INVESTIGATING THE INFLUENCE OF T-SHAPED MICROCHANNEL ARCHITECTURE ON FLOW PATTERNS AND LEAKAGE USING THE LATTICE BOLTZMANN METHOD

Bachelor Thesis

to obtain the degree Bachelor of Science
at Delft University of Technology,
to be defended on June 28th 2022.

by

H.W.M. BAETSEN

Student number: 5175666
Project duration: 19th of April, 2022 - 28th of June, 2022
Thesis committee: Dr. Ir. M. Rohde, TU Delft, supervisor
Dr. Ir. A. G. Denkova TU Delft
A. Sudha

ABSTRACT

Microfluidic liquid-liquid extraction is a fast and safe alternative for transferring radioisotopes from one carrier liquid to another. This thesis investigates the effect of the channel geometry on the flow patterns and leakage observed in a two-phase microfluidic T-channel using n-heptane-water and toluene-water systems. The color-gradient RK multiphase Lattice Boltzmann Method is used in combination with multiphase fluid-fluid and fluid-solid boundary conditions.

As Capillary numbers increase, the flow pattern changes from slug flow to parallel flow due to the increasing dominance of viscous forces. The main parallel flow regime occurs for water Capillary numbers above 2×10^{-4} and n-heptane Capillary numbers above 1.25×10^{-4}. In the transition region between slug and parallel flow, a flow pattern is observed where an eventual parallel flow pushes out several initial slugs. In accordance with previous research, leakage at the outlet of the simulated T-channel always occurs in either direction. For volumetric flow rate ratios $\Phi_{water}/\Phi_{toluene}$ below 0.87, leakage of toluene into the water outlet occurs. Above this value leakage into the toluene outlet takes place. The observed rate is in disagreement with the theoretically and experimentally found value of 0.66 for a Y-channel. Near the transition flow rate ratio, droplets leak into the outlet whereas parallel outlet flow behaviour occurs further away from the boundary.

Generally, parallel flow is observed for a broader range of low inlet velocities compared to a Y-channel, with similar overall leakage performance. Experimental research for confirmation is a required follow-up. The current implementation of the Lattice Boltzmann method was not able to simulate transition flow as observed in experiments. Thus, further improvement of the multiphase Lattice Boltzmann Method is desirable.

CONTENTS

1

INTRODUCTION

The extraction of solvents from one liquid to another is a common challenge in many fields of research and industrial applications. Liquid-liquid extraction using microchannels shows great potential in facilitating efficient and safe extraction. Microfluidic systems have caught interest due to the high surface-to-volume ratio resulting in high mass and heat transfer ratios [1–4]. Small volumes in microchannels allow for higher control of the fluid flow and a more controlled extraction, making them suitable for dealing with dangerous chemicals [5, 6]. In addition, their small scale enables highly parallelizable setups [7, 8]. This is desirable in applications that require low residence times, such as radioisotope production. As a result, liquid-liquid extraction using microchannels has been implemented in a variety of different applications such as industrial scale metal extraction [9], bio-medicine [10] and sample pre-treatment processes [11]. Yet the behaviour inside microchannels remains hard to predict, raising the need for additional research.

In the current work the implementation of microfluidic systems is investigated in the field of medical radioisotope production. As radioisotopes have a finite half-life, microfluidic extraction is investigated as an option to minimize residence time inside the separation stage. The short diffusion distance could allow for shorter extraction times [12]. Additionally, the small-scale 'online' application of microfluidic extractors lowers the radiation exposure to researchers.

1.1. MICROFLUIDIC FLOW

Microfluidics is the domain of science and technology concerned with fluid behaviour at a small scale (< 1mm) [13]. The microfluidic channel is a common instrument within microfluidics. In microfluidic liquid-liquid extraction (LLE), two immiscible fluids enter a microchannel from separate inlets. Often one fluid is aqueous and the other one is organic to ensure immiscibility. In the channel the two fluids propagate together, with mass exchange taking place as a result of differing solubilities of the solute in the two phases. A schematic example of microfluidic LLE can be seen in Figure 1.1.

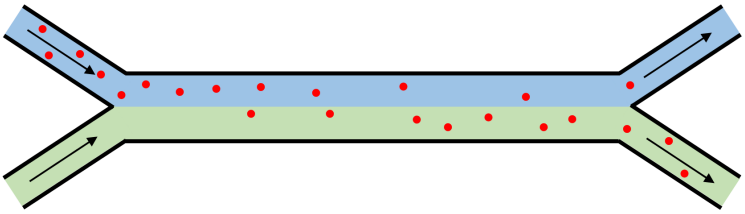


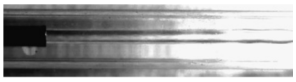
Figure 1.1: Schematic of liquid-liquid extraction between two immiscible fluids in a double-Y microchannel. Solutes are transferred from the blue phase to the green phase.

Figure 1.1 depicts the ideal case where the fluids flow parallel from the inlets to the outlets. However, this is not the only flow pattern observed. The flow pattern plays an important role in the surface-to-volume ratio and therefore in the efficiency of the extraction. Moreover, the flow pattern influences the amount of leakage at the end of the channel, which in turn determines whether an additional separation step is required.

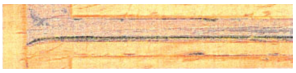
FLOW PATTERNS

The influence of fluid properties and channel architecture on microfluidic flow is a topic that has been of interest the last decades. As channel widths are typically a few hundred micrometers, the dominating forces vary from a traditional, larger-scale fluid system. When considering the flow of multiple fluids at the micro scale, their interaction at the interface plays a more important role while gravity is often negligible [14, 15]. Flow consisting of multiple fluids is called multiphase flow. Generally, multiphase flow types can be classified in 4 main categories: parallel flow phenomena, slug flow phenomena, droplet flow phenomena and transition flow phenomena. In literature, many different names have been used to describe the observed patterns [1, 15–17]. In order to prevent ambiguity, each flow pattern will be briefly introduced.

(a) Parallel flow, adapted from Darekar et al. [1].



(b) Parallel flow, adapted from M.Kashid and L. Kiwi-Minsker [18].



(c) Parallel flow, adapted from Zhao et al. [19].

Figure 1.2: Examples of parallel flow patterns in microchannels.

Parallel flow patterns include all patterns in which the two phases flow side by side. This can happen on either side of the channel (Figure 1.2a), or in an annular way where one phase is enclosed by the other phase. (Figure 1.2b). Slug and droplet flows are flows in which either of the phases is dispersed in the other phase. In slug flow, the dispersed phase forms slugs that occupy (nearly) the entire width of the channel. In droplet flow, the dispersed phase droplets are separated from the wall by a layer of the continuous phase. Examples of different slug and droplet flows can be seen in figures 1.3-1.4.

(a) Slug flow, adapted from Darekar et al. [1].

(b) Slug flow, adapted from M.Kashid and L. Kiwi-Minsker [18].

(c) Slug flow, adapted from Zhao et al. [19].

Figure 1.3: Examples of slug flow patterns in microchannels.

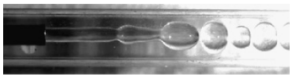
(a) Droplet flow, adapted from Darekar et al. [1].

(b) Droplet flow, adapted from Zhao et al. [19].

Figure 1.4: Examples of droplet flow patterns in microchannels.

Transition flow is less clearly defined and it encompasses flow patterns often referred to as slug-droplet flow (Figure 1.5a-1.5b) and deformed interface flow (Figure 1.5c). In this work, transition flow will be referred to as the phenomena of a starting parallel flow that breaks up into a slug/droplet flow inside the channel at a fixed point as explained by Z. Liu [14].

(a) Slug-droplet flow, adapted from Darekar et al. [1].

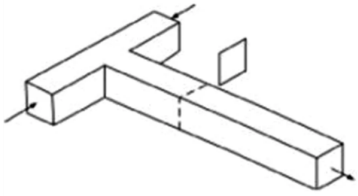


(b) Slug-droplet flow, adapted from M.Kashid and L. Kiwi-Minsker [18].

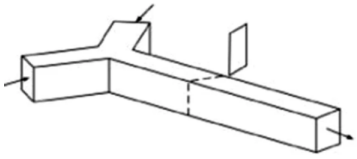
(c) Transition flow, adapted from M.Kashid and L. Kiwi-Minsker [18].

Figure 1.5: Examples of deformed interface flow patterns in microchannels.

The aforementioned flow patterns are dependent on fluid characteristics and channel architecture. Qian et al. give a broad overview of possible channel architectures [15]. In this work T-channels (Figure 1.6a) and Y-channels (Figure 1.6b) are considered, as these are the most researched and offer the most reference results. However, more elaborate channel types such as axisymmetrical channels [20] and cross-shaped microchannels [21] have also been researched.



(a) Schematic of a T-channel, adapted from M.Kashid and L. Kiwi-Minsker [18].



(b) Schematic of a Y-channel, adapted from M.Kashid and L. Kiwi-Minsker [18].

Figure 1.6: Two commonly implemented microchannels, the T-channel and the Y-channel.

The effect of fluid properties has been investigated Darekar et al. using a Y-channel [1]. They experienced slug flow as a result of a dominance in interfacial tension. Droplet flow occurred in regions where the inertial force is large enough to compete with the interfacial force. Parallel flow was observed when the inertial force dominates. Crucially they found that an increase of the dispersed phase flow rate leads to parallel flow.

In their research on the flow of immiscible fluids in a rectangular T-channel, Zhao et al. found similar results with higher Weber numbers (regions where inertial forces dominate over interfacial tension) leading to parallel flow [19]. This recurring pattern is also confirmed by Dessimoz et al. using a Y-channel [16]. Zhao et al. found slug flow at high continuous to dispersed velocity ratios, with the slug size decreasing as the inlet velocity increases. Even though they observed parallel flow at lower Weber numbers than Darekar et al., they did experience unstable flow interfaces for higher Weber numbers [19]. Y-channel research showed mixed results regarding unstable interface forming at high Weber numbers, with Darekar et al. not experiencing this phenomena while Dessimoz et al. did [16].

Besides the shape of the microchannel, the material choice plays a role in the fluid behaviour as shown by Salim et al [22]. It affects the way in which fluids 'stick' to the wall. This phenomenon is called the *wettability* of the channel wall. The more wettable a wall is, the more a fluid attaches to it. Upon investigating the channel architecture and material choice, Darekar et al. concluded that all flow transitions seem to occur at lower flow rates for smaller diameter channels. In addition, if the wall is less wettable for the dispersed fluid, slug flow occurs more easily [1]. This agrees with the research done into wettability by Zhao et al. [23]. As a reference, M. Kashid and L. Kiwi-Minsker [18] compared flow pattern maps from T and Y channels. They found a broader region of deformed interface flow for a Y-channel compared to a T-channel. They created a general flow type prediction table based on these results, in combination with a set of dimensionless numbers that describe the different forces.

To further compare channel architectures, research has been done into specific flow patterns. Ushikubo et al. [24] and Shui et al. [25] focused on the formation of droplets in both T and Y channels, investigating the effect of fluid properties and channel architecture on the droplet size. However, in the current implementation parallel flow is the most interesting domain. Goyal et al. have shown that the extraction of radioactive copper-64 can be done with an extraction efficiency of up to 95% using parallel flow [26]. In the extraction of radioisotopes the minimization of residence time is one of the key interests. It is therefore desired to operate with no leakage at the outlet. When leakage occurs, a certain amount of the organic fluid leaks into the water outlet and/or vice versa. In additional to practical issues such as the need for an additional separation step, there are strict requirements for the purity of radio-pharmaceuticals that emphasize the need for proper phase separation [27]. This outlet leakage has not yet been researched extensively. Most recently, Z. Liu investigated flow patterns and leakage phenomena prevalent in a Y-channel both experimentally and numerically [14]. From these experiments, leakage was found at all inlet velocities when using water and toluene, which is undesirable.

A T-channel could therefore offer a valuable alternative if the amount of leakage is limited.

1.2. PROBLEM DESCRIPTION AND THESIS OUTLINE

In this present work two immiscible fluids in a T-shape microchannel will be investigated using numerical simulation as a cost-effective alternative to experiments. The power of numerical simulation in microfluidics has been proven extensively [28–32]. The resulting flow patterns will be compared to the results obtained by Z. Liu for a Y-channel [14]. In addition, the leakage at the outlet of the T-channel will be simulated for various mass flow ratios and compared to the results obtained by Z. Liu. By doing so, a better decision can be made regarding which channel architecture is more suitable for the extraction of radioisotopes. The fluid properties will be portrayed by means of the Capillary number, which is the ratio between viscous and interfacial forces. To facilitate this comparison, the same liquids will be used as used by Z. Liu. The following research questions have been defined:

1. How do the observed flow patterns in a simulated T-channel with water and n-heptane compare to the flow patterns in a Y-channel as observed by Z. Liu?
   (a) How does the occurrence of flow patterns differ between T and Y-channel research as a function of the fluid Capillary numbers?
   (b) How do the transitions between regimes differ between T and Y-channel simulation?
2. How do the observed flow patterns in a simulated T-channel compare to experimental results for a T-channel based on the evaluation of dominating forces?
3. What leakage is observed at the end of the T-channel as a result of parallel flow using water and toluene compared to the results obtained for a Y-channel by Z. Liu?

In order to answer these questions Chapter 2 discusses the required theory associated with multiphase flow and the Lattice Boltzmann method. Subsequently, the multiphase Lattice Boltzmann model variant used will be discussed in detail in Chapter 3. In Chapter 4 the obtained results will be presented and compared to the results found for a Y-channel by Z. Liu. Finally an overview of conclusions and recommendations will be presented.

2
THEORY

This chapter provides an outline of the physics behind multiphase flow and a dimensionless framework in which fluid properties can be defined. Subsequently the general principle of the Lattice Boltzmann Method is introduced, which provides a base for the multiphase model used to simulate a T-channel.

2.1. FLUID DYNAMICS

In order to understand the behaviour in a microchannel, first fluid dynamics must be discussed. Fluid dynamics is the study of the propagation of fluids, which can be characterised by a set of main equations and dimensionless numbers. When dealing with multiple fluids, the interaction between the two fluids will have to be taken into account in the form of contact angles and interface tension. [33]

2.1.1. GOVERNING EQUATIONS

There are two important sets of equations in fluid dynamics that define the flow of fluids at any scale: the continuity equation and the Navier-Stokes equations. The continuity equation describes the mass flow in and out of a system and ensures mass conservation. For a single fluid, this equation can be written using the fluid density ρ and velocity $\mathbf{u}$ [34]:

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

where $\rho\mathbf{u}$ is referred to as the momentum density. The continuity equation translates as: the sum of the density change over time and the flux of momentum density must be zero. More generally, the right hand side of the equation contains a production (source) term. However, flow where no additional mass is produced is considered. Hence this term equals zero. In the case of incompressible flow, this equation simplifies into:

$$\nabla \cdot \mathbf{u} = 0. \tag{2.2}$$

Together with the conservation of mass, the conservation of momentum is ensured by the Navier-Stokes equations (NSEs). In the case of incompressible liquid flow, the NSEs can be written in terms of the kinematic viscosity ν of the fluid in combination with the material derivative:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \tag{2.3}$$

and the Laplacian operator:

$$\Delta = \nabla \cdot \nabla. \tag{2.4}$$

Using these mathematical definitions the incompressible NSE can be written as [34]:

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla p + \rho \nu \Delta \mathbf{u} + \mathbf{F}. \tag{2.5}$$

Equation 2.5 states that the change in momentum density ($\rho\mathbf{u}$) is due to three factors: a pressure gradient ($\nabla\rho$), the viscosity times the velocity field ($\rho\nu\Delta\mathbf{u}$) and an external force ($\mathbf{F}$). Brennen et al. [33] provide an extensive overview of the application of these principles in multiphase fluid dynamics.

2.1.2. MULTIPHASE FLOW

Generally, fluids are described based on their density ρ and their kinematic viscosity ν. However, when working with multiphase flow, interfacial phenomena also need to be taken into account. These interfacial phenomena can be defined using a set of interfacial tensions σ_i.[1] The interfacial tension between two liquids (or liquid-gas) will be referred to as σ. In addition to this fluid-fluid interaction, wetting at fluid-solid boundaries plays an important role in microfluidic systems [25]. Wetting is a result of fluid-solid interactions manifesting in a surface tension. The extent to which wetting takes place is observed as the adhesion between a fluid and an interface. It can be described using the contact angle θ_C, formed by the interfacial tension between the two fluids (or gas-fluid) σ_{LG} and the surface tension from both liquids σ_{SG} and σ_{SL}. A schematic illustration of the contact angle formed in a droplet at a solid interface can be seen in Figure 2.1.

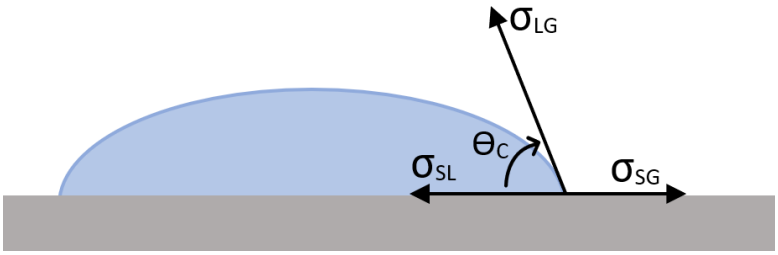


Figure 2.1: The contact angle θ_c is formed by the liquid surface tension σ_{SL}, gas surface tension σ_{SG} and liquid-gas interface tension σ_{LG}.

A smaller contact angle means that the liquid has a bigger affinity for the solid interface and therefor it is more wetting. The contact angle also allows for differentiation between hydrophilic ($\theta_c < 90°$) and hydrophobic ($\theta_c > 90°$) materials [35]. It is given by Young's law [36]:

$$\cos(\theta_c) = \frac{\sigma_{SG} - \sigma_{SL}}{\sigma_{LG}}. \tag{2.6}$$

[1] It is important to differentiate well between interfacial tension (between two liquids) and surface tension (between a liquid and a solid).

The contact angle depends on many factors such as interface roughness [37] and surface tension [38], ranging from macroscopic to molecular scales [39]. In practice, the contact angle is measured macroscopically. When dealing with dynamic conditions (moving liquids), hysteresis of the contact angle takes place. Contact angle hysteresis is the phenomenon of the angle varying based on the fluid velocity and flow direction [35]. The implementation of the contact angle hysteresis is left out as it is tough to implement, because the advancing and receding contact angles are unknown. For the interested reader, Liu et al. discuss a possible implementation of the contact angle hysteresis [40].

2.1.3. DIMENSIONLESS NUMBERS

Based on the physics of fluids and multiphase flow, the system and material properties can be defined in terms of dimensionless numbers. A big reason for this is the so-called law of similarity [34]. The law of similarity states that fluids characterized by the same dimensionless numbers obey the same physics when scaled by typical length and velocity scales. Dimensionless numbers typically relate time scales, length scales and/or the influence of different forces to each other. In this research, 4 dimensionless numbers will be used. They are described in table 2.1, based on fluid properties as well as the typical length scale L. In this work the typical length scale is taken as the channel width.

Table 2.1: An overview of relevant dimensionless numbers and their formulas, along with the relation of forces which they represent. In all the equations the characteristic length scale is denoted by L. Furthermore, the fluid is described by its density ρ, velocity u and kinematic viscosity ν. In the Capillary and Weber number the fluid interfacial tension is characterized by σ. The Bond number is dependent on the gravitational acceleration g and density difference between the two phases $\Delta\rho$.

Reynolds number[2]	Re	$\frac{uL}{\nu}$	Inertial force / Viscous force
Capillary number	Ca	$\frac{\nu\rho u}{\sigma}$	Viscous force / Interfacial tension
Weber number	We	$\frac{\rho L u^2}{\sigma}$	Inertial force / Interfacial tension
Bond number	Bo	$\frac{\Delta\rho g L^2}{\sigma}$	Gravitational force/ Interfacial tension

When considering microfluidic flow the Bond number is typically very small. If the Bond number is smaller than 0.05 the effect of gravity can be neglected [41]. Calculating the Bond number for the fluids and channel architecture used in this research results in 6.25×10^{-4}, hence the effect of gravity can be neglected.

In addition to dimensionless numbers, multiphase flow systems are often described using ratios of volumetric flow rates. The volumetric flow rate Φ is the volume of fluid that moves through a system per unit time. It can be calculated from the fluid velocity u and the surface area A it flows through: $\Phi = uA$.

[2]The Reynolds number is the only dimensionless number of these four that represents a force relation between two intrinsic forces, the other three all concern the influence of multiple phases.

2.2. COMPUTATIONAL FLUID DYNAMICS

Numerical simulations are a particularly powerful method of research in microfluidics. As the NSEs rarely have an analytical outcome, the field of Computational Fluid Dynamics (CFD) offers an alternative in the form of numerical simulation. CFD is cost-effective as it does not require an experimental setup.

There is a vast amount of different CFD methods available. Each method comes with its advantages and disadvantages. The most straight-forward form of CFD for multiphase flow is direct discretization of the Navier Stokes Equation using the Finite Difference method as described by S. Patanakar [42]. However, the simplicity of this method comes at the cost of weak stability and poor conservation of mass [43]. A similar approach using the Finite Volume method solves the conservation issue, but it is difficult to define for complex grid architectures [43]. As an alternative to these direct methods, there is also a range of particle-based solvers that operate on the microscopic scale instead of the macroscopic scale. Several particle-based methods more suitable for multiphase flow simulations are the Phase Field method [44], Volume of fluid methods [45, 46], Level-set methods [47, 48] and the Lattice Boltzmann method (LB). The LB method simulates fluids as groups of particles using a probabilistic particle distribution. By doing so, it is a method at the so-called mesoscopic scale. This scale lies in between microscopic and macroscopic scale [34]. It is the method of choice for this research due to its high parallelizability and ease of boundary condition implementation [49], in addition to its strong connection to macroscopic conservation laws [34].

2.3. THE LATTICE BOLTZMANN METHOD

In order to properly convey the working of the LBM, its application on a single phase will be discussed first. Chapter 3 will cover the used multiphase LBM in detail.

2.3.1. OVERVIEW

The Lattice Boltzmann method is based on the solution of the Boltzmann equation. The Boltzmann equation describes the propagation of gases based on probabilistic particle distributions [34]. These probabilistic particle distributions $f(\mathbf{x}, \xi, t)$ are also known as particle populations. They encompass the behaviour of a particle group based on its position $\mathbf{x}$, lattice velocity ξ and time t. From this particle population the macroscopic moments can be calculated to find density and velocity:

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

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

In addition to providing the macroscopic properties, the evolution over time of the distribution function can be determined. This is done by solving the Boltzmann equation [34]:

$$\frac{\partial f}{\partial t} + \xi_i \frac{\partial f}{\partial x_i} + \frac{F_i}{\rho} \frac{\partial f}{\partial \xi_i} = \Omega(f). \tag{2.9}$$

This equation follows straight forward from the derivative of f to t. With $i = x, y, z$ being the Cartesian coordinates and F_i/ρ as the body force. The last term $\Omega(f)$ is commonly referred to as the collision operator. In fluid mechanics, this collision operator dictates how the particle distributions interact with each other as a function of the fluid viscosity. The original collision operator used by Boltzmann is a cumbersome double integral over velocity space in order to make sure all two-particle collisions are accounted for [34]. However, a much more often used and easy-to-implement collision operator is the Bhatnagar-Gross-Krook (BGK) collision operator [50]:

$$\Omega(f) = -\frac{f - f^{\text{eq}}}{\tau}\Delta t. \tag{2.10}$$

The BGK collision operator can be interpreted as the tendency of the system f to fall back to its equilibrium state f^{eq} over a characteristic *relaxation time* τ, which depends on the viscosity of the fluid. Using a mathematical Chapman-Enskog analysis it can be shown that this simple operator is enough to recover the NSE and continuity equation behaviour from the Boltzmann Equation [34].

Discretizing the Boltzmann equation into distribution functions f_i associated with specific directions i results in the Lattice Boltzmann equation (LBE) [34]:

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) + S_i(\mathbf{x}, t). \tag{2.11}$$

Equation 2.11 is one of the most important equations in the LBM. It describes how the particle distribution propagates in direction i over 1 time step Δt, due to the collision operator Ω_i and a source term S_i.

The simulation grid is defined by a discrete time step Δt and space lattice characterized by a spacing Δx. By choosing a limited fixed set of directions, a discrete velocity set $\{\mathbf{e}_i\}$ can be defined. The most important aspect of this velocity set is that each velocity maps a population exactly to another lattice point over a time step Δt. This way the spatial grid is kept over time. The velocity sets used in LBM are named based on the number of spatial dimensions d and the number of discrete velocities q in the form: DdQq. The most commonly used velocity sets are D2Q9 and D3Q19 for 2D and 3D respectively. Both of these can be seen in Figure 2.2. In order to make sure each velocity component maps to a lattice point they are defined in terms of the lattice speed $c = \Delta x/\Delta t$. For D2Q9 this results in:

$$[e_0\,, e_1\,, e_2\,, e_3\,, e_4\,, e_5\,, e_6\,, e_7\,, e_8] = c\begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}. \tag{2.12}$$

By discretizing the velocity sets in this way, the macroscopic properties from equations 2.7 and 2.8 can be retrieved by a sum over the velocity sets [34]:

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

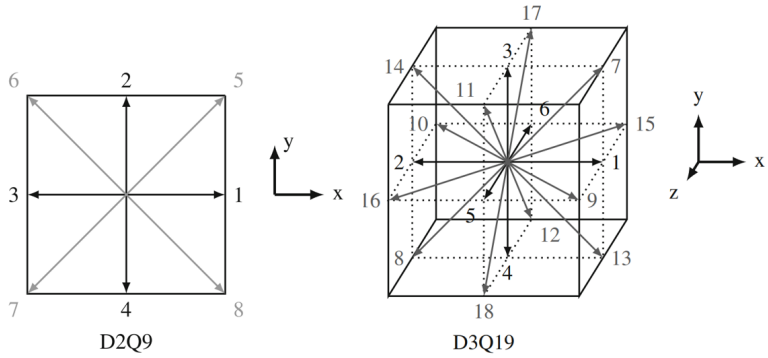


Figure 2.2: Graphical illustrations of the D2Q9 and D3Q19 velocity sets. In both cases the zeroth component e_0 is given by the velocity [0,0]. Adapted from Krüger et al. [34].

2.3.2. EQUILIBRIUM DISTRIBUTION

The lattice speed of sound c_s is related to the lattice speed c: $c_s^2 = c^2/3$. The lattice speed of sound is used to calculate the equilibrium distribution function that is used in the BGK operator (Equation 2.10). The equilibrium function is derived using a so-called Hermite expansion. Fortunately, only the first three terms are required to recover the macroscopic characteristic of fluid dynamics [34]:

$$f_i^{\mathrm{eq}}(\mathbf{x},t) = w_i \rho \left[1 + \frac{\mathbf{e_i}\cdot\mathbf{u}}{c_s^2} + \frac{(\mathbf{e_i}\cdot\mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2} \right]. \tag{2.14}$$

In Equation 2.14 each velocity e_i is assigned a specific weight w_i to account for the magnitude of the velocity. For the D2Q9 velocity set these weights correspond to:

$$\mathbf{w} = \begin{cases} w_0 = \frac{4}{9}, \\ w_{1-4} = \frac{1}{9}, \\ w_{5-8} = \frac{1}{36}. \end{cases} \tag{2.15}$$

Using again the Chapman-Enskog expansion the kinematic viscosity ν can be retrieved as a function of the relaxation time τ [34]:

$$\nu = c_s^2 \left(\tau - \frac{\Delta t}{2} \right). \tag{2.16}$$

2.3.3. STREAMING AND COLLIDING

The most common and intuitive way of implementing the LBM is by separating Equation 2.11 into a streaming and a colliding step. This implementation splits the LBE in a separated local step (colliding) and a parallelizable non-local step (steaming). In the collision step, the collision operator is computed, creating a local post-collision distribution f_i^*. When using the previously introduced BGK collision operator, the discretized collision step reads as:

$$f_i^*(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{\Delta t}{\tau}\left[f_i(\mathbf{x},t) - f_i^{\mathrm{eq}}(\mathbf{x},t)\right]. \tag{2.17}$$

After the post-collision distribution is computed locally, it is simply streamed along its velocity direction i to a neighbouring lattice point:

$$f_i(\mathbf{x}+\mathbf{e_i}\Delta t, t+\Delta t) = f_i^*(\mathbf{x},t). \tag{2.18}$$

After streaming the physical properties of the system are calculated, which are then again used for the calculation of the next equilibrium function in the collision step. This way the process is repeated.

2.3.4. UNIT CONVERSION

An important aspect of the LBM is proper unit conversion. As the LBE is solved in a discrete grid, it requires conversion factors in order to be matched to corresponding physical properties. In other words, conversion factors (C) between *lattice quantities* (depicted with an asterisk *) and *physical properties* are required. For example, when connecting the channel length l [m] to the simulation channel length l^* [lattice units] ([lu]), the the conversion factor C_l is used:

$$l^* = \frac{l}{C_l}. \tag{2.19}$$

As any mechanical quantity can be described in terms of the SI quantities length l, time t and mass m, exactly three independent conversion factors are needed [34]. For a full derivation of the used conversion factors see Appendix A.1. Conventionally, the lattice time step Δx^* and Δt^* are set to 1, such that $C_l = \Delta x$ and $C_t = \Delta t$.

The law of similarity (Chapter 2.1.3) is used to determine the conversion factors. It implies that for identical flow behaviour, the simulation and real-world dimensionless numbers must be exactly the same. For example, the Reynolds number must equate:

$$\mathrm{Re} = \frac{ul}{\nu} = \frac{u^* l^*}{\nu^*}. \tag{2.20}$$

Inserting the definition of the conversion factors from Equation 2.19 into Equation 2.20 yields a relationship between the conversion factors:

$$\frac{C_u C_l}{C_\nu} = 1. \tag{2.21}$$

2.4. MULTIPHASE LATTICE BOLTZMANN METHOD

Multiphase flow introduces contact angle phenomena and surface tension as additional boundary conditions. Separating the two phases is an issue that has been tackled in many different ways. Some examples of multiphase LBMs are the Multiphase Shan-Chen model, the Free energy model and the Color-gradient RK model. All of which have been described thoroughly by Huang et al. [49].

In this research the Color-gradient RK model will be used. By not explicitly tracking the fluid interface, it allows for strict mass conservation for each separate fluid, as well as great flexibility in adjusting interfacial tension. Furthermore, it allows the user to separately adjust density and viscosity ratios, with accurate performance over many different viscosity ratios [51, 52]. When combined with the right collision operator it can even deal with high density ratios [53].

3
MODEL DESCRIPTION

As argued in Section 2.4, the color-gradient Rothman-Keller (RK) model will be used. First introduced by Rothman and Keller in 1988 [54]. The current implementation is based on the implementation using a Multiple Relaxation Time (MRT) Collision operator by Lallemand and Luo [55]. Some additions to this model will be made as proposed by Ba et al. [53] and Liu et al. [40].

3.1. MODEL OUTLINE

The color-gradient model gets its name from the way in which it characterizes different fluids. Each fluid k has a separate distribution function f^k. Typically, the model uses a 'red' and a 'blue' fluid. A total distribution function can be defined from these separate distribution functions:

$$f_i(\mathbf{x}, t) = \sum_k f_i^k(\mathbf{x}, t). \tag{3.1}$$

Using Equation 2.13 the density and velocity can be calculated for the separate phases by summing over f_i^k, or for the total system by summing over f_i. Analogous to the general LBM, the algorithm for the color-gradient model consists of a streaming and a collision step. However, a so-called recoloring step is added in between to separate the two fluids. In the streaming step each phase f^k is streamed separately from its post-recoloring distribution f^{k+}:

$$f_i^k(\mathbf{x} + \mathbf{e_i}\Delta t, t + \Delta t) = f_i^{k+}(\mathbf{x}, t). \tag{3.2}$$

$\mathbf{e_i}$ represent the velocities from the D2Q9 velocity set introduced in Section 2.3.1. The collision step contains an additional second operator $(\Omega^k)^{(2)}$ (perturbation operator) that takes care of interfacial phenomena between the two phases:

$$f_i^{k*}(\mathbf{x}, t) = f_i^k(\mathbf{x}, t) + (\Omega_i^k)^{(1)} + (\Omega_i^k)^{(2)}. \tag{3.3}$$

In Equation 3.3, f_i^{k*} represent post-collision distributions. After the collision step the single-phase distributions f^{k+} are redistributed from the total distribution.

3.1.1. COLLISION OPERATOR

The BGK collision operator introduced in Equation 2.10 introduces unwanted error terms when recovering physical behaviour as shown by Huang et al. [56], especially when simulating multiple phases with different densities. A better alternative is the MRT collision operator as described by Ba et al. [53]:

$$(\Omega_i^k)^{(1)} = \sum_j (\mathbf{M^{-1}S})_{ij}(m_j^k - m_j^{k,\mathrm{eq}}) + \sum_j (\mathbf{M^{-1}})_{ij} C_j^k. \tag{3.4}$$

In order to be able to use this MRT collision operator, the distribution functions f^k are mapped to their moment space counterparts m^k. A linear orthogonal transformation matrix $\mathbf{M}$ and its inverse $\mathbf{M^{-1}}$ are used to do so. Their values for the D2Q9 velocity model can be found in appendix A.2. The moment space distribution functions can be determined as follows:

$$m_i^k = \sum_j \mathbf{M}_{ij} f_j^k, \qquad m_i^{k,\mathrm{eq}} = \sum_j \mathbf{M}_{ij} f_j^{k,\mathrm{eq}}. \tag{3.5}$$

Similar to the standard LBM, the equilibrium distribution function f^{eq} can be derived using a second order Hermite expansion. Alternatively, Ba et al. chose to implement a third order Hermite expansion. They found it cancels out the unwanted error term that can occur in the third order velocity moment, while still fulfilling the first and second order velocity moments given in Equation 2.13 [53]. Upon comparing both equilibrium functions in the current research, a significant increase in computation time is observed while no accuracy is gained. Hence the second order expansion is used:

$$f_i^{k,(\mathrm{eq})}(\mathbf{x}, t) = \rho_k \left(\phi_i^k + w_i \left[1 + \frac{e_i \cdot \mathbf{u}}{c_s^2} + \frac{(e_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2} \right] \right). \tag{3.6}$$

As the D2Q9 is used, the weights w_i as defined in Equation 2.15 remain. In Equation 3.6, ρ_k is the density of the individual phases. The density ratio of the fluids is ensured using the parameter α^k:

$$\frac{\rho_1}{\rho_2} = \frac{1 - \alpha^{(2)}}{1 - \alpha^{(1)}}. \tag{3.7}$$

This density ratio is included in the equilibrium distribution using a direction-dependent factor ϕ_i^k:

$$\phi_i^k = \begin{cases} \alpha^k, & i = 0, \\ \left(1 - \alpha^k\right)/5, & i = 1,2,3,4, \\ \left(1 - \alpha^k\right)/20, & i = 5,6,7,8. \end{cases} \tag{3.8}$$

Furthermore, Equation 3.4 contains a source term C_j^k that ensures the exact recovery of the NSEs. A detailed explanation of the source term is provided by Ba et al. [53]. The remaining matrix $\mathbf{S}$ is a diagonal relaxation matrix that specifies a relaxation rate s_i to each velocity component in f_i. s_0, s_3 and s_5 are associated with conserved moments and have been proven not to influence the derivation of the NSEs [57], therefore they are set to 1. s_1, s_2, s_4 and s_6 are independent and can be adjusted for optimal stability. The values as suggested by Lallemand and Luo [58] are used:

$$\mathbf{S} = \mathrm{diag}[1, 1.63, 1.54, 1, 1.92, 1, 1.92, s_7, s_8]. \tag{3.9}$$

The remaining relaxation parameters s_7 and s_8 are related to the phase relaxation times: $s_7 = s_8 = 1/\tau$. In pursuance of a smooth transition of the relaxation time over the phase

interfaces, an interpolation scheme as proposed by Grunau et al is used.[59]. This is especially important if the difference between relaxation times τ_1 and τ_2 is large. The full interpolation scheme can be found in appendix A.3. Analogous to the single-phase LBM the simulation viscosity can be found from these relaxation times:

$$\nu_k = c_s^2(\tau_k - 0.5)\Delta t. \tag{3.10}$$

3.1.2. PERTURBATION OPERATOR

The perturbation operator $(\Omega^k)^{(2)}$ is introduced to include fluid-fluid interfacial tension into the LBM. Analogous to the collision operator, it is also implemented in the moment-space as proposed by Liu et al [40]:

$$(\Omega^k)^{(2)} = \mathbf{M}^{-1}\left(\mathbf{I} - \frac{1}{2}\mathbf{S}\right)\mathbf{M}\tilde{\mathbf{F}}_s. \tag{3.11}$$

The same relaxation matrix from Equation 3.9 is used in combination with a 9x9 unity matrix $\mathbf{I}$ and the mapping matrices used in the collision operator. Ba et al. demonstrate how the elements of the interfacial force matrix $\tilde{\mathbf{F}}_s$ can be derived from the the interfacial body force $\mathbf{F}_s$ and the extrapolated phase relaxation time τ^k [53]:

$$\tilde{\mathbf{F}}_S = A^k w_i \left(1 - \frac{\tau^k}{2}\right)\left[\frac{(\mathbf{e}_i - \mathbf{u})}{c_s^2} + \frac{(\mathbf{e}_i \cdot \mathbf{u})\,\mathbf{e}_i}{c_s^4}\right]\cdot \mathbf{F}_s, \tag{3.12}$$

where A^k is the fraction of the interfacial tension due to the fluid k, which must satisfy $\sum_k A^k = 1$. The interfacial body force $\mathbf{F}_s$ depends on the interface tension coefficient σ, the local curvature of the interface κ and the gradient of the phase field $\nabla\rho^N$:

$$\mathbf{F}_s = -\frac{1}{2}\sigma\kappa\nabla\rho^N. \tag{3.13}$$

The phase-field ρ^N is a crucial parameter in the color-gradient model. As the color-gradient RK method does not track the fluid interface explicitly, the phase field is used to locate fluid interfaces. As a result, the interfacial force is only implemented in the region where fluid mixing takes place. The phase field is merely a function of the fluid densities:

$$\rho^N = \frac{\rho_1 - \rho_2}{\rho_1 + \rho_2}. \tag{3.14}$$

As a result, the normal vector $\mathbf{n}$ at the fluid interface can be expressed in terms of the phase-field:

$$\mathbf{n} = \frac{-\nabla\rho^N}{|\nabla\rho^N|}. \tag{3.15}$$

This allows the interface curvature κ to be numerically implementable with a finite difference scheme which approximates the actual curvature:

$$\kappa = n_x n_y\left(\frac{\partial}{\partial y}n_x + \frac{\partial}{\partial x}n_y\right) - n_x^2\frac{\partial}{\partial y}n_y - n_y^2\frac{\partial}{\partial x}n_x. \tag{3.16}$$

3.1.3. REDISTRIBUTION

As the LBM does not track the fluid-fluid interface, it can not guarantee immiscibility. Instead of a tracked interface there exists a diffuse interface with a finite thickness. This diffuse interface is used in the redistribution step to separate the fluids. In order to do so, the two phases must be redistributed from the total post-collision and -perturbation population f_i^* to the separate distributions f^{1+} and f^{2+}. This step is also known as the recoloring step:

$$f_i^{1+} = \frac{\rho_1}{\rho} f_i^* + \beta \frac{\rho_1 \rho_2}{\rho} w_i \cos(\lambda_i) |e_i|, \tag{3.17}$$

$$f_i^{2+} = \frac{\rho_2}{\rho} f_i^* - \beta \frac{\rho_1 \rho_2}{\rho} w_i \cos(\lambda_i) |e_i|. \tag{3.18}$$

In equations 3.17-3.18, β is a numerical constant that dictates the diffuse interface thickness. The recoloring step is governed by the the angle λ_i between the phase field gradient and the lattice vectors $\mathbf{e}_i$:

$$\cos(\lambda_i) = \frac{\mathbf{e}_i \cdot \nabla \rho^N}{|\mathbf{e}_i| |\nabla \rho^N|}. \tag{3.19}$$

3.2. BOUNDARY CONDITIONS

The above sections provide the framework in order to retrieve information that obeys the NSEs. However, the NSEs are governed largely by boundary conditions (BCs). Due to their importance, all boundary conditions in the simulation will be discussed. In a T-channel this concerns inlet boundary conditions, outlet boundary conditions and walls. At the walls the contact angle as discussed in Chapter 2.1.2 must also be enforced.

3.2.1. WALLS

At all the walls of the T-channel a no-slip boundary condition is enforced. The no-slip boundary condition ensures that the particles close to the wall have the same velocity as the wall itself, i.e. they do not slip against the wall. This condition is implemented using one of the oldest and most commonly used LBM boundary condition methods: a bounce-back scheme. In short, bounce-back enforces particle populations that hit the wall to reflect to the fluid node they came from. This simple method ensures strict mass conservation [34]. Half way bounce-back is the method of choice, as opposed to full way bounce-back, due to its ease of implementation and the fact that it only takes 1 time step instead of 2. Both methods have been described in detail by Krüger et al. [34].

As can be seen in Figure 3.1, a particle streaming from a fluid node x_N into a solid wall node x_{N+1} encounters the wall in between at time $\Delta t/2$. It is reflected back to the fluid node x_N where it arrives at time Δt. This way, the entire reflection process takes place in 1 time step. As the corners have fluid coming in from more directions, they have an extended bounce-back implementation.

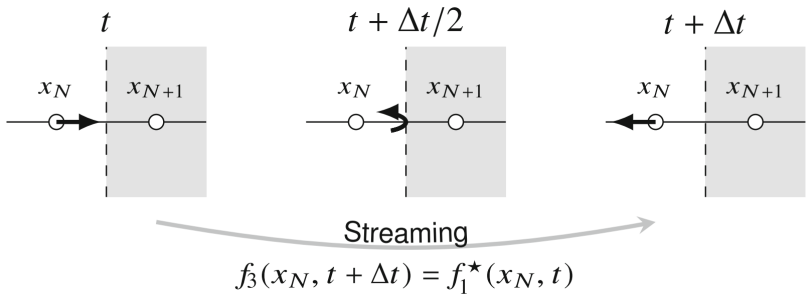


Figure 3.1: Graphical illustration of the half way bounce-back scheme where a population is reflected to the fluid node it came from. Adapted from Krüger et al. [34].

CONTACT ANGLE

As the phenomena of wetting occurs at the walls, it is expected to influence the wall boundary conditions. With the contact angle θ_c defined in Figure 2.1 the geometrical formulation proposed by Ding and Spelt [60] is used to implement wetting in LBM. In this formulation Θ_w is defined as:

$$\Theta_w = \tan\left(\frac{\pi}{2} - \theta_c\right). \tag{3.20}$$

In combination with the unit normal wall vector $\mathbf{n}_w$, the unit tangent wall vector $\mathbf{t}_w$ and the phase field defined in Equation 3.14, the following expression is found:

$$\mathbf{n}_w \cdot \nabla\rho^N = -\Theta_w \left|\mathbf{t}_w \cdot \nabla\rho^N\right|. \tag{3.21}$$

The implementation of Equation 3.21 makes use of the fact that the walls are situated in between nodes as explained in half way bounce-back. As an example, a bottom wall with a solid node at $y = 0$ and a fluid node at $y = 1$ can be discretized using the central difference method:

$$\rho^N_{x,1} = \rho^N_{x,1} + \Theta_w \left|\mathbf{t}_w \cdot \nabla\rho^N\right| \Delta x. \tag{3.22}$$

An extrapolation scheme proposed by Huang et al. [61] is used to determine the tangential component of the gradient of the phase function at the wall:

$$\mathbf{t}_w \cdot \rho^N = 1.5\partial_x \rho^N|_{x,1} - 0.5\partial_x \rho^N|_{x,2}, \tag{3.23}$$

where the partial derivatives are approximated using a central difference method. From Equation 3.22 follows that this boundary condition occurs as an implicit boundary condition for the phase field near the wall.

3.2.2. INLETS

The inlets of the physical channel are assumed to be sufficiently long to ensure a steady-state developed flow profile. This allows the use of a shorter inlet with an enforced inlet fluid velocity $\mathbf{u}_{in}$. Its implementation is similar to bounce-back at a moving-wall, which 'pushes' the fluid into the channel at a constant velocity. A. Ladd proposes the following [62]:

$$f_i^k(\mathbf{x}_b, t + \Delta t) - f_i^{k*}(\mathbf{x}_b, t) - 2w_i \rho_w^k \frac{\mathbf{c}_i \cdot \mathbf{u}_{in}}{c_s^2}, \tag{3.24}$$

where a subscript w refers to a property near the wall. In the simulation, the inlet velocities $\mathbf{u}_{in}$ are determined from the desired Capillary numbers for both phases.

3.2.3. OUTLETS

As the full outlet length is not of interest, the outlet boundary condition must be defined to fully bound the simulation regime. In order for the bulk physics not to be affected by computational errors occurring due to the choice of boundary condition, the Convective Boundary Condition (CBC) is implemented. As shown by Lou et al. this method outperforms other commonly used methods as the Neumann boundary condition and the extrapolation boundary condition [63]. A wall at $x = N$ obeys the CBC as follows:

$$\frac{\partial \chi}{\partial t} + U\frac{\partial \chi}{\partial x} = 0, \tag{3.25}$$

where χ is the variable constrained at the outlet. As suggested by Lou et al. the typical velocity normal to the outlet U is approximated with the average velocity over the outlet width:

$$U(t) = U_{\text{avg}}(t) = \frac{1}{M+1}\sum_{y} u(N-1, y, t). \tag{3.26}$$

In Equation 3.26 the width of the outlet is $M+1$ and the velocity u is taken at $x = N-1$ at a time t for all values of y that correspond to the outlet width. The constrained variable χ is the distribution function f^k. This way the CBC is discretized using a first order derivation approximation to find the mesoscopic implementation for the particle distributions f^k:

$$f_i^k(N, j, t+\Delta t) = \frac{f_i^k(N, j, t) + \lambda f_i^k(N-1, j, t+\Delta t)}{1+\lambda}, \tag{3.27}$$

where λ is given by:

$$\lambda = U(t+\Delta t)\frac{\Delta t}{\Delta x}. \tag{3.28}$$

3.3. STABILITY REQUIREMENTS

Due to the vast amount of parameters, there are a number of stability requirements that need to be taken into account. Huang et al. found that the interface thickness parameter β should be not too large, else the simulation will get unstable [49]. As per convention β is set to = 0.7 [40, 52]. Krüger et al. describe how typical velocities are limited by the lattice speed of sound [34]. In practice, this means that inlet velocities should be kept below 0.1 or even 0.03. As this is the case for single phase LBM, the real stability limit for multiphase LBM may lie even lower than 0.03. Furthermore, Krüger et al. find that the relaxation time of the individual phases should not be much larger than 1.

3.4. MODEL PARAMETERS

The physical channel used by Z. Liu [14] has a width d of 100 μm and a length of 2 cm. However, for simulation purposes a channel length l of 2 mm was used. This choice was motivated by research that showed that the channel length influences the flow pattern when it exceeds 4m [64]. In the simulation the width of the channel is set to $l^* = 10$ lu, corresponding with a conversion factor C_l of 10^{-5} m/lu. From this follows the simulation channel length $l^* = 200$ lu. Both the inlet and outlet lengths have been set to 45 lu, corresponding to 450μm. This choice is made based on the assumption made regarding the developed inlet flow boundary conditions (Section 3.24). In order to save computation times, the channel used for the simulation of flow patterns has a single outlet where the convective boundary condition is applied. Figure 3.2 shows the single-T channel which corresponds to the simulation domain being limited to the inlet and first part of the channel. For the leakage simulation, a double-T channel with a main channel length of 200 lu is implemented.

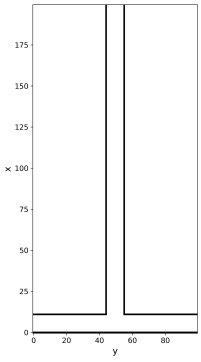


Figure 3.2: Outline of the single T-channel used for flow pattern investigation.

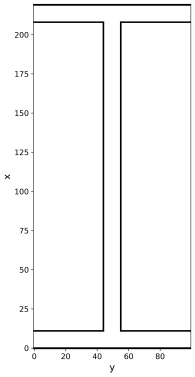


Figure 3.3: Outline of the double T-channel used for leakage investigation.

As the inlets for both phases are identical in size, the volumetric flow rate ratio introduced in Section 2.1.3 for the aqueous (acq) and organic (org) phase simplifies to:

$$\frac{\Phi_{org}}{\Phi_{acq}} = \frac{u_{org}}{u_{acq}}, \tag{3.29}$$

where u_{acq} and u_{org} are the inlet velocities applied to the boundary condition given in Equation 3.24.

FLUID PROPERTIES

The fluids used in this work are water, n-heptane and toluene. For the flow pattern investigation water and n-heptane are used, while leakage is simulated using water and toluene. These combinations match the combinations used by Z. Liu [14]. The *physical properties* of all three liquids can be found in table 3.1.

Table 3.1: An overview of the physical properties of the liquids used in the simulations. The contact angles have been experimentally determined by Z. Liu [14].

Fluids	**Kinematic Viscosity** ν [m^2/s]	**Density** ρ [g/cm^3]	**Interfacial tension** σ [mN/m]	**Contact angle** [°]
water	1×10^{-6}	1	-	-
n-heptane	5.68×10^{-7}	0.68	50.2	47
toluene	6.74×10^{-7}	0.87	36.1	49

From these physical properties, a set of simulation parameters can be defined using the conversion derivation in appendix A.1. This conversion leads to the simulation densities ρ^*, density parameters α, relaxation times τ^*, viscosities ν^* and interfacial tensions σ^*. For ease of implementation, the density of n-heptane is approximated as 666.67 kg/m^3. The *simulation parameters* for water and n-heptane can be found in table 3.2.

Table 3.2: An overview of the fluid simulation parameters for water and n-heptane. The values of alpha have been chosen to recover the correct density ratio while ensuring numerical stability.

Fluids	ρ^*	α	τ^*	ν^*	σ^*
water	1.5	5/9	0.5244	0.00816	0.05
n-heptane	1	3/9	0.5139	0.00463	0.05

Similarly table 3.3 contains the simulation parameters for the water-toluene combination.

Table 3.3: An overview of the fluid simulation parameters for water and toluene. The values of alpha have been chosen to recover the correct density ratio while ensuring numerical stability.

Fluids	ρ^*	α	τ^*	ν^*	σ^*
water	1.1494	0.565	0.5280	0.00933	0.0361
toluene	1	0.5	0.5188	0.00628	0.0361

Using the architecture simulation parameters and the fluid simulation parameters, the length conversion factor C_l and the velocity conversion factor C_u can be found. These lead to the physical length of the time steps: $\Delta t_{heptane} = 8.13\times10^{-7}$ s and $\Delta t_{toluene} = 9.33\times10^{-7}$ s.

SIMULATION INITIALIZATION

In both the single T and double T channels the initialization of the system ensures that the complete channel is filled with water, except for the right inlet. The right inlet is initialized with the organic phase. At time step t = 0 this results in the initial situation as displayed in Figure 3.4

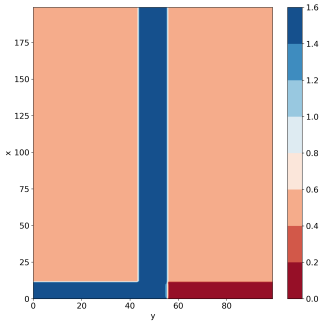


Figure 3.4: Initialization state of a single T-channel. The blue fluid corresponds to water and the red fluid corresponds to n-heptane.

3.5. PYTHON ALGORITHM

The model has been implemented in Python 3.7, making use of the following libraries: `Numba`, `Numpy`, `Matplotlib.pyplot`, `time`, `scipy.io`, `math` and `os`. Figure 3.5 provides a flowchart of the algorithm, saving and plotting of data is done every 5000 time steps. The full python3 algorithm can be found in appendix C.

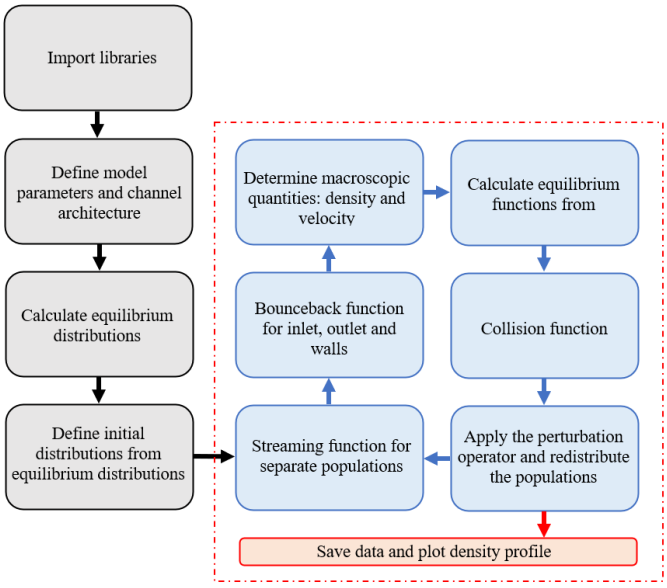


Figure 3.5: Flowchart of the python algorithm.

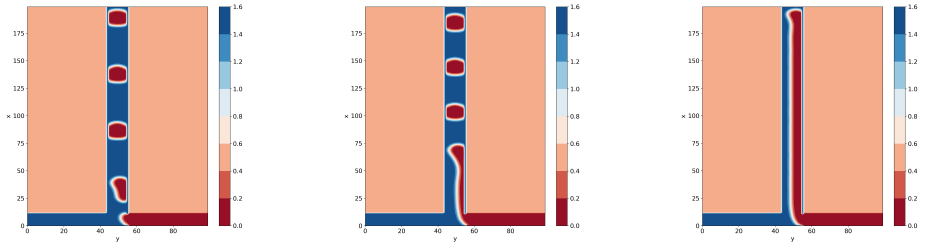
4

RESULTS AND DISCUSSION

Using the model described in Chapter 3, in combination with the model parameters from Section 3.4, simulations have been performed using n-heptane and toluene. The color-gradient model has been validated on two-phase Poiseuille flow by F. De Groot [65]. First, the n-heptane flow patterns for a range of Capillary numbers have been simulated in a single T-channel. The observed flow patterns, along with the corresponding flow map, will be presented and discussed in Section 4.1. In Section 4.2, leakage at the outlet of the T-channel using water and toluene will be analysed and compared to the results obtained by Z. Liu.

4.1. N-HEPTANE FLOW PATTERNS

Simulations of water and n-heptane have been done over a range of water Capillary numbers Ca_w from 10^{-4} to 3×10^{-3} and a range of n-heptane Capillary numbers Ca_h from 4×10^{-5} to 2×10^{-3}. The upper limit of this range ensures inlet velocities that do not exceed the stability requirement given in Section 3.3. Within this range three types of flow patterns have been observed: parallel flow, slug flow and a third pattern which will be referred to as parallel breakup flow. This third pattern will be discussed extensively in Section 4.1.2. Figure 4.1 gives an overview of the observed flow patterns.



(a) Slug flow observed for $Ca_h = 7 \times 10^{-5}$ and $Ca_w = 5 \times 10^{-4}$ at time step 100000.

(b) Parallel breakup flow observed for $Ca_h = 6 \times 10^{-5}$ and $Ca_w = 3 \times 10^{-4}$ at time step 160000.

(c) Parallel flow observed for $Ca_h = 2 \times 10^{-4}$ and $Ca_w = 3 \times 10^{-4}$ at time step 200000.

Figure 4.1: Three flow patterns observed in simulation of a T-channel using water (blue) and n-heptane (red).

Generally speaking, the observed flow pattern moves from slug flow (Figure 4.1a) to parallel flow (Figure 4.1c) as the Capillary number of both phases increases. The parallel

breakup flow pattern observed in Figure 4.1b, occurs in transition regions between slug flow and parallel flow, in addition to high Capillary number regimes. The observed flow patterns as a function of the Capillary numbers of the two phases have been plotted in Figure 4.2.

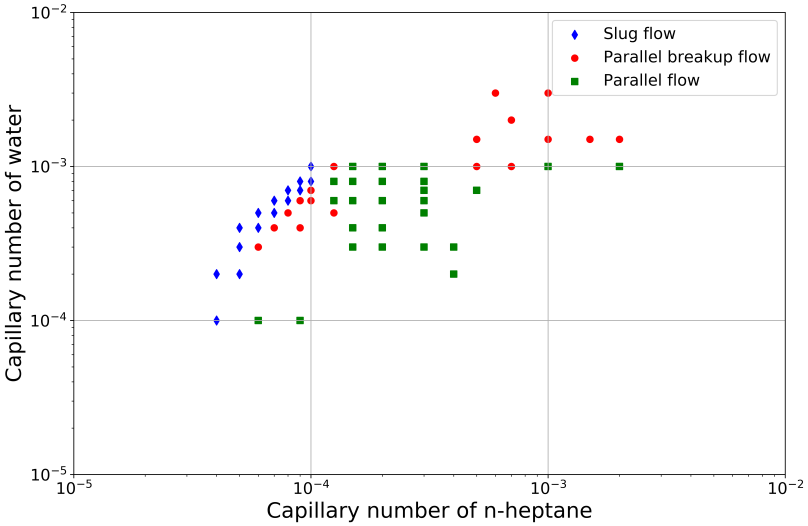


Figure 4.2: Flow map for water and n-heptane generated using a simulation of a T channel.

Figure 4.2 shows how a broad range of Capillary numbers result in parallel flow. This flow pattern occurs for water Capillary numbers from 10^{-4} to 10^{-3} and n-heptane Capillary numbers from 6×10^{-5} to 2×10^{-3}. In contrast, Dessimoz et al. find the slug flow pattern to dominate the flow map of a T-channel [16]. However, their research was done based on the combination of water and toluene. Z. Liu performed Y-channel research into both the water-toluene and water-n-heptane system and found that parallel flow occurs generally for lower Capillary numbers in the case of water-n-heptane compared to water-toluene [14]. So this does not necessarily contradict the observed results.

In order to investigate the effect of junction geometry, the obtained flow map can be compared to the flow map obtained by Z. Liu for a Y-channel. This flow map has been generated based on experimental results and can be seen in Figure 4.3. It is clear that the general trend of both flow maps are similar. Analogous to the T-junction simulation, the flow patterns in a Y-channel move from slug flow to parallel flow as the Capillary numbers increase. Interestingly, the results for a T-channel indicate an earlier transition towards parallel flow. Flow inside the T-channel is parallel for water Capillary numbers as low as 10^{-4} and n-heptane Capillary numbers as low as 6×10^{-5}. In the Y-channel research by Liu parallel flow occurred only for water Capillary numbers of 6×10^{-4} and n-heptane Capillary numbers of 4×10^{-4}.

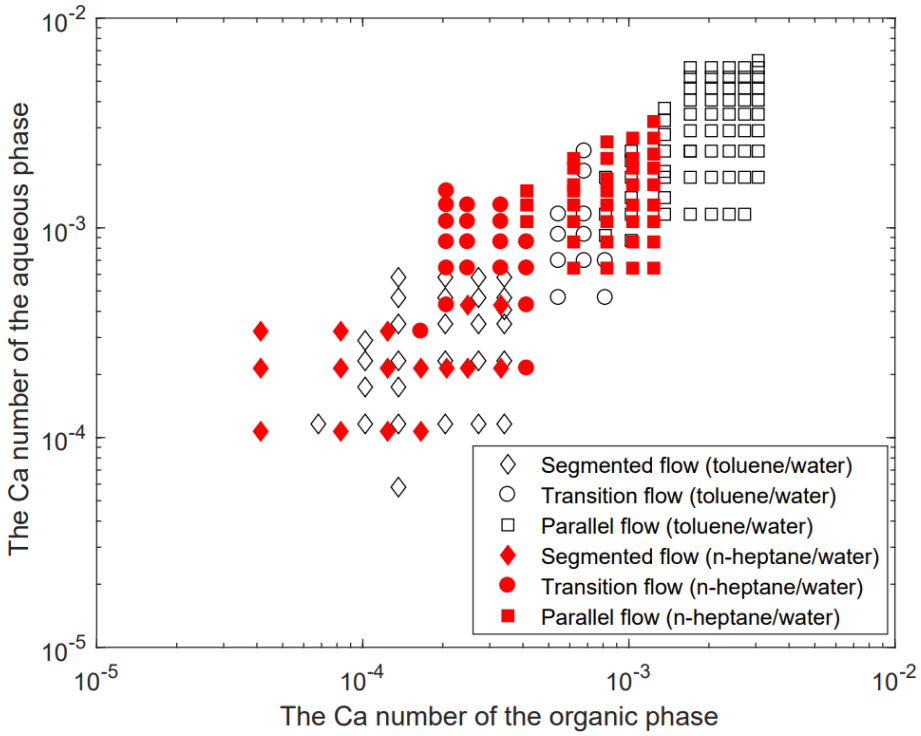


Figure 4.3: Flow map obtained by Z. Liu in a Y-channel experiment. Both the results for the water-n-heptane and water-toluene have been plotted. Adapted from Z. Liu [14].

This result is seemingly unexpected, as Dessimoz et al. found that the flow map for a T-channel is more dominated by slug flow compared to its Y-channel counterpart. However, the early transition into parallel flow, especially for lower water inlet velocities was also found by Salim et al. using a glass T-channel [17]. The parallel flow generated at low water Capillary numbers will be discussed later on, as it differs from the parallel flow observed at higher Capillary numbers. A possible numerical reason for the difference with the Y-channel is the missing of contact angle hysteresis in the model, causing inadequate recreation of slugs for low velocities. The contact angle increases as the fluid moves [40]. At low velocities this impact may be large, as the fluid rests longer against the wall at any point.

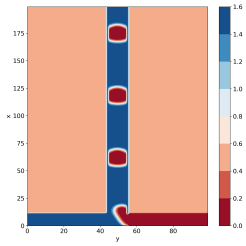
In the high Capillary regions Z. Liu observed parallel flow whereas the simulation indicates parallel breakup flows. These simulations consist of the highest tested inlet velocities: 0.012 for water and 0.02 for heptane at the outer data points. Even though the theoretical stability limit of the single LBM is around 0.03 as described in Section 3.3, the practical velocity limit for multiphase flow simulations might be lower. A. Sudha observed numerical instabilities at similar inlet velocities for a Y-channel. This apparent

lowering in stability may have many reasons such as the large density difference, interfacial effects or the contact angle implementation. However, further investigation into this lies outside the scope of this work.

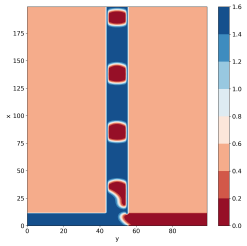
Besides the appearance of parallel breakup flow at high Capillary numbers, another interesting phenomena occurs in parallel flow observed at low water Capillary numbers. In order to investigate the phenomena causing these effects, each flow type will be elaborated on further in detail.

4.1.1. SLUG FLOW

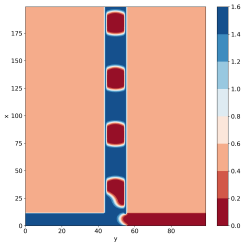
Slug flow is observed when the average Capillary number of the two phases is low (below 5.5×10^{-4}) and the Capillary numbers lie beneath 10^{-3} for water and 10^{-4} for n-heptane. Due to the conservation of mass, the length of the slugs should increase with an increase in the flow ratio. Figure 4.4 indeed shows that the length of the n-heptane slugs increases with the flow ratio $u_{heptane}/u_{water}$. This is also in accordance with research done by Ushikubo et al., where they quantitatively show how the slug/droplet volume of the dispersed phase increases as the flow ratio of the dispersed to continuous phase increases. [24].



(a) Short slug length observed for a flow ratio of $u_{heptane}/u_{water} = 0.33$ at time step 100000.



(b) Medium slug length observed for a flow ratio of $u_{heptane}/u_{water} = 0.44$ at time step 160000.



(c) Long slug length observed for a flow ratio of $u_{heptane}/u_{water} = 0.66$ at time step 160000.

Figure 4.4: Slug flow observed for a n-heptane Capillary number of 5×10^{-5}.

The formation of slugs at the junction of the channel takes place as a 'pinching' motion where the interface of the organic phase starts convex and is pushed to a concave state until the interface reaches the junction corner and it breaks. This phenomenon has also been found in experiments and numerical simulations by Li et al. [66]. An example of this pinching process can be found in appendix Figure B.1. As slug flow occurs for low Capillary numbers, the interfacial force dominates over the viscous force. As a result, the tendency towards interface generation is strong, causing the pinching motion to occur at the inlet. This matches and substantiates the findings by Dessimoz et al. that an increase in interfacial tension leads to slug flow [16].

4.1.2. PARALLEL BREAKUP FLOW

The aforementioned pinching behaviour is also observed inside of the channel. This leads to the formation of parallel breakup flow. The pinching behaviour inside the channel is illustrated in appendix Figure B.2. This behaviour has also been observed by Guillot and Colin, who found that the slug generation inside the main channel cannot be described by the competition between viscous and interfacial forces [67]. Rather, the flow enters the channel as parallel flow. Inside the channel, the tip of the parallel flow moves inwards. This inward movement clogs the channel at some point, causing the blocked water flow to pinch the n-heptane stream until it detaches into a slug.

However, in contrast to experiments [14, 18], the parallel breakup flow results in parallel flow when simulated over extended periods of time. The slugs/droplets formed inside the channel get pushed out over time. *Real* transition flow phenomena are steady-state patterns of an initially parallel flow that breaks up at a (somewhat) fixed point in the channel. Even though prevalent in experiments, this flow pattern has not been observed. Currently, cases have been found from 1-5 preceding slugs/droplets. An example of parallel breakup flow can be seen in Figure 4.1b.

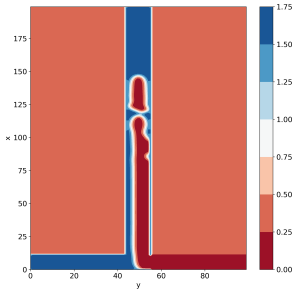


Figure 4.5: Irregularly shaped droplets in parallel breakup flow for $Ca_w = 1.5 \times 10^{-3}$ and $Ca_h = 2 \times 10^{-3}$ at time step 5000.

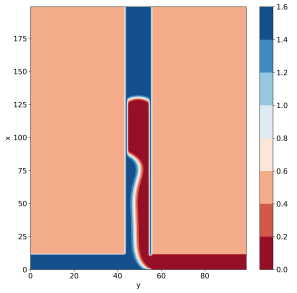


Figure 4.6: Big slug attachment in parallel flow for $Ca_w = 1 \times 10^{-4}$ and $Ca_h = 9 \times 10^{-5}$ at time step 100000.

Parallel breakup flow observed for high water and n-heptane Capillary numbers differ from the parallel breakup flow patterns observed in the transition region between slug and parallel flow. Parallel breakup flow observed in the transition region enclosed by $3 \times 10^{-4} < Ca_w < 10^{-3}$ and $6 \times 10^{-5} < Ca_h < 1.25 \times 10^{-4}$ produces slugs or droplets that are similar in shape to slug flow and experimental research as done by for example Zhao et al. [19]. Parallel breakup flow at high Capillary numbers, such as Figure 4.5, lie past the parallel flow regime and produce irregularly shaped droplets/slugs. As discussed before, this might be an effect of a reduced practical stability limit. It seems likely that experimental research in this region would lead to parallel flow.

4.1.3. PARALLEL FLOW

The initial slug attachment that leads to parallel breakup flow does not always clog the channel and detach. In fact, all parallel flow observed in this research started of with a slug attachment. In the extreme case of very low n-heptane/water flow ratios, the parallel flow is preceded by a large slug followed by a narrow section. This slug does not detach from the parallel flow, but stays attached during the entire propagation. Figure 4.6 shows an example of such slug attachment. The bulk of the observed parallel flow lies in the region spanned by $2 \times 10^{-4} < Ca_w < 10^{-3}$ and $1.25 \times 10^{-4} < Ca_h < 5 \times 10^{-4}$. Comparing Figure 4.1c to 4.6 shows that the attachment in the main region of parallel flow does not occupy a significant length in the simulation domain, whereas the slug attachment for the extreme low velocity ratio flows does. Even though this phenomenon has not been observed in actual experiments, it has been observed by A. Sudha in a LBM simulation of a Y-channel. Seemingly, the slug attachment is a numerical artefact from the LBM. From the bachelor thesis of F. De Groot follows that the slug attachment is an aspect of the LBM which results from the method needing to establish an interface [65]. The large slug attachment may possibly be caused by the inability of the aqueous phase to detach the slug attachment due to the low water inlet velocity. It does not play a crucial role as the initial slug attachment is pushed out over time.

4.2. TOLUENE LEAKAGE

In addition to the simulation of flow patterns using water and n-heptane, the behaviour of water and toluene has been simulated at the outlet of the microchannel. Leakage has been simulated over a range of toluene Capillary numbers in between 6×10^{-4} and 2×10^{-3}, with varying volumetric flow rate ratios. This range of Capillary numbers is lower than the range used by Z. Liu, based on the observation that the T-channel facilitates parallel flow for lower Capillary numbers than the Y-channel. It is important to keep in mind that transition flow was not observed in the previous simulations, so some of the lower Capillary numbers possibly result in transition flow in practice. The direction of the leakage can be found in Figure 4.7.

All simulations showed leakage into either direction. No cases of leakage in both directions or no leakage at all have been found. This is in accordance with the experimental results from Z. Liu [14]. When simulating a Y-channel using the Phase Field method, Z. Liu did however find cases of no leakage. This might be an effect of the Phase Field method tracking the interface of the fluid explicitly, which the LBM does not. Nonetheless, A. Sudha simulated leakage in a Y-channel using the LBM for water and toluene. This simulation did result in some cases of leakage into both outlets. So far it remains undecided whether the LBM offers a significant advantage of the phase field method in leakage simulation.

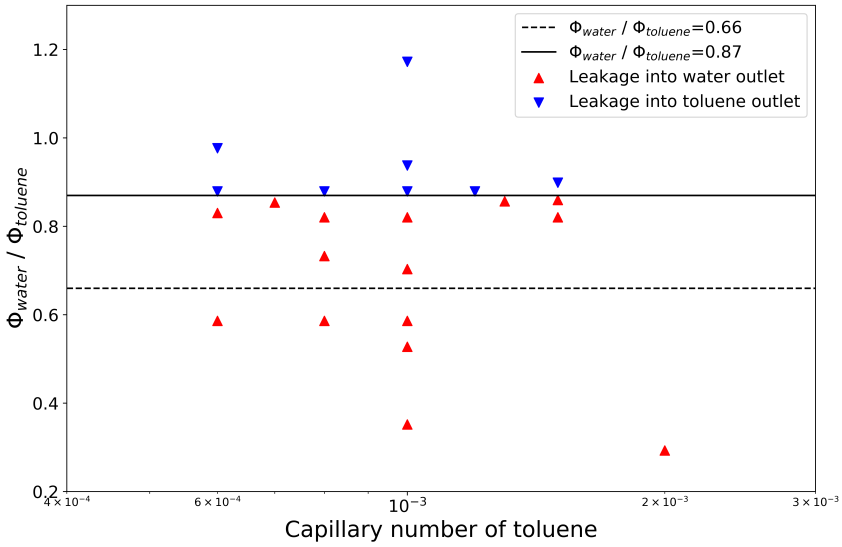


Figure 4.7: Leakage direction in a water-toluene system as a function of toluene Capillary number and the volumetric flow rate ratio.

Pohar et al. have done a quantitative research into parallel flow patterns in a Y-channel. They proposed a correlation for the flow rate ratio required in order to pin the interface between two phases at the middle of the channel outlet [68]:

$$\frac{\Phi_1}{\Phi_2} = \left(\frac{\rho_1 \nu_1}{\rho_2 \nu 2}\right)^{-0.76}. \tag{4.1}$$

For water and toluene this results in a flow rate of 0.66. The experimental results by Z. Liu confirm this flow ratio to be the transition point between leakage into the water outlet to leakage into the toluene outlet. On the contrary, the phase field method was unable to retrieve the transition line in leakage direction [14]. The flow rate of 0.66 has been plotted in Figure 4.7 as a dotted line. The simulations clearly do not match this transition point. Even though the patterns do not behave according to the transition flow ratio determined by Pohar et al., there is a clear transition point between leakage direction that seems to be consistent over the entire simulation range of toluene Capillary numbers. The flow ratio that corresponds to this line lies around 0.87. As this line is also valid for low toluene Capillary numbers, it is therefore unlikely that the possible appearance of transition flow is a nuisance.

4.2.1. LEAKAGE TYPES

For volumetric flow rate ratios far away from the transition line the leakage exhibits parallel flow behaviour in the outlet. This is the case for both leakage into the water outlet and leakage into the toluene outlet, as can be seen in figures 4.8 and 4.9.

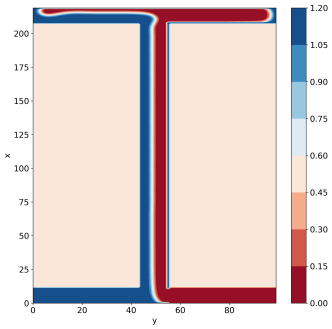


Figure 4.8: Parallel leakage of toluene into the water outlet for a toluene Capillary number of 8×10^{-4} and a volumetric flow rate ratio of 0.59 at time step 70000.

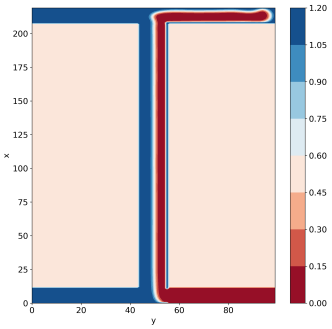


Figure 4.9: Parallel leakage of water into the toluene outlet for a toluene Capillary number of 10^{-3} and a volumetric flow rate ratio of 1.17 at time step 120000.

Closer to the transition line the leakage takes the form of droplet flow, both for leakage into the water outlet and leakage into the toluene outlet. The droplets formed by toluene into the water are hydrophobic and should show less affinity with the wall, as the wall is simulated to be more wettable for water. This effect is correctly retrieved by the model as the toluene droplets (Figure 4.10) form a larger angle to the wall at the contact point compared to the water droplets (Figure 4.11).

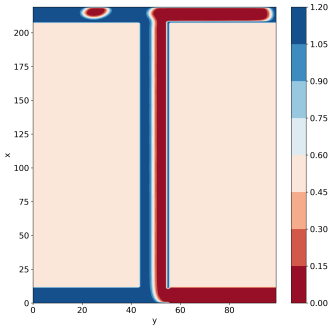


Figure 4.10: Droplet leakage of toluene into the water outlet for a toluene Capillary number of 8×10^{-4} and a volumetric flow rate ratio of 0.73 at time step 70000.

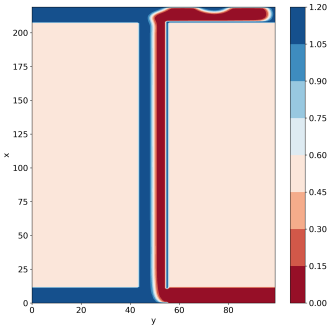


Figure 4.11: Droplet leakage of water into the toluene outlet for a toluene Capillary number of 6×10^{-4} and a volumetric flow rate ratio of 0.98 at time step 100000.

It is remarkable that droplet flow into the toluene outlet is observed instead of parallel flow, even though the wall is hydrophilic. Upon investigating the simulation results it can be seen that the rightmost contact point between the wall, water and toluene progresses very slowly into the toluene outlet, as if it starts to form parallel flow. However, at some point this parallel flow is pinched of by the toluene as it is forced against the outlet top wall, forming a droplet against the wall that gets dragged along by the bulk toluene. From figures 4.12 and 4.13 it can be seen that the process which forms toluene droplets is similar, apart from the fact that the initial advancing flow makes a larger contact angle with the top wall. These observed droplet shapes are in line with experimental research done by Trantidou into the effect of surface wettability on droplet shape [69].

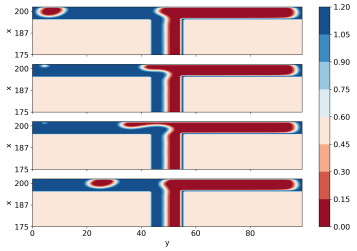


Figure 4.12: Toluene droplet formation leakage over 15000 time steps for a toluene Capillary number of 8×10^{-4} and a volumetric flow rate ratio of 0.73.

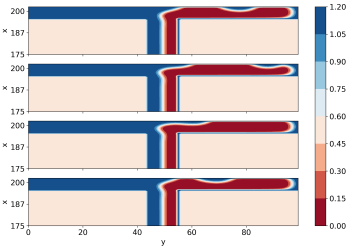


Figure 4.13: Water droplet formation leakage over 15000 time steps for a toluene Capillary number of 6×10^{-4} and a volumetric flow rate ratio of 0.98.

It seems that the interface pinning at the middle of the outlet is less important in contrast to the Y-channel example described by Pohar et al. [68]. Instead, the transition line lies at a volumetric flow rate ratio above which the water flow rate is high enough to block off the toluene flowing to the left. This in turn causes leakage into the toluene outlet as the water creeps along the wall due to the wall being hydrophilic. Hence it does not seem unlikely that the contact angle plays an important role in the determination of the transition line. At higher volumetric flow rate ratios, the water velocity increases to a point where its viscous force is large enough to build a stable interface that does not get pinched of, resulting in parallel flow. The round edge with water around at the end of the toluene in the toluene outlet channel visible in Figure 4.8-4.13 is a result of the outlet boundary condition and is of no concern.

5
CONCLUSION AND RECOMMENDATIONS

In this thesis the behaviour of multiphase flow inside of a microfluidic T-channel has been investigated. A color-gradient multiphase Lattice Boltzmann Method was used to simulate the flow patterns generated by water and n-heptane, in addition to the outlet leakage using water and toluene. The results have been compared to previous experiments with T-channels as well as simulations and experiments done on Y-channels by Z. Liu.

5.1. CONCLUSION

The flow map obtained for the T-channel is dominated by parallel flow in the middle register of n-heptane Capillary numbers (1.25×10^{-4} to 10^{-3}). Several experimental researches found a lesser tendency towards parallel flow in T-channels, often due to the use of different fluids. Generally, the conversions between slug and parallel flow seem to match experimental results.

Compared to the results obtained for a Y-channel by Z. Liu using water and n-heptane, parallel flow occurs for lower Capillary numbers and hence for lower velocities. Parallel flow for lower velocities is a desirable flow pattern as it increases the time available for mass exchange inside the channel. In addition, the range of Capillary numbers that resulted in parallel flow is larger, potentially leading to a broader range of usable operating conditions. High Capillary numbers (above 2×10^{-3}) resulted in distorted initial flow due to the inlet velocity nearing a practical stability limit.

While the droplet formation in the simulations matched experimental research, the transition flow regime as experimentally and numerically found by Z. Liu did not appear. A comparable flow pattern was observed which moved to parallel flow over time. The same lack of transition flow simulation capabilities was also experienced by A. Sudha in Y-channel simulation, making it likely that it is a Lattice Boltzmann Method shortcoming. This shortcoming might be due to the time required by the Lattice Boltzmann Method to establish an initial interface. This same phenomenon also leads to the slug attachment that is observed in parallel flow, especially being significant for low water Capillary numbers (10^{-4} and below) in combination with low volumetric flow ratios.

All simulated cases of parallel flow for the water-toluene system led to leakage in either direction, with no cases without leakage. This corresponds with the experimental results found for a Y-channel by Z. Liu. The lattice Boltzmann method shows a clear improvement over the Phase Field method used by Z. Liu for the simulation of leakage. The transition line between leakage into the water outlet and leakage into the toluene outlet is fixed at 0.87 for all simulated toluene Capillary numbers. This is in disagreement with the analytically computed value for a Y-channel, meaning that the transition between leakage direction for a T-channel lies higher than for a Y-channel. In other words, the amount of water that can flow through the channel for a fixed toluene flow rate before the leakage direction shifts is larger than in the case of a Y-channel.

For flow rates far away from the transition flow rate, the leakage occurred as parallel flow into the outlet channel. Flow rates close to the transition line led to droplet leakage. The phenomena of wetting was successfully retrieved from the leakage simulation as the toluene droplets showed less affinity with the wall compared to the water droplets. As the contact angle plays an important role in the formation of droplets near the flow rate ratio transition line, it is not unlikely that the contact angle may have influence on the position of the transition line.

As neither the T-channel nor the Y-channel exhibited cases without leakage the separation remains an issue. However, the lower boundary for parallel flow found for a T-channel does make it potentially useful for liquid-liquid extraction of radioisotopes.

5.2. RECOMMENDATIONS

The current implementation of the color-gradient Lattice Boltzmann Method is unable to retrieve transition flow for either T-channels nor Y-channels, it is therefore recommended to further investigate the origin of this shortcoming. A possible research includes the implementation of the introduced contact angle hysteresis. In addition a more elaborate research into the influence of outlet boundary conditions is a recommended addition to the Lattice Boltzmann implementation at hand. With a better understanding of the boundary condition influence, simulations over the full length of the channel might be desirable to confirm the observed flow patterns.

Further research into the transition flow rate regarding outlet leakage direction is advised in order to more accurately predict the leakage in a microfluidic T-channel. Experimental research into the leakage at the outlet of such channels is required to confirm this transition. Subsequently, a method to eliminate leakage at the outlet of the channel must be found in order to be able to implement this separation technique successfully.

Lastly, it is recommended to extend the comparison between T and Y-channels to the effect of the angle between the two channels. Instead of focusing on 180°, inlet angles greater than 180° or smaller than a typical Y-channel might offer an alternative.

BIBLIOGRAPHY

[1] M. Darekar, K. K. Singh, S. Mukhopadhyay, and K. T. Shenoy, *Liquid-liquid two-phase flow patterns in y-junction microchannels,* Industrial and Engineering Chemistry Research **56**, 12215 (2017).

[2] E. Rebrov, M. de Croon, and J. Schouten, *Design of a microstructured reactor with integrated heat-exchanger for optimum performance of a highly exothermic reaction,* Catalysis Today **69**, 183 (2001).

[3] M.-A. Schneider, T. Maeder, P. Ryser, and F. Stoessel, *A microreactor-based system for the study of fast exothermic reactions in liquid phase: characterization of the system,* Chemical Engineering Journal **101**, 241 (2004).

[4] N. Sen, K. K. Singh, S.Mukhopadhyay, K. Shenoy, and S.K.Ghosh, *Continuous, solvent free, high temperature synthesis of ionic liquid 1-butyl-3-methylimidazolium bromide in a microreactor,* BARC Newsletter **334**, 20 (2013).

[5] J. C. Brandt and T. Wirth, *Controlling hazardous chemicals in microreactors: Synthesis with iodine azide,* Beilstein Journal of Organic Chemistry **5** (2009).

[6] X. Zhang, S. Stefanick, and F. J. Villani, *Application of microreactor technology in process development,* Organic Process Research & Development **8**, 455 (2004), https://doi.org/10.1021/op034193x .

[7] M. Darekar, K. K. Singh, S. Mukhopadhyay, and K. T. Shenoy, *Single-stage microscale extraction: Studies with single microbore tubes and scale-up,* Separation and Purification Technology **158**, 160 (2016).

[8] A. Tonkovich, D. Kuhlmann, A. Rogers, J. McDaniel, S. Fitzgerald, R. Arora, and T. Yuschak, *Microchannel technology scale-up to commercial capacity,* Chemical Engineering Research and Design **83**, 634 (2005), 7th World Congress of Chemical Engineering.

[9] M. L. P. Reddy, T. P. Rao, and A. D. Damodaran, *Liquid-liquid extraction processes for the separation and purification of rare earths,* Mineral Processing and Extractive Metallurgy Review **12**, 91 (1993), https://doi.org/10.1080/08827509508935254 .

[10] S. Lee, M. Park, C. Park, B. Kim, J. Lee, S. Choi, S. Nam, S. Park, and Y. Choy, *Implantable micro-chip for controlled delivery of diclofenac sodium,* Journal of Controlled Release **196**, 52 (2014), publisher Copyright: © 2014 Elsevier B.V. All rights reserved.

[11] P. G. Mazzola, A. M. Lopes, F. A. Hasmann, A. F. Jozala, T. C. V. Penna, P. de Oliveira Magalhães, C. de Oliveira Rangel-Yagui, and A. Pessoa, *Liquid–liquid extraction of biomolecules: an overview and update of the main techniques,* Journal of Chemical Technology & Biotechnology **83**, 143 (2008).

[12] D. Ciceri, J. M. Perera, and G. W. Stevens, *The use of microfluidic devices in solvent extraction,* Journal of Chemical Technology & Biotechnology **89**, 771 (2014), https://onlinelibrary.wiley.com/doi/pdf/10.1002/jctb.4318 .

[13] N.-T. Nguyen, S. T. Wereley, and S. S. A. Mousavi, *Fundamentals and applications of Microfluidics* (Artech House, 2019).

[14] Z. Liu, *Simulating multiphase flows inside a microfluidic channel with the phase field method*, Ph.D. thesis, TU Delft (2022).

[15] J. yuan Qian, X. juan Li, Z. Wu, Z. jiang Jin, and B. Sunden, *A comprehensive review on liquid–liquid two-phase flow in microchannel: flow pattern and mass transfer,* Microfluidics and Nanofluidics **23** (2019), 10.1007/s10404-019-2280-4.

[16] 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 **63**, 4035 (2008).

[17] A. Salim, M. Fourar, J. Pironon, and J. Sausse, *Oil-water two-phase flow in microchannels: Flow patterns and pressure drop measurements,* Canadian Journal of Chemical Engineering **86**, 978 (2008).

[18] M. Kashid and L. Kiwi-Minsker, *Quantitative prediction of flow patterns in liquid-liquid flow in micro-capillaries,* Chemical Engineering and Processing: Process Intensification **50**, 972 (2011).

[19] Y. Zhao, G. Chen, and Q. Yuan, *Liquid-liquid two-phase flow patterns in a rectangular microchannel,* AIChE Journal **52**, 4052 (2006).

[20] L. Wu, X. Liu, Y. Zhao, and Y. Chen, *Role of local geometry on droplet formation in axisymmetric microfluidics,* Chemical Engineering Science **163**, 56 (2017).

[21] L. Derzsi, M. Kasprzyk, J. P. Plog, and P. Garstecki, *Flow focusing with viscoelastic liquids,* Physics of Fluids **25**, 092001 (2013), https://doi.org/10.1063/1.4817995 .

[22] A. Salim, M. Fourar, J. Pironon, and J. Sausse, *Oil–water two-phase flow in microchannels: Flow patterns and pressure drop measurements,* The Canadian Journal of Chemical Engineering **86**, 978 (2008), https://onlinelibrary.wiley.com/doi/pdf/10.1002/cjce.20108 .

[23] Y. Zhao, Y. Su, G. Chen, and Q. Yuan, *Effect of surface properties on the flow characteristics and mass transfer performance in microchannels,* Chemical Engineering Science **65**, 1563–1570 (2010).

[24] F. Y. Ushikubo, F. S. Birribilli, D. R. Oliveira, and R. L. Cunha, *Y- and t-junction microfluidic devices: effect of fluids and interface properties and operating conditions,* Microfluidics and Nanofluidics **17**, 711 (2014).

[25] L. Shui, J. C. Eijkel, and A. van den Berg, *Multiphase flow in microfluidic systems - control and applications of droplets and interfaces,* Advances in Colloid and Interface Science **133**, 35 (2007).

[26] S. Goyal, A. V. Desai, R. W. Lewis, D. R. Ranganathan, H. Li, D. Zeng, D. E. Reichert, and P. J. Kenis, *Thiolene and sifel-based microfluidic platforms for liquid-liquid extraction,* Sensors and Actuators, B: Chemical **190**, 634 (2014).

[27] P. Martini, A. Adamo, N. Syna, A. Boschi, L. Uccelli, N. Weeranoppanant, J. Markham, and G. Pascali, *Perspectives on the use of liquid extraction for radioisotope purification,* Molecules **24**, 334 (2019).

[28] D. Erickson, *Towards numerical prototyping of labs-on-chip: Modeling for integrated microfluidic devices,* Microfluidics and Nanofluidics **1**, 301–318 (2005).

[29] W. Jeon and C. B. Shin, *Design and simulation of passive mixing in microfluidic systems with geometric variations,* Chemical Engineering Journal **152**, 575 (2009).

[30] M. M. Dupin, I. Halliday, and C. M. Care, *Simulation of a microfluidic flow-focusing device,* Phys. Rev. E **73**, 055701 (2006).

[31] I. Cimrák, M. Gusenbauer, and T. Schrefl, *Modelling and simulation of processes in microfluidic devices for biomedical applications,* Computers & Mathematics with Applications **64**, 278 (2012), mathematical Methods and Models in Biosciences.

[32] U. Hashim, P. N. A. Diyana, and T. Adam, *Numerical simulation of microfluidic devices,* in *2012 10th IEEE International Conference on Semiconductor Electronics (ICSE)* (2012) pp. 26–29.

[33] C. E. Brennen, *Fundamentals of multiphase flow* (Cambridge University Press, 2005) p. 345.

[34] K. Timm, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, *The lattice Boltzmann method: Principles and practice* (Springer International Publishing, 2017).

[35] Y. Yuan and T. R. Lee, *Contact angle and wetting properties,* Springer Series in Surface Sciences **51**, 3 (2013).

[36] H. Bruus, *Theoretical Microfluidics* (Oxford University Press, 2008) Chap. 7, pp. 123–136.

[37] F. Bottiglione, G. Carbone, and B. N. J. Persson, *Fluid contact angle on solid surfaces: Role of multiscale surface roughness,* The Journal of Chemical Physics **143**, 134705 (2015), https://aip.scitation.org/doi/pdf/10.1063/1.4932104 .

[38] D. Bonn, J. Eggers, J. Indekeu, J. Meunier, and E. Rolley, *Wetting and spreading,* Rev. Mod. Phys. **81**, 739 (2009).

[39] T. D. Blake, *The physics of moving wetting lines,* Journal of Colloid and Interface Science **299**, 1 (2006).

[40] H. Liu, Y. Ju, N. Wang, G. Xi, and Y. Zhang, *Lattice boltzmann modeling of contact angle and its hysteresis in two-phase flow with large viscosity difference,* Phys. Rev. E **92**, 033306 (2015).

[41] P. Cheng, H.-Y. Wu, and F.-J. Hong, *Phase-Change Heat Transfer in Microsystems,* Journal of Heat Transfer **129**, 101 (2006), https://asmedigitalcollection.asme.org/heattransfer/article-pdf/129/2/101/5632638/101_1.pdf .

[42] S. V. Patankar, *Numerical Heat Transfer and fluid flow* (Routledge, 1980).

[43] J. H. Ferziger, M. Perić, and R. L. Street, *Computational methods for fluid dynamics* (Springer, 2020).

[44] I. Steinbach, *Phase-field models in materials science,* Modelling and Simulation in Materials Science and Engineering **17**, 073001 (2009).

[45] M. Renardy, Y. Renardy, and J. Li, *Numerical simulation of moving contact line problems using a volume-of-fluid method,* Journal of Computational Physics **171**, 243 (2001).

[46] C. Hirt and B. Nichols, *Volume of fluid (vof) method for the dynamics of free boundaries,* Journal of Computational Physics **39**, 201 (1981).

[47] Y. Chen, R. Mertz, and R. Kulenovic, *Numerical simulation of bubble formation on orifice plates with a moving contact line,* International Journal of Multiphase Flow **35**, 66 (2009).

[48] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations,* Journal of Computational Physics **79**, 12 (1988).

[49] H. Huang, M. C. Sukop, and X.-Y. Lu, *Rothman-keller multiphase lattice boltzmann model,* in *Multiphase lattice boltzmann methods: Theory and application* (John Wiley and Sons, Inc., 2015) Chap. 4, pp. 1–54, 1st ed.

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

[51] T. Reis and T. N. Phillips, *Lattice boltzmann model for simulating immiscible two-phase flows,* Journal of Physics A: Mathematical and Theoretical **40**, 4033 (2007).

[52] H. Liu, A. J. Valocchi, and Q. Kang, *Three-dimensional lattice boltzmann model for immiscible two-phase flow simulations,* Phys. Rev. E **85**, 046309 (2012).

[53] Y. Ba, H. Liu, Q. Li, Q. Kang, and J. Sun, *Multiple-relaxation-time color-gradient lattice boltzmann model for simulating two-phase flows with high density ratio,* Physical Review E **94** (2016), 10.1103/PhysRevE.94.023310.

[54] D. H. Rothman and J. M. Keller, *Immiscible cellular-automaton fluids,* Journal of Statistical Physics **52**, 1119 (1988).

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

[56] H. HUANG, J.-J. HUANG, X.-Y. LU, and M. C. SUKOP, *On simulations of high-density ratio flows using color-gradient multiphase lattice boltzmann models,* International Journal of Modern Physics C **24**, 1350021 (2013), https://doi.org/10.1142/S0129183113500216 .

[57] Z. Guo, C. Zheng, and B. Shi, *Discrete lattice effects on the forcing term in the lattice boltzmann method,* Phys. Rev. E **65**, 046308 (2002).

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

[59] D. Grunau, S. Chen, and K. Eggert, *A lattice boltzmann model for multiphase fluid flows,* Physics of Fluids A: Fluid Dynamics **5**, 2557 (1993), https://doi.org/10.1063/1.858769 .

[60] H. Ding and P. D. M. Spelt, *Wetting condition in diffuse interface simulations of contact line motion,* Phys. Rev. E **75**, 046708 (2007).

[61] J.-J. Huang, H. Huang, and X. Wang, *Numerical study of drop motion on a surface with stepwise wettability gradient and contact angle hysteresis,* Physics of Fluids **26**, 062101 (2014), https://doi.org/10.1063/1.4880656 .

[62] A. J. C. Ladd, *Numerical simulations of particulate suspensions via a discretized boltzmann equation. part 1. theoretical foundation,* Journal of Fluid Mechanics **271**, 285–309 (1994).

[63] Q. Lou, Z. Guo, and B. Shi, *Evaluation of outflow boundary conditions for two-phase lattice boltzmann equation,* Physical Review E - Statistical, Nonlinear, and Soft Matter Physics **87** (2013), 10.1103/PhysRevE.87.063301.

[64] J. Jovanovic, E. Rebrov, T. Nijhuis, M. Kreutzer, V. Hessel, and J. Schouten, *Liquid-liquid flow in a capillary microreactor: Hydrodynamic flow patterns and extraction performance,* Industrial & Engineering Chemistry Research **51**, 1020 (2012).

[65] F. De Groot, *Investigating the effect of microchannel junction geometry on two-phase flow using the Lattice Boltzmann Method*, Bachelor's thesis, TU Delft (2021).

[66] X.-B. Li, F.-C. Li, J.-C. Yang, H. Kinoshita, M. Oishi, and M. Oshima, *Study on the mechanism of droplet formation in t-junction microchannel,* Chemical Engineering Science **69**, 340 (2012).

[67] P. Guillot and A. Colin, *Stability of parallel flows in a microchannel after a t junction,* Physical Review E - Statistical, Nonlinear, and Soft Matter Physics **72** (2005), 10.1103/PhysRevE.72.066301.

[68] A. Pohar, M. Lakner, and I. Plazl, *Parallel flow of immiscible liquids in a microreactor: Modeling and experimental study,* Microfluidics and Nanofluidics **12**, 307 (2012).

[69] T. Trantidou, Y. Elani, E. Parsons, and O. Ces, *Hydrophilic surface modification of pdms for droplet microfluidics using a simple, quick, and robust method via pva deposition,* Microsystems & amp; Nanoengineering **3** (2017), 10.1038/micronano.2016.91.

A

APPENDIX | MODEL

A.1. UNIT CONVERSION METHOD

Converting the physical units into simulation units is done using the Capillary and Reynolds numbers:

$$Re = \frac{uL}{\nu}, \qquad Ca = \frac{\nu\rho u}{\sigma}. \tag{A.1}$$

The channel width $L = 100\mu$m is $L^* = 10$lu in the simulation. Resulting in a fixed conversion factor of $C_L = 10\mu$m. The density conversion and interface tension conversion are also fixed but depend on the combination of liquids. The known properties therefor are L, L^* (channel) and $\rho, \rho^*, \nu, \sigma, \sigma^*$ (fluids). First, the general approach is explained followed by the results obtained for water-heptane and water-toluene.

The first step is to pick a fixed Capillary number, say 10^{-4}, and derive the physical velocity from it as the viscosity, density and interface tension are known:

$$Ca = \frac{\nu\rho u}{\sigma} = 10^{-4} \quad \Rightarrow \quad u = \frac{\sigma Ca}{\nu\rho}. \tag{A.2}$$

With the physical velocity known the corresponding Reynolds number Re can be calculated from the known physical channel width and fluid viscosity. The law of similarity states:

$$Ca = \frac{\nu\rho u}{\sigma} = \frac{\nu^*\rho^* u^*}{\sigma^*} = Ca^* \tag{A.3}$$

$$Re = \frac{uL}{\nu} = \frac{u^* L^*}{\nu^*} = Re^*. \tag{A.4}$$

Using this law of similarity, two equations with two unknowns: ν^* and u^* remain. These can be found by substituting u^* from Ca into Re^* to find ν^*:

$$u^* = \frac{\sigma^* Ca}{\nu^*\rho^*} \quad \Rightarrow \quad (\nu^*)^2 = \frac{\sigma^* L^* Ca}{\rho^* Re}. \tag{A.5}$$

With ν^* known the required simulation relaxation time τ^* can be determined:

$$\tau^* = 3\nu^* + \frac{1}{2}. \tag{A.6}$$

Applying this procedure to the combination of water and n-heptane and water and toluene based on their properties defined in table 3.1 results in the model parameters in tables 3.2 and 3.3. The value of α was determined to achieve optimal stability.

A.2. MRT MOMENT OPERATOR

Equations A.7 and A.8 show the moment operator matrices used in combination with the D2Q9 model.

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{A.7}$$

$$\mathbf{M}^{-1} = \frac{1}{36} \begin{bmatrix} 4 & -4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & -1 & -2 & 6 & -6 & 0 & 0 & 9 & 0 \\ 4 & -1 & -2 & 0 & 0 & 6 & -6 & -9 & 0 \\ 4 & -1 & -2 & -6 & 6 & 0 & 0 & 9 & 0 \\ 4 & -1 & -2 & 0 & 0 & -6 & 6 & -9 & 0 \\ 4 & 2 & 1 & 6 & 3 & 6 & 3 & 0 & 9 \\ 4 & 2 & 1 & -6 & -3 & 6 & 3 & 0 & -9 \\ 4 & 2 & 1 & -6 & -3 & -6 & -3 & 0 & 9 \\ 4 & 2 & 1 & 6 & 3 & -6 & -3 & 0 & -9 \end{bmatrix} \tag{A.8}$$

A.3. RELAXATION TIME INTERPOLATION

The Grunau interpolation scheme [49, 59] uses a dimensionless density ratio parameter $\psi(\mathbf{x})$ to adjust the relaxation time:

$$\psi(\mathbf{x}) = \frac{\rho_1(\mathbf{x}) - \rho_2(\mathbf{x})}{\rho_1(\mathbf{x}) + \rho_2(\mathbf{x})}. \tag{A.9}$$

This parameter is used to determine a local relaxation time $\tau(\mathbf{x})$ at location $\mathbf{x}$.

$$\tau(\mathbf{x}) = \begin{cases} \tau_1 & \psi > \delta \\ g_1(\psi) & \delta \geq \psi > 0 \\ g_2(\psi) & 0 \geq \psi \geq -\delta \\ \tau_2 & \psi < -\delta \end{cases} \tag{A.10}$$

In equation A.10 τ_1 and τ_2 are the relaxation times for the two phases. δ is a parameter that governs the interface thickness. Based on convention it is set to $\delta = 0.98$ [49]. At locations well outside the interface width, the individual relaxation times are used. In the interface region the interpolation functions g_1 and g_2 are used:

$$\begin{cases} g_1(\psi) = s_1 + s_2\psi + s_3\psi^2, \\ g_2(\psi) = t_1 + t_2\psi + t_3\psi^2, \end{cases} \tag{A.11}$$

with the constants s_{1-3} and t_{1-3} given by:

$$s_1 = t_1 = 2\frac{\tau_1\tau_2}{\tau_1+\tau_2}, \tag{A.12}$$

$$s_2 = 2\frac{\tau_1 - s_1}{\delta}, \tag{A.13}$$

$$s_3 = -\frac{s_2}{2\delta}, \tag{A.14}$$

$$t_2 = 2\frac{t_1 - \tau_2}{\delta}, \tag{A.15}$$

$$t_3 = \frac{t_2}{2\delta}. \tag{A.16}$$

B

APPENDIX | FLOW PATTERNS

PINCHING MOTION AT INLET

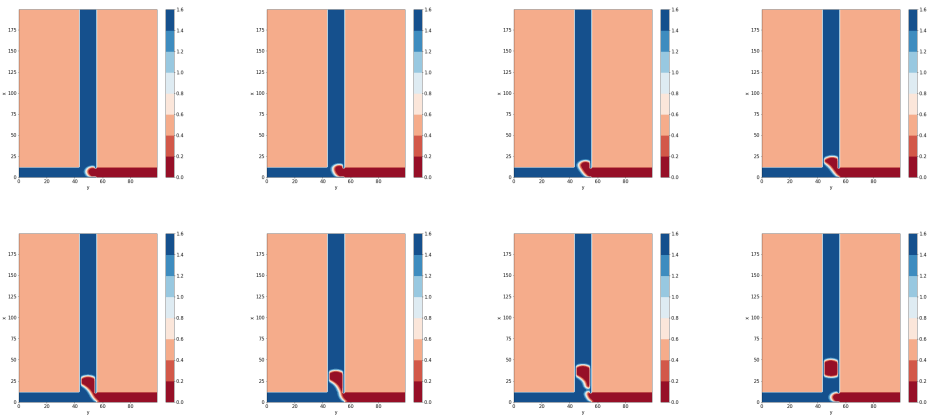


Figure B.1: Pinching occurring at the T-junction for slug flow at $Ca_{n-heptane} = 5 \times 10^{-5}$ and $Ca_{water} = 2 \times 10^{-4}$. Time steps 15000-50000 have been plotted every other 5000 time steps.

PINCHING MOTION IN CHANNEL

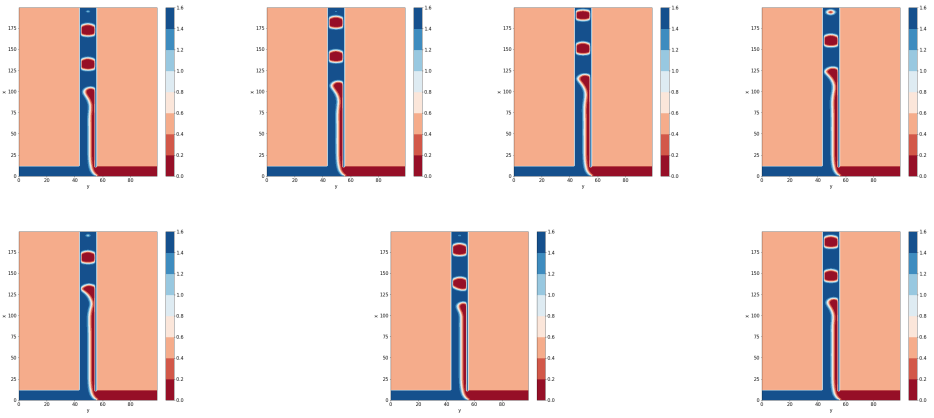


Figure B.2: Pinching occuring inside the channel for parallel breakup flow at $Ca_{n-heptane} = 6 \times 10^{-5}$ and $Ca_{water} = 3 \times 10^{-4}$. Time steps 220000-250000 have been plotted every other 5000 time steps.

C

APPENDIX | CODE

```
# -*- coding: utf-8 -*-
"""
Microchannel code to simulate outlet for Toluene and water

@author: asudha1
Adapted by hbaetsen from 26-05 onwards.
"""
#%% Import Packages
import numpy as np
import time
import scipy.io as sio
import numba as nb
import math
import matplotlib.pyplot as plt
import os
#from sys import exit
#%% parameters

####### CHANGE EACH ITERATION ########
runno = 'leakage_toluene_22'
############  END CHANGE #############

start=time.time() #Determine code execution time
lx=220 #grid dimensions
ly=100
cw= int(10) # Channel width

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= 0.0361; #interface tension # Start value 0.05
df= 0.000000015; #body force, if needed
alphar= 0.565; #parameters that determine initial density
alphab= 0.5;
rhori= 1.1494; #initial density water
rhobi= 1;   #initial density toluene
tm= 300000; #max time steps
uib= -0.0068952344570719125 #inlet velocity toluene
uir= 0.0060613105762369 88  #inlet velocity water
#Relaxation time
taur=0.5280           # Water
taub=0.5188           # toluene
#Parameters for calculating pressure
csqr=3*(1-alphar)/5; csqb= 3*(1-alphab)/5;

theta_w = 49*np.pi/180 # contact angle water (radians): (Z. Liu, 2022)
#theta=  math.tan(43.4*np.pi/180) #tan of contact angle
theta = math.tan(np.pi/2 - theta_w)

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 < (cw+1)):
                if (y == 0):
                    obst[x,y] = 0.5 # Left inlet
                elif (y == ly-1):
                    obst[x,y] = 0.7 # Right inlet
            elif (x == (cw+1)):
                if (y < (ly//2 -(cw//2 +1)) or y > (ly//2 +(cw//2))):
                    obst[x,y] = 2 # Bottom inlet wall
            elif ((cw+1) < x < (lx-2-cw)):
                if (y == (ly//2 -(cw//2 +1))):
                    obst[x,y] = 3 # Right wall
                elif (y == (ly//2 +(cw//2))):
                    obst[x,y] = 4 # Left wall
                elif ((ly//2 -(cw//2 +1)) < y < (ly//2 +(cw//2))):
                    obst[x,y] = 0
                else:
                    obst[x,y] = -1 # Empty Space
            elif (x== (lx-2-cw)): # outlet wall channel side
                if (y < (ly//2 -(cw//2 +1)) or y > (ly//2 +(cw//2))):
                    obst[x,y] = 8
            elif ((lx-2-cw) < x < (lx-1)):
                if (y == 0):
                    obst[x,y] = 11 # Left Outlet
                elif (y == ly-1):
                    obst[x,y] = 12 # Right Outlet
            elif (x==lx-1): # Outlet bottom wall
                obst[x,y] = 7


    # Inlet junction corners
    obst[(cw+1),(ly//2 -(cw//2 +1))] = 5
    obst[(cw+1),(ly//2 +(cw//2))] = 6

    # Outlet junction corners
    obst[(lx-cw-2),(ly//2 -(cw//2 +1))] = 9
    obst[(lx-cw-2),(ly//2 +(cw//2))] = 10

    # Defining initial densities -> Also at the boundaries
    for x in nb.prange(0,lx):
        for y in nb.prange(0,ly):
            if (x <= (cw+1) and y <= (ly//2 +(cw//2))):
                rhor[x,y] = rhori
            elif (x <= (cw+1) and y > (ly//2 +(cw//2))):
                rhob[x,y] = rhobi
            elif ((ly//2 -(cw//2 +1)) <= y <= (ly//2 +(cw//2)) and x > (cw+1)):
                rhor[x,y] = rhori
                rhob[x,y] = 0
            elif ((x >= lx-2-cw)):
                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]
        # These equations are per MP LB, they differ from Ba et al as the factor 3 is here
        # included in the lattice speed of sound.
        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 (fill in the otherwise missing populations)
            if (((obst[x,y] == 1) and (y == 0)) or ((obst[x,y] ==8) 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))  or ((obst[x,y] ==8) 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)) or ((obst[x,y] ==8) 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)) or ((obst[x,y] ==7) 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)) or ((obst[x,y] ==7) 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)) or ((obst[x,y] ==7) 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]

        # Outlets
        elif (obst[x,y] == 11): # Left outlet
            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] == 12): # Right outlet
            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]

        # Inlet junction corners
        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]

        # Outlet junction corners
        elif (obst[x,y] == 11):
            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[8,x+1,y-1] = f[8,x,y]

        elif (obst[x,y] == 12):
            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[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
            # INLET SIDE & OUTLET SIDE
            elif (((obst[x,y] == 1) and (y == 0)) or ((obst[x,y] == 8) 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)) or ((obst[x,y] == 8) 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)) or ((obst[x,y] == 8) and (y == ly-1))):
                f[8,x+1,y-1] = fc[6,x+1,y-1]
            elif (((obst[x,y] == 2) and (y == 0)) or ((obst[x,y] == 7) 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)) or ((obst[x,y] == 7) and (0 < y < ly-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)) or ((obst[x,y] == 7) 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] == 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]

            # Corners of T-junction
            # INLET
            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
            elif (obst[x,y] == 9):
                f[1,x+1,y] = fc[3,x+1,y]
                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]
                f[8,x+1,y-1] = fc[6,x+1,y-1]
            elif (obst[x,y] == 10):
                f[1,x+1,y] = fc[3,x+1,y]
                f[4,x,y-1] = fc[2,x,y-1]
                f[5,x+1,y+1] = fc[7,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]

            #Outlets, Lou et al 2011
            elif (obst[x,y] == 11):
                f[2,x,y+1]= (f[2,x,y+1]+f[2,x,y+2]*np.mean(uy[(ly-cw-1):-1,y+2]))/ \
                (1+np.mean(uy[(ly-cw-1):-1,y+2]))
                f[5,x,y+1]= (f[5,x,y+1]+f[5,x,y+2]*np.mean(uy[(ly-cw-1):-1,y+2]))/ \
```

```
                (1+np.mean(uy[(ly-cw-1):-1,y+2]))
                f[6,x,y+1]= (f[6,x,y+1]+f[6,x,y+2]*np.mean(uy[(ly-cw-1):-1,y+2]))/ \
                (1+np.mean(uy[(ly-cw-1):-1,y+2]))

            elif (obst[x,y] == 12):
                f[4,x,y-1]= (f[4,x,y-1]+f[4,x,y-2]*np.mean(uy[(ly-cw-1):-1,y-2]))/ \
                (1+np.mean(uy[(ly-cw-1):-1,y-2]))
                f[7,x,y-1]= (f[7,x,y-1]+f[7,x,y-2]*np.mean(uy[(ly-cw-1):-1,y-2]))/ \
                (1+np.mean(uy[(ly-cw-1):-1,y-2]))
                f[8,x,y-1]= (f[8,x,y-1]+f[8,x,y-2]*np.mean(uy[(ly-cw-1):-1,y-2]))/ \
                (1+np.mean(uy[(ly-cw-1):-1,y-2]))


#   #Corners of outlet boundaries
    f[6,lx-1,1] = fc[8,lx-1,1]
    f[5,lx-2-cw,1] = fc[7,lx-2-cw,1]
    f[7,lx-1,-2] = fc[5,lx-1,-2]
    f[8,lx-2-cw,-2] = fc[6,lx-2-cw, -2]

    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) or (obst[i,j] == 8)):
                        #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) or (obst[i,j] == 7)):
                        #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 - INLET
                        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

                        # Junction Corners - OUTLET
                        if((obst[i,j] == 8)):
                            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] == 9)):
                            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

                        #Outlets
                        if(obst[i,j]==11):
                            rhon[i,j]=rhon[i,j+1]
                        if(obst[i,j]==12):
                            rhon[i,j]=rhon[i,j-1]

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

            # Outlet corners
            rhon[lx-cw-2,0] = (rhon[lx-cw-1,0]+rhon[lx-cw-2,1])/2
            rhon[lx-1,0] = (rhon[lx-2,0]+rhon[lx-1,1])/2
            rhon[lx-cw-2,ly-1] = (rhon[lx-cw-1,ly-1]+rhon[lx-cw-2,ly-2])/2
            rhon[lx-1,ly-1] = (rhon[lx-2,ly-1]+rhon[lx-1,ly-2])/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)
                     phi= (rhor[i,j]-rhob[i,j])/(rhor[i,j]+rhob[i,j])
                     #Relaxation parameter as a function of position
                     #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 -> values from Liu et al.
                     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 (eq 18-20)
                     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)
    #Gradient of phase field function, some special operations have been done for a few nodes, probably not needed for a T-channel.
    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:
                          # Liu et al eq 20
                          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)])
     mapnaam = 'T_channel_Run_{}'.format(runno)
     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{}'.format(mapnaam, t)
     sa=sio.savemat(fname,dict)
     return sa

#%% Make directory for files to go to.
def makedir(runno):
    try:
```

```
        os.mkdir('T_channel_Run_{}'.format(runno))
    except OSError:
        print('Iteration number already in use, do you wish to overwrite run {}'.format(runno))
        answer = input()
        if answer == 'yes':
            return
        else:
            exit()

#%% 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)

# Create the directory for the outputfiles.
makedir(runno)

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)

            if (t%(5*nw)==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)
                intermediate = time.time()
                print('Running time', intermediate-start)
end= time.time()
print('Running Time:',end-start)

#End
```