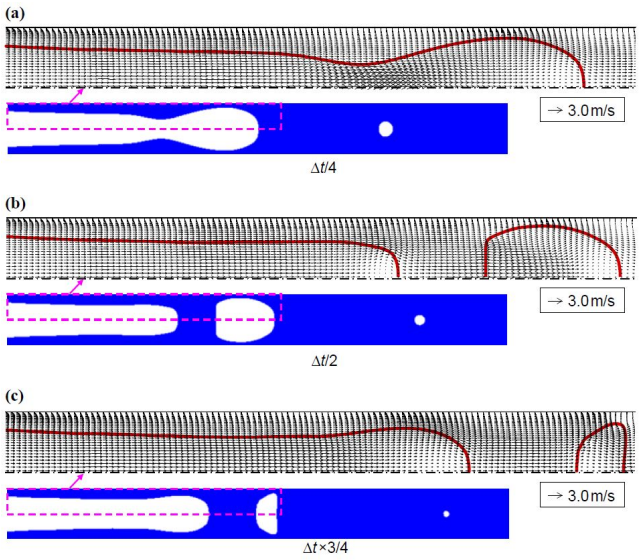
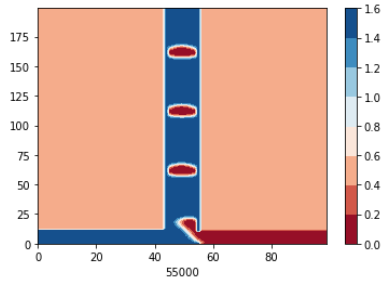


Figure 18: Flow pattern similar found in the research of Sun et al [10], similar looking to the transition flow pattern found in this research

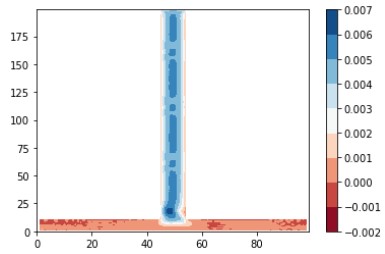
The transition flow pattern does seem to occur more often in numerical simulations of microfluidic two-phase flow. However, it hasn't been discovered yet in experimental research. As there have been performed a lot of studies on flow pattern behaviour in microchannels so far, the most plausible reason for this type of flow pattern to occur in simulations is some kind of numerical stability issue. If this would not be the issue, this type of flow would be a physical phenomenon. In that case, it would be expected that this flow pattern would have been discovered already in experimental research at some point. It would be interesting however to dedicate more research to the subject in order to find out what numerical issue is causing this flow pattern to occur in the simulations and improve the model.

4.2.4 Parallel Flow

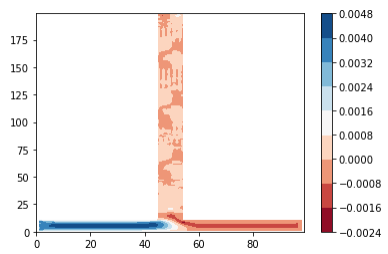
Finally, the parallel flow results will be discussed. As can be seen from figures 12g, 12h and 12i, when parallel flow is formed in the main channel, a slug is formed at the start of the simulation, directly followed by the parallel flow. This slug attachment to the parallel flow has also been found by Anand Sudha in his simulations of a water-heptane system in a Y-shaped microchannel. This is a phenomenon which is not physically correct, as physically would be expected that the flow starts off immediately to be parallel, not preceded by a slug attachment. Besides the non physical slug formation before the parallel flow, the parallel flow is simulated quite properly by the numerical model. In order to determine where the numerical error could be located in the model, a plot of the velocity contours of each type of flow pattern has been made in order to determine whether an error occurs in the velocities in either direction. These velocity contours are shown in figure 19 below. The velocity contours in the x-direction (vertical direction) and the y-direction (horizontal direction) are shown for one case of each flow pattern. The corresponding density contours are also shown as a frame of reference.



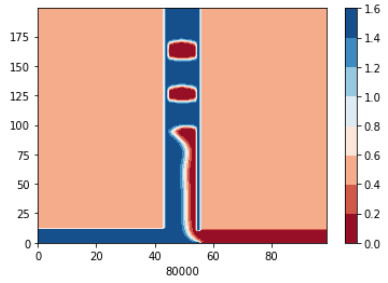
(a) Density contour of slug flow



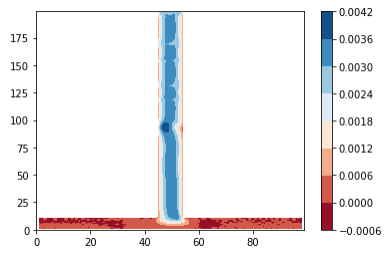
(b) Velocity contour in the x-direction of the slug flow example



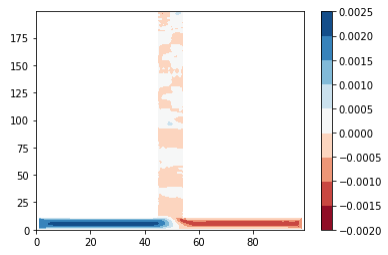
(c) Velocity contour in the y-directino of the slug flow example



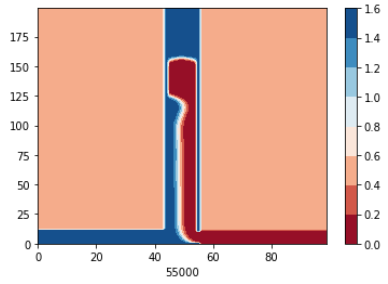
(d) Density contour of transition flow



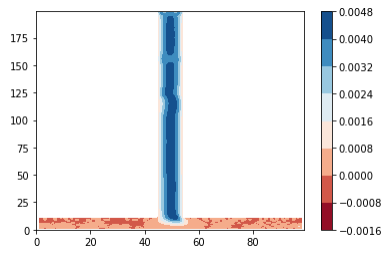
(e) Velocity contour in the x-direction of the transition flow example



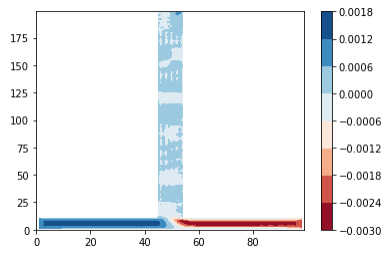
(f) Velocity contour in the y-direction of the transition flow example



(g) Density contour of parallel flow



(h) Velocity contour (x-direction) of the parallel flow example



(i) Velocity contour (y-direction) of the parallel flow example

Figure 19: Velocity contours of the three types of flow patterns, in the x-direction and the y-direction, with the corresponding density plot as a reference.

As can be seen from the velocity contours, there are no sudden increases in velocity visible. Furthermore, the y-velocity (horizontal, alongside the inlet direction) is of neglectable magnitude (10^{-2} times the velocities in the y-direction) in the x-direction, and vice versa for the x-velocity. This all looks quite physically correct, so there doesn't seem to occur an error in the area where the particle velocities are determined.

Since it doesn't seem to be a big error in the model, the slug attachment could be due to the fact that the LBM algorithm needs some time to develop a velocity profile at the inlet junction. Whereas the slug attachment eventually moves out of the main channel after some time has passed, this observation can be considered as a numerical side effect rather than an error in the model.

5 Conclusion

In this research, the effect of junction geometry on the created flow pattern in a microchannel has been investigated. Also, the accuracy of the Lattice Boltzmann Color-Gradient RK model has been analyzed for simulation of two-phase flow in T-shaped and Y-shaped microchannels. Hence, at the beginning of this research the following research questions have risen:

How does the inlet junction geometry of a microchannel affect the created flow pattern of two immiscible fluids?

1. Does a correlation exist between the flow pattern and the Capillary numbers of the fluids?
2. How do the flow patterns created in a T-shaped microchannel differ from the flow patterns created in a Y-shaped microchannel?
3. Do the computational results of this research match the experimental results found in literature?

First of all can be concluded that the geometry of the inlet junction does seem to affect the created flow pattern. As can be concluded from the flow pattern maps from figure 13 for a T-shaped channel and figure 14, the transition from slug flow to parallel flow seems to occur at lower Capillary numbers for water and n-heptane compared to the Y-shaped channel. From this observation, the T-shaped channel seems to show a greater tendency for parallel flow. Thereby can be concluded from figures 13 and 14 that a similar relation exists between the created flow patterns in the microchannel and the capillary numbers of the fluids. Both channel geometries result in slug flow at low capillary numbers, while parallel flow is obtained at higher capillary numbers. Furthermore, from the simulations ran in this research for the T-channel can be concluded that lower capillary numbers of the fluids result in greater slug lengths. An increase of capillary numbers first results in smaller slug lengths, before transitioning the flow pattern to parallel flow.

Secondly, the Lattice Boltzmann Model used in this research to simulate the flow patterns of water and n-heptane inside a T-shaped microchannel seems to give very plausible results overall. The ratio $\frac{\text{Slug length}}{\text{Inter-slug length}}$ versus the ratio $\frac{\text{Inlet velocity n-heptane}}{\text{Inlet velocity water}}$ that are found from the simulations are about equal to each other, which shows there is no mass leakage occurring in the simulations. Furthermore, the slug lengths of slugs formed in one simulation are all equal to each other which is also to be physically expected. Also, the model results in simulations of parallel flow at higher capillary numbers and in slug flow patterns at low capillary numbers which also agrees with physical experiments described in literature. The flow pattern map for corresponding Weber numbers to the simulations shows a transition from slug to parallel flow at Weber numbers of the same order of magnitude as found in experimental research by Yagodnitsyna et al for different liquid/liquid systems. Given the conclusions mentioned above, the overall accuracy of the LBM used in this research seems to be quite decent.

However, the model does show some flaws in the simulated flow patterns. The transition flow mentioned before in this research seems to be a non-physical flow pattern, indicating there is some numerical instability or error occurring in the model. Furthermore, the simulations form a slug attachment prior to parallel flow in the channel which is also a non-physical phenomenon. However, as this seems to be a numerical side effect rather than an error, this is not the biggest problem appearing in the numerical model.

6 Recommendations

Based on the research performed in this work, a couple of topics have been discovered in order that require further research. These recommendations will be explained below.

First of all, as mentioned before, the transition flow pattern found in this research is an interesting topic to focus more research on. This research could not verify whether the transition flow pattern is a physically correct result, or if it is found in the simulations due to some error or inaccuracy in the numerical model. However, the latter option is more likely to be the case, therefore it would be recommendable to focus more in-depth research on this type of flow in the simulation to find out if there is any numerical instability causing the flow to behave this way in the simulation. If no error is found, it would be recommendable to perform some physical experiments on this topic in order to verify whether the transition flow might be a proper physical phenomenon.

Secondly, in order to acquire more detailed information on two-phase flow regimes and the effect of junction geometry on this, it would be advisable to also look at flow pattern behaviour in the entire microchannel, complete with T-junction inlet and outlet. As this research only focused on the effect of inlet junction geometry on the obtained flow pattern, no information on the effect of outlet geometry on the phase separation have been gained. However, the geometry of the outlet is of great importance on the separation of the two fluids. Therefore, it would be best if the entire T-channel complete with inlet and outlet would be simulated and compared to a Y-channel, in order to collect more data on how the junction geometry on the outlet affects the phase separation. A lot of useful information could be collected this way on what geometry separates the phases more cleanly.

At last, further research to the effect of inlet junction geometry would be recommended. From this research can be concluded that a T-shaped channel is more favourable for obtaining parallel flow compared to a Y-shaped channel. However, it could be possible that other geometries are even more favourable for parallel flow. For example, research could be done to the effect on the obtained flow patterns when the angle between the inlet channels is even higher than the 180°of a T-channel. Another option is to study the effect on the obtained flow pattern in a T-shaped microchannel with inlet channels perpendicular to each other.

References

[1] J.M. Martínez-Ageitos A. Soto A. Arce, A. Marchiaro. Citrus essential oil deterpenation by liquid-liquid extraction. *The Canadian Journal of Chemical Engineering*, pages 366–370, 2008.

[2] N. Evseev A. Demianov, O. Dinariev. Density functional modelling in multiphase compositional hydrodynamics. *The Canadian Journal of Chemical Engineering 89*, pages 206–226, 2011.

[3] A.V. Bilsky A.A. Yagodnitsyna, A.V. Kovalev. Flow patterns of immiscible liquid-liquid flow in a rectangular microchannel with t-junction. *Chemical Engineering Journal*, pages 547–554, 2016.

[4] S. Zaleski-G. Zanetti A.K. Gunstensen, D.H. Rothman. Lattice boltzmann model of immiscible fluids. *Physical Review*, page 4320, 1991.

[5] H. Willaime C.N. Baroud. Multiphase flows in microfluidics. *Comptes Rendus Physique*, pages 547–555, 2004.

[6] B.D. Nichols C.W. Hirt. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics 39*, pages 201—225, 1979.

[7] J. Indekeu et al D. Bonn, J. Eggers. Wetting and spreading. *Reviews of Modern Physics*, pages 739–804, 2009.

[8] G. Biswas F. Durst D. Gerlach, G. Tomar. Comparison of volume-of-[U+FB02]uid methodsfor surface tension-dominant two-phase [U+FB02]ows. *International Journal of Heat and Mass Transfer 49*, pages 740–754, 2006.

[9] K. Eggert D. Grunau, S. Chen. A lattice boltzmann model for multiphase fluid flows. *Physics of Fluids A: Fluid Dynamics 5*, pages 2557–2562, 1993.

[10] P. Ding D. Sun, J. Xu. Numerical research on relationship between flow pattern transition and condensation heat transfer in microchannel. *Engineering Computations: International Journal for Computer-Aided Engineering and Software*, pages 0264–4401, 2013.

[11] P. Angeli et al D. Tsaoulidis, V. Dore. Flow patterns and pressure drop of ionic liquid–water two-phase flows in microchannels. *International Journal of Multiphase Flow*, pages 1–10, 2013.

[12] M. Darekar and K.K. Singh et. al. Liquid−liquid two-phase flow patterns in y-junction microchannels. *Industrial Engineering Chemistry Research*, pages 12215–12226, 2017.

[13] A. de Schepper. Liquid-liquid extraction of germanium by lix 63. *Hydrometallurgy*, pages 291–298, 2005.

[14] A.L. Dessimoz, L. Cavin, A. Renken, and L. Kiwi-Minsker. Liquid–liquid two-phase flow patterns and mass transfer characteristics in rectangular glass microreactors. *Chemical Engineering Science*, pages 4035–4044, 2007.

[15] T. Krüger et. al. *The Lattice Boltzmann Method: Principles and Practice*. Springer International Publishing, 2017.

[16] R.J. Good. Contact angle, wetting, and adhesion: a critical review. *Journal of adhesion science and technology*, pages 1269–1302, 1992.

[17] S. Goyal, A.V. Desai, and R.W. Lewis et al. Thiolene and sifel-based microfluidic platforms for liquid–liquid extraction. *Sensors and Actuators B: Chemical*, pages 634–644, 2013.

[18] P.D.M. Spelt H. Ding. Onset of motion of a three-dimensional droplet on a wall in shear flow at moderate reynolds numbers. *Journal of Fluid Mechanics*, pages 341–362, 2008.

[19] XY. Lu H. Huang, M. Sukop. *Multiphase Lattice Boltzmann Methods: Theory and Application*. Wiley Blackwell, 2015.

[20] N. Wang G. Xi H. Liu, Y. Ju. Lattice boltzmann modeling of contact angle and its hysteresis in two-phase flow with large viscosity difference. *Physical Review*, 2015.

[21] M. Care A. Hollis I. Halliday, R. Law. Improved simulation of drop dynamics in a shear flow at low reynolds and capillary number. *Physical Review*, page 056708, 2006.

[22] X. Yin J. Huang, F. Xiao. Lattice boltzmann simulation of pressure-driven two-phase flows in capillary tube and porous medium. *Computers and Fluids*, pages 134–145, 2017.

[23] C. Liu J. Tu, G.H. Yeoh. *Computational Fluid Dynamics: A Practical Approach.* Butterworth-Heinemann, 2018.

[24] E.G. Puckett J.E. Pilliod Jr. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics 199*, pages 465—502, 2004.

[25] K.F. Jensen J.G. Kralj, H.R. Sahoo. Integrated continuous microfluidic liquid–liquid extraction. *Lab on a chip*, pages 256–263, 2007.

[26] R.L. Street J.H. Ferziger, M. Perić. *Computational Methods for Fluid Dynamics.* Springer International Publishing, 2020.

[27] M. Kashid and L. Kiwi-Minsker. Quantitative prediction of flow patterns in liquid–liquid flow in microcapillaries. *Chemical Engineering and Processing: Process Intensification*, pages 972–978, 2011.

[28] A. van den Berg L. Shui, J.C.T. Eijkel. Multiphase flow in microfluidic systems – control and applications of droplets and interfaces. *Advances in Colloid and Interface Science*, pages 35–49, 2007.

[29] A.J.C. Ladd. Numerical simulations of particulate suspensions via a discretized boltzmann equation. part 1. theoretical foundation. *Journal of Fluid Mechanics*, pages 285–309, 1994.

[30] L.Sang, Y. Hong, and F. Wang. Investigation of viscosity effect on droplet formation in t-shaped microchannels by numerical and analytical methods. *Microfluid Nanofluid*, pages 621–635, 2009.

[31] P. Smereka S. Osher M. Sussman, E. Fatemi. An improved level set method for incompressible two-phase flows. *Computers Fluids 27*, pages 663—680, 1998.

[32] J. Sanchez M. Younas, S. Druon$_B$ocquet.$Kineticanddynamicstudyofliquid\check{}liquidextractionofcopperinahfmc$: $Experimentation, modeling, andsimulation.AmericanInstituteofChemicalEngineersJournal, pages1469-$ $-1480, 2009.$

[33] H. Hustedt M.R. Kula, K.H. Kroner. Purification of enzymes by liquid-liquid extraction. *Advances in Biochemical Engineering*, pages 74–117, 2005.

[34] L.S. Luo P. Lallemand. Theory of the lattice boltzmann method: Dispersion, dissipation, isotropy, galilean invariance, and stability. *Physical Review*, 2000.

[35] N. Syna et al P. Martini, A. Adamo. Perspectives on the use of liquid extraction for radioisotope purification. *Molecules*, pages 334–351, 2019.

[36] Y. Amini P.F. Jahromi, J. Karimi-Sabet. Ion-pair extraction-reaction of calcium using y-shaped microfluidic junctions:an optimized separation approach. *Chemical Engineering Journal*, 2017.

[37] B. Shi Q. Lou, Z. Guo. Evaluation of outflow boundary conditions for two-phase lattice boltzmann equation. *Physical Review*, page 063301, 2013.

[38] D. Qian and A. Lawal. Numerical study on gas and liquid slugs for taylor flow in a t-junction microchannel. *Chemical Engineering Science*, pages 7609–7625, 2006.

[39] Jy. Qian, Xj. Li, Z. Wu, and Zj Jin et al. Slug formation analysis of liquid-liquid two-phase flow in t-junction microchannels. *Journal of Thermal Science and Engineering Applications*, 2018.

[40] Jy. Qian, Xj. Li, Z. Wu, and Zj Jin et al. A comprehensive review on liquid–liquid two-phase flow in microchannel: flow pattern and mass transfer. *Microfluidics and Nanofluidics*, 2019.

[41] R. Raj, N. Mathur, and V.V. Buwa. Numerical simulations of liquid-liquid flows in microchannels. *Industrial Engineering Chemistry Research*, pages 10606–10614, 2010.

[42] D. Martnez W. Matthaeus S. Chen, H. Chen. Lattice boltzmann model for simulation of magnetohydrodynamics. *Physical Review*, page 3776, 1991.

[43] J.A. Sethian S. Osher. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics 79*, pages 12—49, 1988.

[44] R.P. Fedkiw S. Osher. Level set methods: An overview and some recent results. *Journal of Computational Physics 169*, pages 463—502, 2001.

[45] T.G. Mason T. Cubaud. Capillary threads and viscous droplets in square microchannels. *Physics of Fluids*, page 053302, 2015.

[46] C. Zhu Y. Ma T. Fu, L. Wei. Flow patterns of liquid–liquid two-phase flow in non-newtonian fluidsin rectangular microchannels. *Chemical Engineering and Processing*, pages 114–120, 2015.

[47] B. Hasslacher P. Lallemand Y. Pomeau J.P. Rivet U. Frisch, D. d'Humières. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, pages 649–707, 1987.

[48] S. Banerjee V.E. Badalassi, H.D. Ceniceros. Computation of multiphase systems with phase field models. *Journal of Computational Physics 190*, pages 371—397, 2003.

[49] R. Zhang X. He, S. Chen. A lattice boltzmann scheme for incompressible multiphase flow and its application in simulation of rayleigh–taylor instability. *Journal of Computational Physics 152*, pages 642–663, 1999.

[50] H. Chen X. Shan. Lattice boltzmann model for simulating flows with multiple phases and components. *Physical Review 47*, 1993.

[51] Q. Li et al Y. Ba, H. Liu. Multiple-relaxation-time color-gradient lattice boltzmann model for simulating two-phase flows with high density ratio. *Physical Review*, page 023310, 2017.

[52] T. Randall Lee Y. Yuan. Contact angle and wetting properties. *Surface Science Techniques*, pages 3–34, 2013.

[53] Q. Yuan Y. Zhao, G. Chen. Liquid–liquid two-phase flow patterns in a rectangular microchannel. *American Institute of Chemical Engineers Journal*, pages 4052–4060, 2006.

[54] Q. Yuan Y. Zhao, G. Chen. Liquid–liquid two-phase mass transfer in the t-junction microchannels. *American Institute of Chemical Engineers Journal*, pages 3042–3053, 2007.

[55] T. Young. Iii. an essay on the cohesion of fluids. *Philosophical transactions of the royal society of London*, pages 65–87, 1805.

A Unit Conversion of Model Parameters

In order to relate the results from the numerical simulations to the physical world, conversion factors have been determined for the parameters used in the model. For calculating these conversion factors, model parameters are always referred to as the symbol for the physical quantity, with a * added. For example, if the physical length is referred to as x, the model length will be indicated as x*. The conversion factors have been determined as follows.

First of all, a conversion factor for the channel width has been determined. The physical channel width has been chosen to be $L = 100\mu m$, and the width of the channel simulated in the model is chosen to be $L = 10lu$ (length units). The spatial conversion factor can be thus determined as $C_l = \Delta x = \frac{100 \cdot 10^{-6} m}{10 lu} = 10^{-5} m/lu$. The second parameter to be converted is the time t. The time conversion factor C_t can be written as Δt and can be calculated using equation 49. [15]

$$\Delta t = c_s^{*2} \left(\tau^* - \frac{1}{2} \right) \frac{\Delta x^2}{\nu} \tag{49}$$

The lattice speed of sound c_s^{*2} is in this case $\frac{1}{3}$ and the relaxation times are chosen to be $\tau^*_{water} = 0.5244$ and $\tau^*_{n-heptane} = 0.5139$. Furthermore, the kinematic viscosities of water and n-heptane are $\nu_{water} = 10^{-6}\ m^2/s$ and $\nu_{n-heptane} = 5.676 \cdot 10^{-7}\ m^2/s$. Working out equation 49 for either one of the fluids, in this case water is used, gives for the time conversion factor:

$$\Delta t = \frac{1}{3} (0.5244 - 0.5) \frac{(10^{-5})^2}{10^{-6}} = 8.13 \cdot 10^{-7} \tag{50}$$

Now that the conversion factors for time and space are determined, the conversion factors for velocity and kinematic viscosity can be easily determined from these two conversion factors. As the physical unit of kinematic viscosity is m^2/s and the unit of velocity is m/s, the conversion factors for these quantities can be calculated according equations 51 and 52:

$$C_\nu = \frac{\Delta x^2}{\Delta t} \tag{51}$$

$$C_u = \frac{\Delta x}{\Delta t} \tag{52}$$

By working out the upper equations, conversion factors for the kinematic viscosity and the velocity are found to be $C_\nu = 1.23 \cdot 10^{-4}$, and $C_u = 12.3$.

B Mass Leakage Plot

Whenever slug length occurs in the main channel, the ratio between the slug length and the space in between slugs should be equal to the ratio of the inlet velocities of the two fluids. If not, mass leakage is occurring in the model. In order to verify wether the LBM simulates slug flow accurately, the ratios $\frac{\text{slug length}}{\text{lenght space in between slugs}}$ have been determined and plotted against the corresponding ratios $\frac{\text{inlet velocity n-heptane}}{\text{inlet velocity water}}$, together with a trend line. This plot is shown below in figure 20. The plot shows a proper linear relationship between the ratios, meaning no mass leakage is occurring in the simulations.

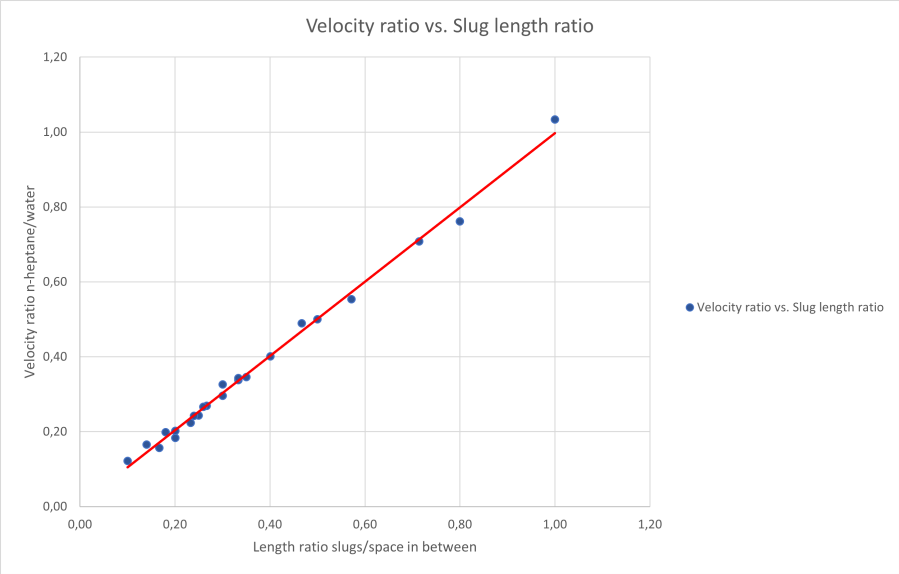


Figure 20: The ratios of slug lenght vs length in between slugs plotted versus the corresponding ratios of the inlet velocities of the fluids, toghether with a trend line to show the relation between the two ratios.

C Python Script

```
#%% Import Packages
import numpy as np
import time
import scipy.io as sio
import numba as nb
import math
import matplotlib.pyplot as plt

#%% parameters
start=time.time() #Determine code execution time
lx=200 #grid dimensions
ly=100
ex= np.array([0.0,1,0,-1,0,1,-1,-1,1]) #D2Q9
ey= np.array([0.0, 0, 1, 0, -1, 1, 1, -1, -1])
cc=1 #lattice speed
csqu= pow(cc,2)/3
#Weights
w1=4/9
w2=1/9
w3=1/36
wk=np.zeros(9)
wk[0]=w1
wk[1:5]=w2
wk[5:]=w3
#rsq is the length of the velocities,ie the direction,diagonal is sqrt(2)
rsq= np.zeros(9)
rsq[0]=0;rsq[1:5]=1;rsq[5:]=np.sqrt(2)
#Bi=np.zeros(9)
#Bi[0]=-4/27
#Bi[1:5]=2/27
#Bi[5:]=1/36
nw=1000 #Dump data every nw time steps
```

```
beta=0.7 #Parameter adjusting interface thickness    Start value: 0.7
sigma= 0.05; #interface tension # Start value 0.05
df= 0.000000015; #body force, if needed
alphar= 5/9; #parameters that determine initial density
alphab= 3/9;
rhori= 1.5; #initial density water
rhobi= 1;   #initial density n-Heptane
tm= 150000; #max time steps
uib= -0.0025 #inlet velocity n-Heptane
uir= 0.0015  #inlet velocity water
#Relaxation time
taur=0.5244             # Water
taub=0.5139             # n-Heptane
#Parameters for calculating pressure
csqr=3*(1-alphar)/5; csqb= 3*(1-alphab)/5;
theta=  math.tan(43.4*np.pi/180) #tan of contact angle
Tw=np.zeros((lx,ly)) #Contact angle matrix
Tw[:,:]= theta
s=np.zeros((9,9))
#MRT matrix
M= [[1.0,1,1,1,1,1,1,1,1],[-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]]
M=np.array(M)


#%% Initialize density and velocity
def initialize(Tw):
    obst= np.zeros((lx,ly)) #wall definition
    ux= np.zeros((lx,ly)) #x velocity
    uy= np.zeros((lx,ly)) #y velocity
    rhor= np.zeros((lx,ly)) #Density of red fluid
    rhob= np.zeros((lx,ly)) #Density of blue fluid
    Uib=np.zeros((12,ly)) #Inlet velocity
    Uir=np.zeros((12,ly))
    Uir[:,:]=10**-8
    Uib[:,:]=10**-8

    # Defining nodes: 0 is fluid node, -1 is empty space, obst >= 1 is solid node
    for x in nb.prange(0,lx):
         for y in nb.prange(0,ly):
            if (x==0): #Top wall
                obst[x,y] = 1
            elif (0 < x < 11):
                if (y == 0):
                    obst[x,y] = 0.5 # Left inlet
                elif (y == ly-1):
                    obst[x,y] = 0.7 # Right inlet
            elif (x == 11):
                if (y < 44 or y > 55):
                    obst[x,y] = 2 # Bottom inlet wall
            elif (11 < x < lx-1):
                if (y == 44):
                    obst[x,y] = 3 # Right wall
                elif (y == 55):
                    obst[x,y] = 4 # Left wall
                elif (44 < y < 55):
                    obst[x,y] = 0
```

```
                else:
                    obst[x,y] = -1 # Empty Space
            elif (x == lx-1):
                if (y == 44):
                    obst[x,y] = 3
                elif (y == 55):
                    obst[x,y] = 4
                elif(44 < y < 55):
                    obst[x,y] = 0.6 # Outlet
                else:
                    obst[x,y] = -1 # Empty Space

    obst[11,44] = 5
    obst[11,55] = 6

    # Defining initial densities
    for x in nb.prange(0,lx):
        for y in nb.prange(0,ly):
            if (x <= 11 and y <= 55):
                rhor[x,y] = rhori
            elif (x <= 11 and y > 55):
                rhob[x,y] = rhobi
            elif (44 <= y <= 55 and x > 11):
                rhor[x,y] = rhori
                rhob[x,y] = 0
            else:               #empty space
                rhor[x,y] = 0.5
                rhob[x,y] = 0.5

    rho=rhor+rhob
    G= np.zeros((lx,ly)) # Gravity term
    G[:,:]=df

    return ux,uy,rhor,rhob,rho,obst,G,Uir,Uib,Tw
[ux,uy,rhor,rhob,rho,obst,G,Uir,Uib,Tw] = initialize(Tw)


#%% Equil Dist. Function
@nb.njit(parallel=True)
def feq(rhor,rhob,ux,uy,alphar,alphab,wk,csqu,ex,ey,lx,ly,obst):
    usqu= ux**2+uy**2
    un= np.zeros((9,lx,ly))
    fer=np.zeros((9,lx,ly))
    feb=np.zeros((9,lx,ly))
    for x in nb.prange(0,lx):
     for y in nb.prange(0,ly):
      for i in nb.prange(0,9,1):
       if (obst[x,y]>=0):
        un[i,x,y]=ex[i]*ux[x,y]+ey[i]*uy[x,y]
        if (i==0):
            fer[i,x,y]=wk[i]*rhor[x,y]*(un[i,x,y]/csqu+ un[i,x,y]**2/(2*csqu**2) -
                   usqu[x,y]/(2*csqu)) +rhor[x,y]*alphar
            feb[i,x,y]=wk[i]*rhob[x,y]*(un[i,x,y]/csqu+ un[i,x,y]**2/(2*csqu**2) -
                   usqu[x,y]/(2*csqu)) +rhob[x,y]*alphab

        elif (i>0 and i<5):
           fer[i,x,y]=wk[i]*rhor[x,y]*(un[i,x,y]/csqu+ un[i,x,y]**2/(2*csqu**2) -
                   usqu[x,y]/(2*csqu)) +rhor[x,y]*(1-alphar)/5
```

```
            feb[i,x,y]=wk[i]*rhob[x,y]*(un[i,x,y]/csqu+ un[i,x,y]**2/(2*csqu**2) -
                    usqu[x,y]/(2*csqu)) +rhob[x,y]*(1-alphab)/5
        else:
             fer[i,x,y]=wk[i]*rhor[x,y]*(un[i,x,y]/csqu+ un[i,x,y]**2/(2*csqu**2) -
                    usqu[x,y]/(2*csqu)) +rhor[x,y]*(1-alphar)/20
             feb[i,x,y]=wk[i]*rhob[x,y]*(un[i,x,y]/csqu+ un[i,x,y]**2/(2*csqu**2) -
                    usqu[x,y]/(2*csqu)) +rhob[x,y]*(1-alphab)/20
    return fer,feb

# Equil. Dist. Fns
[fer,feb]=feq(rhor,rhob,ux,uy,alphar,alphab,wk,csqu,ex,ey,lx,ly,obst)

#Initial probability distribution of particles
fr= fer; fb=feb;
ff= fr+fb; #Overall prob dist.


#%% Streaming Function
@nb.njit(fastmath=True)
def stream(f,lx,ly,obst):
    fs= np.copy(f)
    # Change according to a T-shaped channel
    for x in np.arange(0,lx):
        for y in np.arange(0,ly):
         if(obst[x,y]>=0):
            #Streaming at boundaries
            if ((obst[x,y] == 1) and (y == 0)):
                fs[1,x+1,y] = f[1,x,y]
                fs[2,x,y+1] = f[2,x,y]
                fs[5,x+1,y+1] = f[5,x,y]
            elif ((obst[x,y] == 1) and (0 < y < ly-1)):
                fs[1,x+1,y] = f[1,x,y]
                fs[2,x,y+1] = f[2,x,y]
                fs[4,x,y-1] = f[4,x,y]
                fs[5,x+1,y+1] = f[5,x,y]
                fs[8,x+1,y-1] = f[8,x,y]
            elif ((obst[x,y] == 1) and (y == ly-1)):
                fs[1,x+1,y] = f[1,x,y]
                fs[4,x,y-1] = f[4,x,y]
                fs[8,x+1,y-1] = f[8,x,y]

            elif (obst[x,y] == 0.5):
                fs[1,x+1,y] = f[1,x,y]
                fs[2,x,y+1] = f[2,x,y]
                fs[3,x-1,y] = f[3,x,y]
                fs[5,x+1,y+1] = f[5,x,y]
                fs[6,x-1,y+1] = f[6,x,y]

            elif (obst[x,y] == 0.7):
                fs[1,x+1,y] = f[1,x,y]
                fs[3,x-1,y] = f[3,x,y]
                fs[4,x,y-1] = f[4,x,y]
                fs[7,x-1,y-1] = f[7,x,y]
                fs[8,x+1,y-1] = f[8,x,y]

            elif ((obst[x,y] == 2) and (y == 0)):
                fs[2,x,y+1] = f[2,x,y]
                fs[3,x-1,y] = f[3,x,y]
```

```
    fs[6,x-1,y+1] = f[6,x,y]
elif ((obst[x,y] == 2) and (0 < y < ly-1)):
    fs[2,x,y+1] = f[2,x,y]
    fs[3,x-1,y] = f[3,x,y]
    fs[4,x,y-1] = f[4,x,y]
    fs[6,x-1,y+1] = f[6,x,y]
    fs[7,x-1,y-1] = f[7,x,y]
elif ((obst[x,y] == 2) and (y == ly-1)):
    fs[3,x-1,y] = f[3,x,y]
    fs[4,x,y-1] = f[4,x,y]
    fs[7,x-1,y-1] = f[7,x,y]

elif ((obst[x,y] == 3) and (x < lx-1)):
    fs[1,x+1,y] = f[1,x,y]
    fs[2,x,y+1] = f[2,x,y]
    fs[3,x-1,y] = f[3,x,y]
    fs[5,x+1,y+1] = f[5,x,y]
    fs[6,x-1,y+1] = f[6,x,y]
elif ((obst[x,y] == 3) and (x == lx-1)):
    fs[2,x,y+1] = f[2,x,y]
    fs[3,x-1,y] = f[3,x,y]
    fs[6,x-1,y+1] = f[6,x,y]

elif ((obst[x,y] == 4) and (x < lx-1)):
    fs[1,x+1,y] = f[1,x,y]
    fs[3,x-1,y] = f[3,x,y]
    fs[4,x,y-1] = f[4,x,y]
    fs[7,x-1,y-1] = f[7,x,y]
    fs[8,x+1,y-1] = f[8,x,y]
elif ((obst[x,y] == 4) and (x == lx-1)):
    fs[3,x-1,y] = f[3,x,y]
    fs[4,x,y-1] = f[4,x,y]
    fs[7,x-1,y-1] = f[7,x,y]

elif (obst[x,y] == 0.6):
    fs[2,x,y+1] = f[2,x,y]
    fs[3,x-1,y] = f[3,x,y]
    fs[4,x,y-1] = f[4,x,y]
    fs[6,x-1,y+1] = f[6,x,y]
    fs[7,x-1,y-1] = f[7,x,y]

elif (obst[x,y] == 5):
    fs[1,x+1,y] = f[1,x,y]
    fs[2,x,y+1] = f[2,x,y]
    fs[3,x-1,y] = f[3,x,y]
    fs[4,x,y-1] = f[4,x,y]
    fs[5,x+1,y+1] = f[5,x,y]
    fs[6,x-1,y+1] = f[6,x,y]
    fs[7,x-1,y-1] = f[7,x,y]

elif (obst[x,y] == 6):
    fs[1,x+1,y] = f[1,x,y]
    fs[2,x,y+1] = f[2,x,y]
    fs[3,x-1,y] = f[3,x,y]
    fs[4,x,y-1] = f[4,x,y]
    fs[6,x-1,y+1] = f[6,x,y]
    fs[7,x-1,y-1] = f[7,x,y]
    fs[8,x+1,y-1] = f[8,x,y]
```

```
            else:
                yn= y % (ly-1) + 1
                if (y == ly-1):
                      yn = 0

                xe= x%(lx-1)+1
                if (x == lx-1):
                    xe = 0

                ys = ly - 1 - (ly-y) % (ly)
                xw = lx - 1 - (lx-x) % (lx)
        #Streaming Step at interior
                fs[1,xe,y] = f[1,x,y];
                fs[2,x,yn] = f[2,x,y];
                fs[3,xw,y] = f[3,x,y];
                fs[4,x,ys] = f[4,x,y];
                fs[5,xe,yn]= f[5,x,y];
                fs[6,xw,yn]= f[6,x,y];
                fs[7,xw,ys]= f[7,x,y];
                fs[8,xe,ys]= f[8,x,y];


    for x in np.arange(0,lx):
        for y in np.arange(0,ly):
    #Propagation
            f[1:,x,y]=fs[1:,x,y]

    return f


#%% Bounceback function
@nb.njit(fastmath=True)#(parallel=True)
def bounceback1(f,ff,obst,fe,fc,uib,uir,ex,ey,ux,uy,wk,rho,csqu,tau):
   for x in nb.prange(0,lx):
       for y in nb.prange(0,ly):

            #Inlet, Kruger Equation 5.26
            if (obst[x,y] == 0.5):
                f[2,x,y+1] = fc[4,x,y+1] - (2*wk[4]*rho[x,y+1]*ey[4]*uir)/csqu
                f[5,x+1,y+1]=fc[7,x+1,y+1] - (2*wk[7]*rho[x+1,y+1]*ey[7]*uir)/csqu
                f[6,x-1,y+1]=fc[8,x-1,y+1] - (2*wk[8]*rho[x-1,y+1]*ey[8]*uir)/csqu
            elif (obst[x,y] == 0.7):
                f[4,x,y-1] = fc[2,x,y-1] - (2*wk[2]*rho[x,y-1]*ey[2]*uib)/csqu
                f[7,x-1,y-1] = fc[5,x-1,y-1] - (2*wk[5]*rho[x-1,y-1]*ey[5]*uib)/csqu
                f[8,x+1,y-1] = fc[6,x+1,y-1] - (2*wk[6]*rho[x+1,y-1]*ey[6]*uib)/csqu

            # BB for walls, change for T channel
            elif ((obst[x,y] == 1) and (y == 0)):
                f[5,x+1,y+1] = fc[7,x+1,y+1]
            elif ((obst[x,y] == 1) and (0 < y < ly-1)):
                f[1,x+1,y] = fc[3,x+1,y]
                f[5,x+1,y+1] = fc[7,x+1,y+1]
                f[8,x+1,y-1] = fc[6,x+1,y-1]
            elif ((obst[x,y] == 1) and (y == ly-1)):
                f[8,x+1,y-1] = fc[6,x+1,y-1]
            elif ((obst[x,y] == 2) and (y == 0)):
                f[6,x-1,y+1] = fc[8,x-1,y+1]
```

```
            elif ((obst[x,y] == 2) and (0 < x < lx-1)):
                f[3,x-1,y] = fc[1,x-1,y]
                f[6,x-1,y+1] = fc[8,x-1,y+1]
                f[7,x-1,y-1] = fc[5,x-1,y-1]
            elif ((obst[x,y] == 2) and (y == ly-1)):
                f[7,x-1,y-1] = fc[5,x-1,y-1]

            # Main Channel
            elif ((obst[x,y] == 3) and (x < lx-1)):
                f[2,x,y+1] = fc[4,x,y+1]
                f[5,x+1,y+1] = fc[7,x+1,y+1]
                f[6,x-1,y+1] = fc[8,x-1,y+1]
            elif ((obst[x,y] == 3) and (x == lx-1)):
                f[6,x-1,y+1] = fc[8,x-1,y+1]
            elif ((obst[x,y] == 4) and (x < lx-1)):
                f[4,x,y-1] = fc[2,x,y-1]
                f[7,x-1,y-1] = fc[5,x-1,y-1]
                f[8,x+1,y-1] = fc[6,x+1,y-1]
            elif ((obst[x,y] == 4) and (x == lx-1)):
                f[7,x-1,y-1] = fc[5,x-1,y-1]

            # Corners of T-junction
            elif (obst[x,y] == 5):
                f[2,x,y+1] = fc[4,x,y+1]
                f[3,x-1,y] = fc[1,x-1,y]
                f[5,x+1,y+1] = fc[7,x+1,y+1]
                f[6,x-1,y+1] = fc[8,x-1,y+1]
                f[7,x-1,y-1] = fc[5,x-1,y-1]
            elif (obst[x,y] == 6):
                f[3,x-1,y] = fc[1,x-1,y]
                f[4,x,y-1] = fc[2,x,y-1]
                f[6,x-1,y+1] = fc[8,x-1,y+1]
                f[7,x-1,y-1] = fc[5,x-1,y-1]
                f[8,x+1,y-1] = fc[6,x+1,y-1]

            #Outlet, Lou et al 2011
            elif (obst[x,y] == 0.6):
                f[3,x-1,y]= (f[3,x-1,y]+f[3,x-2,y]*np.mean(ux[x-2,45:55])) \
                /(1+np.mean(ux[x-2,45:55]))
                f[6,x-1,y]= (f[6,x-1,y]+f[6,x-2,y]*np.mean(ux[x-2,45:55])) \
                /(1+np.mean(ux[x-2,45:55]))
                f[7,x-1,y]= (f[7,x-1,y]+f[7,x-2,y]*np.mean(ux[x-2,45:55])) \
                /(1+np.mean(ux[x-2,45:55]))

    #Corners of outlet boundary
    f[6,lx-2,55]=fc[8,lx-2,55]
    f[7,lx-2,44]=fc[5,lx-2,44]

    return f


#%% Function for determining u,v,rho
@nb.njit(parallel=True)
def getuv(obst,ux,uy,rhor,rhob,rho,rhon,fr,fb):
            for i in nb.prange(0,lx):
                for j in nb.prange(0,ly):
                    if (obst[i,j] == 0):
                        rhor[i,j] = fr[0,i,j]+fr[1,i,j]+fr[2,i,j]+ fr[3,i,j]+ \
```

```
                        fr[4,i,j]+fr[5,i,j]+fr[6,i,j]+fr[7,i,j]+fr[8,i,j];
            rhob[i,j] = fb[0,i,j]+fb[1,i,j]+fb[2,i,j]+fb[3,i,j]+ \
                        fb[4,i,j]+fb[5,i,j]+fb[6,i,j] +fb[7,i,j]+fb[8,i,j];
            rho[i,j] = rhor[i,j] + rhob[i,j];

            ux[i,j] = (fb[1,i,j]-fb[3,i,j]+fb[5,i,j]-fb[6,i,j]-fb[7,i,j]+fb[8,i,j]
                       +fr[1,i,j]-fr[3,i,j]+fr[5,i,j]-fr[6,i,j]-fr[7,i,j] + \
                       fr[8,i,j])/rho[i,j];
            uy[i,j] = (fb[2,i,j]-fb[4,i,j]+fb[5,i,j]+fb[6,i,j]-fb[7,i,j]
                       -fb[8,i,j]+fr[2,i,j]-fr[4,i,j]+fr[5,i,j]+fr[6,i,j] - \
                       fr[7,i,j] - fr[8,i,j])/rho[i,j]

        #Phase field function which determines fraction of each fluid, Liu et al 2015
        rhon[i,j]= (rhor[i,j]-rhob[i,j])/(rhor[i,j]+rhob[i,j])

p=rho/3 #pressure

#Inlet
rhon[:,0] = rhon[:,1]
rhon[:,ly-1] = rhon[:,ly-2]

for i in nb.prange(0,lx):
    for j in nb.prange(0,ly):
         if(obst[i,j]>0):
            ie= i%(lx-1)+1
            if (i==lx-1):
                ie = 0
            iw=lx-1-(lx-i)%(lx)
            jn= j%(ly-1)+1
            if (j==ly-1):
                jn = 0
            js= ly-1-(ly-j)%(ly)

            if((obst[i,j] == 1)):
            #Central difference near boundary
                dw1= (rhon[i+1,jn]-rhon[i+1,js])/2
                dw2=(rhon[i+2,jn]-rhon[i+2,js])/2
                rhon[i,j]= rhon[i+1,j]+Tw[i,j]*np.abs(1.5*dw1-0.5*dw2)
            if((obst[i,j] == 2)):
            #Central difference near boundary
                dw1= (rhon[i-1,jn]-rhon[i-1,js])/2
                dw2=(rhon[i-2,jn]-rhon[i-2,js])/2
                rhon[i,j]= rhon[i-1,j]+Tw[i,j]*np.abs(1.5*dw1-0.5*dw2)
            if((obst[i,j] == 3)):
                dw1= (rhon[ie,j+1]-rhon[iw,j+1])/2
                dw2= (rhon[ie,j+2]-rhon[iw,j+2])/2
                rhon[i,j]= rhon[i,j+1]+Tw[i,j]*np.abs(1.5*dw1-0.5*dw2)
            if((obst[i,j] == 4)):
                dw1= (rhon[ie,j-1]-rhon[iw,j-1])/2
                dw2= (rhon[ie,j-2]-rhon[iw,j-2])/2
                rhon[i,j]= rhon[i,j-1]+Tw[i,j]*np.abs(1.5*dw1-0.5*dw2)

            # Junction Corners
            if((obst[i,j] == 5)):
                dw1h= (rhon[i-1,jn]-rhon[i-1,js])/2
                dw2h=(rhon[i-2,jn]-rhon[i-2,js])/2
                rhonh= rhon[i-1,j]+Tw[i,j]*np.abs(1.5*dw1h-0.5*dw2h)
```

```
                        dw1v= (rhon[ie,j+1]-rhon[iw,j+1])/2
                        dw2v= (rhon[ie,j+2]-rhon[iw,j+2])/2
                        rhonv= rhon[i,j+1]+Tw[i,j]*np.abs(1.5*dw1v-0.5*dw2v)

                        rhon[i,j] = (rhonh + rhonv)/2

                    if((obst[i,j] == 6)):
                        dw1h = (rhon[i-1,jn]-rhon[i-1,js])/2
                        dw2h = (rhon[i-2,jn]-rhon[i-2,js])/2
                        rhonh = rhon[i-1,j]+Tw[i,j]*np.abs(1.5*dw1h-0.5*dw2h)

                        dw1v = (rhon[ie,j-1]-rhon[iw,j-1])/2
                        dw2v = (rhon[ie,j-2]-rhon[iw,j-2])/2
                        rhonv = rhon[i,j-1]+Tw[i,j]*np.abs(1.5*dw1v-0.5*dw2v)

                        rhon[i,j] = (rhonh + rhonv)/2

                    #Outlet
                    if(obst[i,j]==0.6):
                      rhon[i,j]=rhon[i-1,j]

            #Inlet Corners
            rhon[0,0] = (rhon[1,0]+rhon[0,1])/2
            rhon[11,0] = (rhon[10,0]+rhon[11,1])/2
            rhon[0,ly-1] = (rhon[1,ly-1]+rhon[0,ly-2])/2
            rhon[11,ly-1] = (rhon[10,ly-1]+rhon[11,ly-2])/2



            rhon[lx-1,44] = (rhon[lx-1,45]+rhon[lx-2,44])/2
            rhon[lx-1,55] = (rhon[lx-1,54]+rhon[lx-2,55])/2

            return rhor,rhob,rho,ux,uy,p,rhon


#%% Moment equation for MRT
@nb.njit(parallel=True)
def meq(obst,f,lx,ly,rho,ux,uy,alpha,M,s):
    fmeq=np.zeros((9,lx,ly))
    fmo=np.zeros((9,lx,ly))
    Mi=np.linalg.inv(M)
    for i in nb.prange(0,lx):
                for j in nb.prange(0,ly):
                   if(obst[i,j] == 0):
                        fmeq[0,i,j]=rho[i,j]
                        fmeq[1,i,j]=rho[i,j]*(-3.6*alpha-0.4+3*(ux[i,j]**2+uy[i,j]**2))
                        fmeq[2,i,j]=rho[i,j]*(5.4*alpha-1.4-3*(ux[i,j]**2+uy[i,j]**2))
                        fmeq[3,i,j]=rho[i,j]*ux[i,j]
                        fmeq[4,i,j]=rho[i,j]*ux[i,j]*(-1.8*alpha-0.2)
                        fmeq[5,i,j]=rho[i,j]*uy[i,j]
                        fmeq[6,i,j]=rho[i,j]*uy[i,j]*(-1.8*alpha-0.2)
                        fmeq[7,i,j]=rho[i,j]*(ux[i,j]**2-uy[i,j]**2)
                        fmeq[8,i,j]=rho[i,j]*ux[i,j]*uy[i,j]
                        for k in nb.prange(0,9):
                            mom=0
                            for l in nb.prange(0,9):
                                mom=mom+M[k,l]*f[l,i,j]
                            fmo[k,i,j]=mom
```

```
    return fmeq,fmo


#%% Collision Function - MRT model, parameters as per Ba et al 2017
@nb.njit(parallel=True)
def collision(obst,ux,uy,rhor,rhob,rho,fr,fb,ff,fer,feb,wk,ex,G,csqu,taur,taub,s,M):
    #Parameters for relaxation parameter
    delt=0.98;
    alp= 2*taur*taub/(taur+taub)
    bet= 2*(taur-alp)/delt;
    kap= -bet/(2*delt);
    eta= 2*(alp-taub)/delt;
    kxi= eta/(2*delt);
    tau=np.zeros((lx,ly))
    #Parameters for calculating pressure
    csqr=3*(1-alphar)/5; csqb= 3*(1-alphab)/5;
    [x,y]= np.where(obst==0)
    Mi=np.linalg.inv(M) # Inverse of moment matrix
    Id=np.identity(9)
    #Calculate equilibrium moments
    [fmer,fmor]= meq(obst,fr,lx,ly,rhor,ux,uy,alphar,M,s)
    [fmeb,fmob]= meq(obst,fb,lx,ly,rhob,ux,uy,alphab,M,s)
    for i in nb.prange(0,lx):
                for j in nb.prange(0,ly):
                   if(0<=obst[i,j]<0.5):
                    jn= j%(ly-1)+1
                    ie= i%(lx-1)+1
                    # if (i==lx-1):
                    #     ie=0
                    js= ly-1-(ly-j)%(ly)
                    iw= lx-1-(lx-i)%(lx)
                    #Relaxation parameter as a function of position
                    phi= (rhor[i,j]-rhob[i,j])/(rhor[i,j]+rhob[i,j])
                    #Relaxation time
                    if (phi>delt):
                       tau[i,j]=taur
                    elif ((phi>0) & (phi<=delt)):
                        tau[i,j]= alp+bet*phi+kap*phi**2
                    elif ((phi<=0) & (phi>=-delt)):
                        tau[i,j]= alp+eta*phi+kxi*phi**2
                    elif (phi<-delt):
                        tau[i,j]=taub
                    #Relaxation Matrix
                    s[0,0]=1;s[1,1]=1.63;s[2,2]=1.54;s[3,3]=1;s[4,4]=1.92;s[5,5]=1;s[6,6]=1.92
                    s[7,7]=1/tau[i,j];s[8,8]=1/tau[i,j]
                    #Remove unwanted terms as per Ba et al, 2017
                    dqxr=1/csqu*(1.8*alphar-0.8)*(wk[1]*ex[1]*rhor[ie,j]*ux[ie,j]+wk[3]*ex[3]* \
                        rhor[iw,j]*ux[iw,j]+wk[5]*ex[5]*rhor[ie,jn]*ux[ie,jn]+wk[6]*ex[6]* \
                        rhor[iw,jn]*ux[iw,jn]+wk[7]*ex[7]*rhor[iw,js]*ux[iw,js]+wk[8]*ex[8]* \
                        rhor[ie,js]*ux[ie,js])
                    dqxb=1/csqu*(1.8*alphab-0.8)*(wk[1]*ex[1]*rhob[ie,j]*ux[ie,j]+wk[3]*ex[3]* \
                        rhob[iw,j]*ux[iw,j]+ wk[5]*ex[5]*rhob[ie,jn]*ux[ie,jn]+wk[6]*ex[6]* \
                        rhob[iw,jn]*ux[iw,jn]+wk[7]*ex[7]*rhob[iw,js]*ux[iw,js]+wk[8]*ex[8]* \
                        rhob[ie,js]*ux[ie,js])
                    dqyr=1/csqu*(1.8*alphar-0.8)*(wk[2]*ey[2]*rhor[i,jn]*uy[i,jn]+wk[4]*ey[4]* \
                        rhor[i,js]*uy[i,js]+ wk[5]*ey[5]*rhor[ie,jn]*uy[ie,jn]+wk[6]*ey[6]* \
                        rhor[iw,jn]*uy[iw,jn]+wk[7]*ey[7]*rhor[iw,js]*uy[iw,js]+wk[8]*ey[8]* \
                        rhor[ie,js]*uy[ie,js])
```

```
                    dqyb=1/csqu*(1.8*alphab-0.8)*(wk[2]*ey[2]*rhob[i,jn]*uy[i,jn]+wk[4]*ey[4]* \
                        rhob[i,js]*uy[i,js]+ wk[5]*ey[5]*rhob[ie,jn]*uy[ie,jn]+wk[6]*ey[6]* \
                        rhob[iw,jn]*uy[iw,jn]+wk[7]*ey[7]*rhob[iw,js]*uy[iw,js]+wk[8]*ey[8]* \
                        rhob[ie,js]*uy[ie,js])
                    Cr=np.zeros((9))
                    Cr[1]=3*(1-s[1,1]/2)*(dqxr+dqyr)
                    Cr[7]=(1-s[7,7]/2)*(dqxr-dqyr)
                    Cb=np.zeros((9))
                    Cb[1]=3*(1-s[1,1]/2)*(dqxb+dqyb)
                    Cb[7]=(1-s[7,7]/2)*(dqxb-dqyb)
                    D=np.dot(Mi,s)
                    #Collision step in moment space
                    for n in nb.prange(0,9):
                        omr=0
                        omb=0
                        for m in nb.prange(0,9):
                            omr=omr+ Mi[n,m]*s[m,m]*(fmor[m,i,j]-fmer[m,i,j])-Mi[n,m]* \
                            (Id[m,m]-s[m,m]/2)*Cr[m]
                            omb=omb + Mi[n,m]*s[m,m]*(fmob[m,i,j]-fmeb[m,i,j])-Mi[n,m]* \
                            (Id[m,m]-s[m,m]/2)*Cb[m]
                        fr[n,i,j]= fr[n,i,j]-omr
                        fb[n,i,j]= fb[n,i,j]-omb
                        #if(i<509):
                        ff[n,i,j]= fr[n,i,j]+fb[n,i,j]
    return fr,fb,ff,tau


#%% Perturbation and Redistribution Function
@nb.njit(parallel=True)
def redistribute(obst,csqu,ux,uy,rhor,rhob,rho,rhon,sigma,fr,fb,ff,ex,ey,lx,ly,wk,rsq,beta,tau,s,M):
    Mi=np.linalg.inv(M)
    [x,y]= np.where((obst==0))
    fc= np.zeros((lx,ly))
    #Unit normal and derivative
    dx= np.zeros((lx,ly))
    dy= np.zeros((lx,ly))
    nx= np.zeros((lx,ly))
    ny= np.zeros((lx,ly))
    dxnx=np.zeros((lx,ly))
    dynx=np.zeros((lx,ly))
    dxny=np.zeros((lx,ly))
    dyny=np.zeros((lx,ly))
    coslam= np.zeros((9,lx,ly)) #angle between color gradient and direction
    un=np.zeros((9,lx,ly))
    Fn=np.zeros((9,lx,ly))
    Fu=np.zeros((lx,ly))
    upx=np.zeros((lx,ly))
    upy=np.zeros((lx,ly))
    Fix=np.zeros((lx,ly))
    Fiy=np.zeros((lx,ly))
    Id=np.identity(9)

    for i in nb.prange(0,lx):
                for j in nb.prange(0,ly):
                    if(obst[i,j] == 0):
                        jn= j%(ly-1)+1
                        ie= i%(lx-1)+1
                        js= ly-1-(ly-j)%(ly)
```

```
                    iw= lx-1-(lx-i)%(lx)

                    # Gradient of phase field function
                    dx[i,j] = 1/csqu*(wk[1]*ex[1]*rhon[ie,j]+wk[3]*ex[3]*rhon[iw,j] \
                            + wk[5]*ex[5]*rhon[ie,jn]+wk[6]*ex[6]*rhon[iw,jn] \
                            + wk[7]*ex[7]*rhon[iw,js]+wk[8]*ex[8]*rhon[ie,js])
                    dy[i,j] = 1/csqu*(wk[2]*ey[2]*rhon[i,jn]+wk[4]*ey[4]*rhon[i,js] \
                            + wk[5]*ey[5]*rhon[ie,jn]+wk[6]*ey[6]*rhon[iw,jn] \
                            + wk[7]*ey[7]*rhon[iw,js]+wk[8]*ey[8]*rhon[ie,js])

                    fc[i,j]= np.sqrt(dx[i,j]**2+dy[i,j]**2)
                    if (fc[i,j]>10**-8):
                        nx[i,j]= -dx[i,j]/fc[i,j]
                        ny[i,j]= -dy[i,j]/fc[i,j]

    for i in nb.prange(0,lx):
        for j in nb.prange(0,ly):
                if(obst[i,j] == 0):
                    jn= j%(ly-1)+1
                    ie= i%(lx-1)+1
                    # if (i==lx-1):
                    #     ie=0
                    js= ly-1-(ly-j)%(ly)
                    iw= lx-1-(lx-i)%(lx)
            #Second derivative for curvature, CSF for interfacial tension
                    dxnx[i,j] = 1/csqu*(wk[1]*ex[1]*nx[ie,j]+wk[3]*ex[3]*nx[iw,j]+ \
                        wk[5]*ex[5]*nx[ie,jn]+wk[6]*ex[6]*nx[iw,jn]+wk[7]*ex[7]*nx[iw,js] \
                        + wk[8]*ex[8]*nx[ie,js])
                    dyny[i,j] = 1/csqu*(wk[2]*ey[2]*ny[i,jn]+wk[4]*ey[4]*ny[i,js]+ \
                        wk[5]*ey[5]*ny[ie,jn]+wk[6]*ey[6]*ny[iw,jn]+wk[7]*ey[7]*ny[iw,js] \
                        + wk[8]*ey[8]*ny[ie,js])
                    dynx[i,j] = 1/csqu*(wk[2]*ey[2]*nx[i,jn]+wk[4]*ey[4]*nx[i,js]+ \
                        wk[5]*ey[5]*nx[ie,jn]+wk[6]*ey[6]*nx[iw,jn]+wk[7]*ey[7]*nx[iw,js] \
                        + wk[8]*ey[8]*nx[ie,js])
                    dxny[i,j] = 1/csqu*(wk[1]*ex[1]*ny[ie,j]+wk[3]*ex[3]*ny[iw,j]+ \
                        wk[5]*ex[5]*ny[ie,jn]+wk[6]*ex[6]*ny[iw,jn]+wk[7]*ex[7]*ny[iw,js] \
                        + wk[8]*ex[8]*ny[ie,js])

                            #Curvature
                    K= nx[i,j]*ny[i,j]*(dynx[i,j]+dxny[i,j])-nx[i,j]**2*dyny[i,j] \
                        - ny[i,j]**2*dxnx[i,j]
                    Fix[i,j]= -0.5*sigma*K*dx[i,j] #Interfacial force, CSF model, Liu et al, 2015
                    Fiy[i,j]= -0.5*sigma*K*dy[i,j]
#Physical velocity modified due to interfacial force, Liu et al 2015
                    upx[i,j]= ux[i,j]+Fix[i,j]/2/rho[i,j]
                    upy[i,j]= uy[i,j]+Fiy[i,j]/2/rho[i,j]
                    Fu[i,j]=Fix[i,j]*upx[i,j]+Fiy[i,j]*upy[i,j]
                    un[0:9,i,j]=ex[0:9]*upx[i,j]+ey[0:9]*upy[i,j]
                    Fn[0:9,i,j]=ex[0:9]*Fix[i,j]+ey[0:9]*Fiy[i,j]
                    F=np.zeros((9))
                    s[0,0]=1;s[1,1]=1.25;s[2,2]=1.14;s[3,3]=1;s[4,4]=1.6;s[5,5]=1;s[6,6]=1.6
                    s[7,7]=1/tau[i,j];s[8,8]=1/tau[i,j]
    #Perturbation operator, Ba et al,2017
                    F= wk[0:9]*(1-0.5/tau[i,j])*((Fn[0:9,i,j]-Fu[i,j])/csqu+ \
                                (Fix[i,j]*ex[0:9]*un[0:9,i,j]+Fiy[i,j]*ey[0:9]*un[0:9,i,j])/csqu**2)
                    P=np.dot(np.dot(Mi,(Id-0.5*s)),M)
                    F=np.dot(P,F) #Force in moment space, Liu et al 2015 eq. 4
```

```
                    #Redistribution
                    for n in nb.prange(0,9):
                        #Cases for denominator=0
                        if (fc[i,j]<10**-8 and fc[i,j]>=0):
                          fr[n,i,j]= ff[n,i,j]*rhor[i,j]/rho[i,j]
                          fb[n,i,j]= ff[n,i,j]*rhob[i,j]/rho[i,j]
                        else:
                            if (n==0):
                              ff[n,i,j]= ff[n,i,j]+F[n]
                              fr[n,i,j]= rhor[i,j]*ff[n,i,j]/rho[i,j];
                              fb[n,i,j]= rhob[i,j]*ff[n,i,j]/rho[i,j];
                            else:
                              coslam[n,i,j]= (ex[n]*dx[i,j]+ ey[n]*dy[i,j])/(rsq[n]*fc[i,j]);
                              ff[n,i,j]= ff[n,i,j]+F[n]
                              tem= rhor[i,j]*rhob[i,j]/(rho[i,j]);
                          #Redistributed f's;
                              fr[n,i,j]= rhor[i,j]*ff[n,i,j]/rho[i,j]+ beta*tem*wk[n]*coslam[n,i,j];
                              fb[n,i,j]= rhob[i,j]*ff[n,i,j]/rho[i,j]-beta*tem*wk[n]*coslam[n,i,j];
    return fr,fb


#%% Output function
@nb.jit#(parallel=True)
def results(lx,ly,obst,rhor,rhob,rho,rhon,ux,uy,p):
     x=np.array([range(0,lx)])
     y=np.array([range(0,ly)])
     print('Iter',t)
     dict={'x':x,'y':y,'rhor':rhor,'rhob':rhob,'rho':rho,'rhon':rhon,'ux':ux,'uy':uy,'p':p,'obst':obst}
     fname= "Itera%s.mat" % t
     sa=sio.savemat(fname,dict)
     return sa


#%% Iterations
fcr=np.zeros((9,lx,ly)) #PDF after redistribution
fcb=np.zeros((9,lx,ly))
rhon=np.zeros((lx,ly))
ux=np.zeros((lx,ly))
uy=np.zeros((lx,ly))
[fer,feb]=feq(rhor,rhob,ux,uy,alphar,alphab,wk,csqu,ex,ey,lx,ly,obst)
#Initial probability distribution of particles
fr= fer; fb=feb;
ff= fr+fb; #Overall prob dist.
fcr=np.copy(fr)
fcb=np.copy(fb)
for t in range(0,tm+1):
            #Streaming
            fr= stream(fr,lx,ly,obst)
            fb= stream(fb,lx,ly,obst)
            ff=fr+fb
            fr= bounceback1(fr,ff,obst,fer,fcr,uib,uir,ex,ey,ux,uy,wk,rhor,csqu,taur)
            fb= bounceback1(fb,ff,obst,feb,fcb,uib,uir,ex,ey,ux,uy,wk,rhob,csqu,taub)
#Determine u,v,rho
            [rhor,rhob,rho,ux,uy,p,rhon]= getuv(obst,ux,uy,rhor,rhob,rho,rhon,fr,fb)
            [fer,feb]=feq(rhor,rhob,ux,uy,alphar,alphab,wk,csqu,ex,ey,lx,ly,obst)
            s=np.zeros((9,9))
            [fr,fb,ff,tau]=collision(obst,ux,uy,rhor,rhob,rho,fr,fb,ff,fer,feb,wk,ex, \
                            G,csqu,taur,taub,s,M)
```

```
            #Redistribution
            [fr,fb]=redistribute(obst,csqu,ux,uy,rhor,rhob,rho,rhon,sigma,fr,fb,ff,ex, \
                           ey,lx,ly,wk,rsq,beta,tau,s,M)
            fcr=np.copy(fr)
            fcb=np.copy(fb)

            # Plot density contour after nw iterations
            if (t%(nw*5)==0):
                plt.contourf(rhor,cmap='RdBu')
                plt.colorbar()
                plt.xlabel(t)
                plt.show()


            #Store data after nw iterations
            if (t%(5*nw)==0):
              results(lx,ly,obst,rhor,rhob,rho,rhon,ux,uy,p)
end= time.time()
print('Running Time:',end-start)

#End
```