Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Analysis of two-phase flows under microgravity (spatial) conditions using OpenFOAM

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Carlos Moreno Tavira

DIRECTOR: Santiago Arias Calderón

DATA: 16 de juliol del 2020

Títol: Anàlisi de flux bifàsic sota condicions de microgravetat (espacial) emprant OpenFOAM

Autor: Carlos Moreno Tavira

Director: Santiago Arias Calderón

Data: 16 de juliol de 2020

Resum

Els fluxos bifàsics han guanyat importància al llarg dels anys gràcies a les útils i múltiples aplicacions que tenen en sistemes espacials. Per exemple, els fluxos bifàsics s'utilitzen als micro-canals de les cel·les de combustible, en control de fluids en Sistemes de Suport Vital i Control de l'Ambient (ECLSS) o en sistemes de control tèrmic. Tot i això, molts dels problemes relacionats amb fluxos bifàsics en condicions de microgravetat segueixen oberts.

En aquest estudi es duen a terme simulacions numèriques de fluxos bifàsics gas-líquid en un capil·lar en forma de T. Com a conseqüència de la interacció entre l'aire i l'aigua, es formen bombolles. La geometria utilitzada és la mateixa que en [1, 2, 3] per tal de fer comparacions fidedignes amb els resultats experimentals obtinguts a la literatura esmentada. S'utilitza OpenFOAM com a *software* principal per fer les simulacions, i ParaView i MATLAB com a eina de post-processament. Com a *solver* s'ha escollit *InterFoam* perquè utilitza un mètode *Volume of Fluid* (VOF) del tipus incompressible, immiscible i isotèrmic.

Abans de configurar els casos definitius de les simulacions, es van realitzar certes validacions. Aquestes validacions estaven relacionades amb la longitud adequada del capil·lar per obtenir un flux completament desenvolupat, amb la qualitat apropiada de la malla per extreure bons resultats i mantenir un nivell acceptable de complexitat computacional, amb l'angle de contacte òptim per tal d'aconseguir un comportament proper a la realitat en termes d'adherència a les parets, i amb la localització adequada de les superfícies de mesura encarregades d'extreure les dades.

Les bombolles són analitzades en termes de la freqüència de generació, volum, longitud i velocitat. La dispersió en el volum de les bombolles es quantifica utilitzant l'índex de polidispersitat. S'ha mesurat la pressió relativa al centre de la geometria emprant una sonda. S'han fet comparacions visuals entre les bombolles de les simulacions i dels experiments. Finalment, els resultats de les simulacions s'ajusten qualitativament als experimentals, validant així la Dinàmica de Fluids Computacional (CFD) com una eina alternativa per estudiar fluxos bifàsics en condicions de microgravetat.

Title: Analysis of two-phase flows under microgravity (spatial) conditions using OpenFOAM

Author: Carlos Moreno Tavira

Advisor: Santiago Arias Calderón

Date: July 16th, 2020

Overview

Two-phase flows have gained importance over the last years due to their multiple and useful applications in space systems. For example, two-phase flows are used in fuel cells micro-channel networks, in the fluid management of Environmental Control and Life Support Systems (ECLSS) or in thermal management systems. However, many problems regarding two-phase flows in microgravity conditions are still open, so further research is needed.

In this study, numerical simulations of gas-liquid two-phase flow are performed in a T-junction capillary. Bubbles are formed as a consequence of the interaction between air and water. The geometry used is the same as in [1, 2, 3] in order to make reliable comparisons with the results extracted from the laboratory experiments performed in the mentioned literature. OpenFOAM is used as the main software for the simulations, and ParaView and MATLAB are used to post-process the data. *InterFoam* is selected as the solver since it uses an incompressible, immiscible and isothermal Volume of Fluid (VOF) method.

Some validations were made before setting up the definitive cases of the simulations. These validations were related to the adequate capillary length in order to obtain fully-developed flows, to the appropriate mesh quality to get good results and maintain an acceptable computational complexity, to the optimal contact angle value to get close to reality bubble behavior in terms of adherence to the walls, and to the right location of the sampling surfaces responsible for extracting the data. An analysis of the fluid velocity profiles along both of the capillaries of the T-junction was also made.

Bubbles are analyzed in terms of their generating frequency, volume, length and velocity. Bubble volume dispersion is quantified using the polydispersity index. A pressure probe is used to measure the gauge pressure at the very center of the T-junction. Visual comparisons are made between simulation bubbles and experimental bubbles. In the end, the results of the simulations qualitatively fitted the experimental data, validating Computational Fluid Dynamics (CFD) as an alternative and correct tool to perform two-phase flow studies under microgravity conditions.

“I do not fear computers.
I fear the lack of them.”

Isaac Asimov

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Dimensionless numbers

Bo	Bond number
Ca	Liquid capillary number
Co	Courant number
Re_M	Mixture Reynolds number
We_G	Gas Weber number

Greek symbols

α	Cell phase fraction
α_{avg}	Average surface phase fraction
Δp	Pressure difference
Δt	Simulation time step
Δx	Cell characteristic length
$\varepsilon_{v_{max}}$	Maximum centerline velocity error
Θ	Contact angle
κ	Mean curvature
μ_G	Gas dynamic viscosity
μ_L	Liquid dynamic viscosity
ν_G	Gas kinematic viscosity
ν_L	Liquid kinematic viscosity
ρ_G	Gas density
ρ_L	Liquid density
σ	Gas-liquid surface tension
σ_f	Bubble frequency standard deviation
σ_{L_B}	Bubble length standard deviation
σ_{U_G}	Bubble velocity standard deviation
σ_{V_B}	Bubble volume standard deviation
τ	Viscous stress
ϕ_c	Capillary diameter

Roman symbols

a_0	Initial slope
A	Cross-sectional capillary area
A_B	Area under the alpha curve for a given bubble

A_c	Capillary area
A_i	Equivalent cell area
C_0	Void fraction distribution coefficient
d	Distance between sampling surfaces
f	Bubble frequency
f_{sat}	Saturation frequency
g	Gravitational acceleration
L	Capillary length
L_B	Bubble length
L_h	Hydrodynamic entry length
p	Pressure
Q_G	Gas volumetric flow rate
Q_L	Liquid volumetric flow rate
T_f	Bubble period
T_s	Bubble crossing time from the first surface to the second
U_G	Bubble velocity
U_M	Mixture superficial velocity
U_{SG}	Gas superficial velocity
U_{SL}	Liquid superficial velocity
v_{max}	Maximum velocity achieved at the capillary centerline
V_B	Bubble volume
V_c	Capillary equivalent volume
x, y, z	Cartesian coordinates

ACKNOWLEDGEMENTS

First, I want to thank my advisor Santiago Arias Calderón for his dedication and solicitude to guide me through this vast project. Without his help it would have been impossible to carry out this study. Throughout the last eight months Santiago has been open to schedule meetings in order to track my progress on the project and, in the exceptional situation of the pandemic state of emergency, he has been available to arrange online meetings and to reply to the emails as soon as possible. At the beginning of the course, I knew barely nothing about multiphase fluids, microgravity or microfluidics and Santiago provided me all the necessary information and tools to correctly perform this study, as well as a continuous guideline to follow.

Second, I would like to thank the computer technicians who helped me with a technical issue related to the EETAC cluster. I explained them the problem and they solved it in a few hours, making it possible for me to continue with the cluster simulations.

Third, I want to show my gratitude to the anonymous users of the online OpenFOAM and ANSYS forums that I visited. They helped me to solve individual and concrete problems that arose during the setup of the simulations. These communities offer altruistic assistance to users around the world and I think that it is mandatory to gratify their work.

Last but not least, I want to thank my family, friends and university colleagues. They have always been present during this journey, giving me the affective support that is needed to avoid mental breakdowns in the toughest moments. Thank you for believing in me.

CHAPTER 1. INTRODUCTION

In this project, numerical simulations of two-phase fluids in a T-junction are performed in microgravity conditions. This first chapter serves as an introduction to microgravity and microfluidics and aims to clarify the motivations of the project.

1.1. Microgravity

The word microgravity is composed by the term micro-, which comes from the Greek *mikros* (very small) and -gravity, which is the phenomenon that keeps all things with mass or energy attracted to each other. So, microgravity means that the perceived accelerations over the body are equivalent to one-millionth (micro, 10^{-6}) of the force of gravity at Earth's surface.

One might think that is impossible to achieve microgravity conditions since it would require to travel very far away into deep space in order to not feel the gravitational pull of any astronomical body. That is, indeed, true. No matter how far you are from Earth, you will still feel a small gravitational pull (from the Earth or from some other massive body). To give some examples: in low Earth orbit (LEO, from 200 to 2000 km above Earth surface), where the International Space Station orbits, the gravitational pull from Earth is 90% of the gravity felt at its surface (approximately $9\ m/s^2$). To reduce the gravity of Earth by a factor of one million, it is needed to be at 6 million kilometers from the planet and to reduce the Sun's gravity the same factor is needed to be 3.7 billion kilometers away [4]. At these huge distances we would achieve microgravity conditions.

So, the next reasonable question that could be asked is why astronauts in the ISS experience weightlessness, if Earth's gravitational pull is $9\ m/s^2$. The reason is that the microgravity term can be a little confusing: it does not refer to absolute gravity, but the perceived acceleration over the body. A less confusing term would be "micro-g" conditions, referring to a small amount of g-forces experienced by the body (almost zero-g conditions).

Then, in order to understand microgravity, the meaning of g-forces must be explained. Zero-g conditions are the conditions of absence of g-forces. A g-force is a measurement of the acceleration that causes the perception of weight. Right now, the average reader of this final degree project should be experiencing a g-force of 1 g, since the reader is sitting (or standing) exclusively subject to Earth's gravity acceleration. If the reader was placed inside a Top Fuel dragster (see Fig. 1.1), an accelerating racing car able to reach a speed of 539 km/h in just 3.62 s [5], it would experience a g-force of 5.4 g.

Fig. 1.1 Miller Lite Top Fuel dragster [5]

If the reader were to jump from, for example, the Tower of Pisa, it would experience almost zero-g conditions since it is free falling towards Earth and not feeling the normal reaction force from the ground.

Something similar happens in space when a satellite orbits a planet. Following the ISS example, astronauts within it are in continuous free fall. That is, the g-forces acting on them are equal to zero. Actually, they are not zero but almost zero, since the ISS is not in perfect vacuum (it is 400 km above the ground and there is still some skin friction drag) and tidal forces, which appear from gradients in the gravitational field, are also present. Therefore, we can say that they are under microgravity conditions.

Over the last 30 years, the top space agencies of the world, such as the National Aeronautics and Space Administration (NASA), the European Space Agency (ESA) or the Japan Aerospace Exploration Agency (JAXA), have been conducting microgravity experiments. Gravity is the dominant force on Earth, from the way materials and substances interact to the way life has developed. But when it comes to space, aboard a spacecraft orbiting Earth, the most intuitive phenomena behave in an unintuitive way. For example: convection, buoyancy, hydrostatic pressure and sedimentation processes are significantly altered. Therefore, scientific disciplines like fluid physics, combustion or crystal growth are hugely affected by the absence of gravity.

In order to carry out these experiments, some test platforms that replicate a free fall situation have been built on Earth and its surroundings. Each one of them has its positive and negative aspects, but the most important feature is the available free fall time (bigger free fall times allow to perform longer experiments):

- Drop towers and drop shafts: capsule experiments are dropped from the top of a tower, achieving microgravity conditions during the fall. One of the most important drop towers of the world is the ZARM, funded by ESA and located at the Universität Bremen. This tower is 146 m high, ensuring 4.74 s of free fall. In 2004, a catapult system was installed on the surface of the tower, doubling the trajectory length and thus extending the period spent in microgravity to 9.3 s (duration that no other drop facility can provide) [6].

- Parabolic flights: weightlessness is achieved by performing an aerobatic maneuver called parabola. These flights are carried on board modified versions of commercial aircraft (A300 Zero-G and B727 G-Force One are the most used). Initially, the aircraft climbs with a pitch angle of 45º, up to a point where thrust is reduced and the nose is lowered to maintain a "zero lift" configuration. From this point on, a ballistic trajectory is followed, letting the plane fly "by itself" and setting the thrust to exactly compensate drag. By doing so, almost 25 s of free fall are experienced [7]. The main drawback of this method is that the g-forces are of the order of 10^{-2} g, which are not perfect microgravity conditions. Fig. 1.2 shows the trajectory of the parabolic flight.

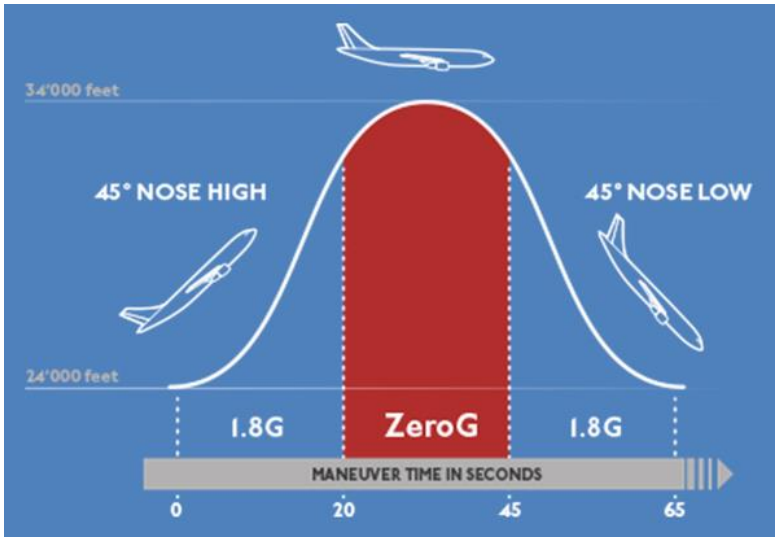

Fig. 1.2 A typical parabolic flight trajectory [8]

- Sounding rockets: a rocket is launched up to 1200 km in height [9]. When all the fuel is burned, the payload (which carries the experiment) is separated from the motor and follows a free trajectory, falling back to Earth. During this time, the experiment is conducted in microgravity conditions. After the payload re-enters the atmosphere, it is brought down to the surface by using a parachute and then is retrieved. Up to 15 minutes of free fall can be achieved using this type of rockets.

- Orbiting spacecraft: all the facilities previously mentioned share a common problem: the weightlessness phenomenon only lasts a few seconds or minutes. Longer experiments in microgravity must be placed inside an orbiting spacecraft, which is in a state of free fall around Earth, providing a high-quality microgravity environment. An example of an orbiting spacecraft is the ISS, which has been orbiting Earth for 20 years.

To summarize, the different facilities explained in this chapter are shown in Fig. 1.3, comparing the quality of the microgravity environment with the time that they can last in weightlessness conditions.

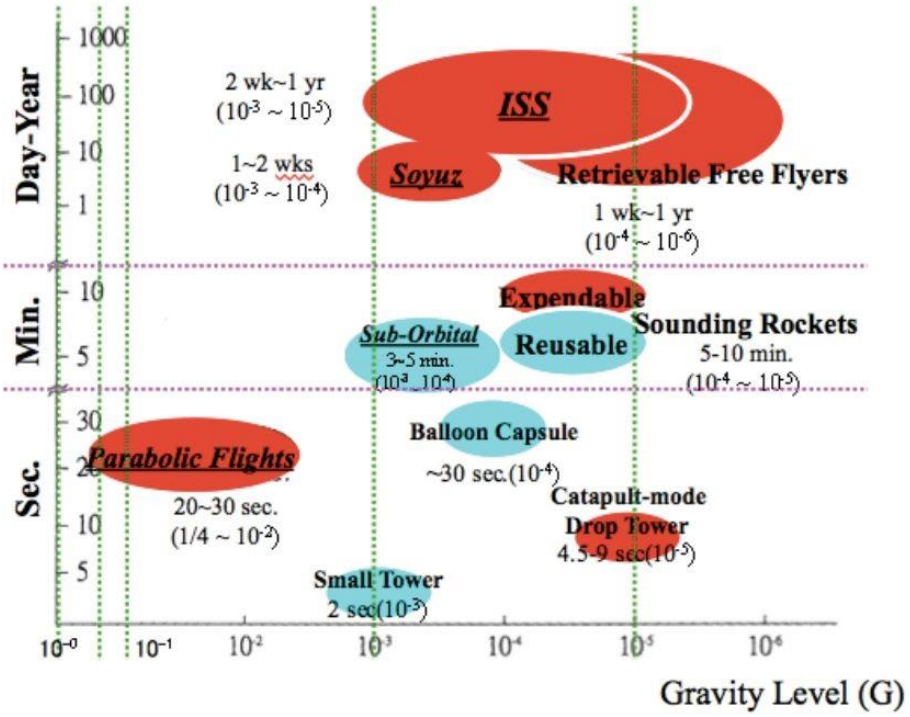

Fig. 1.3 Microgravity facilities comparison chart [10]

1.2. Microfluidics

The study of two-phase flows has gained importance over the last decades due to its presence in multiple applications in space systems. For example: in power and thermal management, liquid oxygen and hydrogen are vaporized and mixed in order to provide electricity. In fluid management, liquids are transferred from one location to another through pipes. In Environment Control and Life Support Systems (ECLSS), waste water bioreactors consume non-potable water containing soap, urine and other wastes and transform it into potable water [11]. These are just three examples of two-phase flows applications in space missions, but there are many more and new ones will come in the future.

In order to test new technologies on Earth, one could use the experimental platforms explained in subchapter 1.1 (like drop towers, for example), but they can be expensive and difficult to build.

Microfluidics is the science that studies the behavior of fluids trough micro-channels or capillaries, dealing with tiny volumes of fluid (down to femtoliters). At this scale fluids behave in a very different manner. In particular, there is a non-

dimensional number called the Bond number (which is further explained in Chapter 2) that relates the gravitational and surface tension forces. In microfluidics, the Bond number is always so small that gravitational forces can be neglected, thus achieving micro-g conditions without the need of travelling to space or using microgravity facilities on Earth. Although microfluidics is useful to replicate microgravity conditions, its main applications are not related to fluids in space, but to food industry, biology and medicine (among others):

- Microfluidic fuel cells: the channels of these cells are in the order of micrometers and, at this tiny scale, flow is completely laminar and no turbulence occurs (Reynolds number is usually smaller than 100) [12]. Using that principle, efficient alkaline fuel cells can be manufactured without a solid membrane separating oxidant and fuel. These fuel cells provide cheap and efficient power for small electronic devices.

- Laboratory-on-a-chip: integrates the functionalities of a full laboratory within one microfluidic chip. Several experiments can be done using these chips, but they are usually used to control the surrounding environment (pH control, for example). An example of the different devices that can be integrated in this kind of chips is shown in Fig. 1.4.

- Cell analysis: in cell cytometry, an accurate and precise control of flow is required. Microfluidic flow controllers are used in order to regulate pressure, volume and vacuum.

- Blood flow replication: the rise of microfluidic technologies has allowed to replicate vascular systems' properties and has helped to detect pathological changes to red and white blood cells. Moreover, some complex blood diseases such as thrombosis have been successfully reproduced [13].

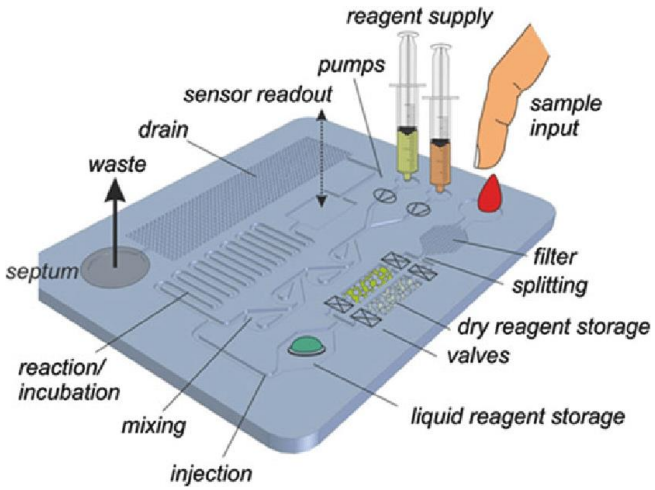

Fig. 1.4 Lab-on-a-chip devices [14]

1.3. Aim of the project

As said before, two-phase flows are present in multiple space subsystems. Different methods have been studied to control the dynamics of these flows in space, but the problem is still open and extensive research can be done. The method used in this project to study two-phase flows is a T-junction bubble generator. A T-junction is the 90 degrees intersection of two different pipes or capillaries (if the pipes are very small, as it is in our case). Tiny scales allow us to work on the microfluidics field and achieve similar conditions to the spatial ones, since gravity has a minor importance when comparing it to capillary effects.

The T-junction configuration can be seen in Fig. 1.5. Air is injected through the main (horizontal) capillary and water through the vertical one. The interaction between both fluids will generate bubbles that will develop as they travel through the main channel and, finally, they will exit the geometry at the outlet.

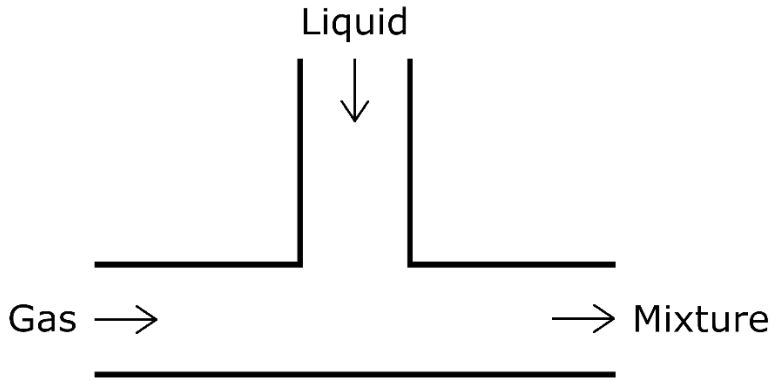

Fig. 1.5 Sketch of the T-junction case

This same problem was also studied in [1], where physical experiments were conducted comparing the injection method shown in the figure above with the inverse one (injecting air through the side channel and water through the main channel). The inverse configuration has been deeply studied before; see [2, 3, 15]. However, in this study no laboratory experiments will be performed. Instead, numerical simulations will be run using OpenFOAM, a reliable, powerful and open source Computational Fluid Dynamics (CFD) software. This program has been chosen mainly because no license is needed in order to use it and it offers a similar performance compared to other software such as ANSYS or STAR-CCM+. The simulated bubble generation process will be compared with experimental results, expecting to get a similar bubble behavior.

One key phenomenon to study in order to control the bubble formation in a T-junction is the squeezing to dripping transition. The gas breakup mechanism is strongly affected by the capillary number (explained in the following chapter), leading to these two different regimes of formation of bubbles. One of the objectives of this project is to extract conclusions about them.

The organization of this project is the following:

1. An introductory chapter to explain the different dimensionless numbers that play an important role in this study, e.g., capillary number, Bond number, Courant number, etc.

2. A chapter describing the methodology. Introduction to CFD and, particularly, OpenFOAM. Explanation of the geometry, the mesh and the boundary conditions. Calculation of the different parameters of study.

3. A validation chapter where preliminary tests are done in order to correctly set up the different input parameters of the simulations, e.g., microchannel dimensions, contact angle value, measuring surfaces location, etc.

4. Discussion of the results for the different cases. Analysis of average air fraction and bubble frequency, volume, velocity and length. Comparison with experimental results. Discussion of squeezing to dripping transition.

5. Derive conclusions and propose future work.

CHAPTER 2. THEORETICAL FRAMEWORK

In this chapter, the characteristics of both fluids are listed and an explanation of dimensionless numbers and other important parameters is done.

2.1. Fluids characteristics

During the whole project the sub-index G is used for the gaseous phase (air) and the sub-index L for the liquid phase (water). Both fluids are considered incompressible and in isothermal conditions at a room temperature of 25 ºC, as in [1, 2, 3, 15]. According to this, standard values at 25 ºC are used for dynamic viscosity ($\mu_G = 10^{-5}\ Pa \cdot s$ and $\mu_L = 10^{-3}\ Pa \cdot s$), density ($\rho_G = 1.225\ kg/m^3$ and $\rho_L = 10^3\ kg/m^3$) and gas-liquid surface tension ($\sigma = 0.072\ N/m$). OpenFOAM uses kinematic viscosity instead of dynamic viscosity, so the value of dynamic viscosity must be divided by density ($\nu_G = 8.163 \cdot 10^{-6}\ m^2/s$ and $\nu_L = 10^{-6}\ m^2/s$).

2.2. Dimensionless numbers

The main control parameter of this study is the velocity of injection of both fluids. The gas and liquid superficial velocities are defined in Eq. (2.1) and Eq. (2.2), respectively.

$$U_{SG} = \frac{Q_G}{A} \tag{2.1}$$

$$U_{SL} = \frac{Q_L}{A} \tag{2.2}$$

Where Q_G and Q_L are the gas and liquid volumetric flow rates and A is the cross-section area of the capillary.

The first important dimensionless number is the Bond number (also called Eötvös number), which represents the balance between gravitational and surface tension forces and it is useful to characterize the shape of the bubbles moving inside the liquid. It is defined in Eq. (2.3).

$$Bo = \frac{(\rho_L - \rho_G) g \phi_c^2}{\sigma} \quad \textbf{(2.3)}$$

Where g is the acceleration of gravity and ϕ_c is the diameter of the capillary. As it can be seen, the Bond number does not depend on the injection velocity of the fluids.

In this project, $Bo = 0.139$, so it can be affirmed that surface tension forces play a much bigger role than the gravitational ones. Moreover, Suo and Griffith [16] stated that gravitational effects become negligible with respect to capillary effects when $Bo < 0.29$. Therefore, gravity can be neglected during this whole study thanks to the small diameter of the channels.

Another important dimensionless number is the Weber number, often used when analyzing interfaces between two fluids. It measures the relative importance of the inertia of a fluid compared to its surface tension. The gas Weber number is defined in Eq. (2.4).

$$We_G = \frac{\rho_G \phi_c U_G^2}{\sigma} \quad \textbf{(2.4)}$$

Where U_G is the gas velocity, commonly referred as bubble velocity. The bubble velocity can be calculated following Eq. (2.5).

$$U_G = C_0 U_M = C_0 (U_{SG} + U_{SL}) \quad \textbf{(2.5)}$$

Where C_0 is the void fraction distribution coefficient, that can be obtained by using a linear fitting over the data of the bubble velocity as a function of the mixture superficial velocity U_M.

Rezkallah [17] stated that, when $We_G < 2$, capillary forces overcome inertial forces and the bubble formation process is governed just by the surface tension. As it can be seen in Table 2.1, the maximum value for We_G is way smaller than 2, so the surface tension forces will be dominant during the simulations.

The dimensionless number that indicates whether the flow is laminar or turbulent is the Reynolds number. It is defined as the relation between inertial forces and viscous forces. The average mixture superficial velocity Reynolds can be seen in Eq. (2.6).

$$Re_M = \frac{\rho_L \phi_c U_M}{\mu_L} \tag{2.6}$$

This Reynolds number is always below 2300 in this study, which is the transition from laminar to turbulent for flows in a pipe [18]; see Table 2.1. Therefore, we can ensure that we are working under laminar flow conditions (as expected in microfluidics).

Finally, the capillary number compares the viscous drag forces to the surface tension forces acting across the interface of both fluids. The liquid superficial velocity capillary number is defined in Eq. (2.7).

$$Ca = \frac{We_{SL}}{Re_{SL}} = \frac{\mu_L U_{SL}}{\sigma} \tag{2.7}$$

Note that Ca only depends on the liquid superficial velocity, since both the viscosity of the liquid and the gas-liquid surface tension have a constant value.

In summary, the values shown in Table 2.1 indicate that gravity has a negligible effect in the bubble generation process with respect to capillary effects ($Bo < 0.29$), that viscous forces are more important than inertial forces ($Re_M < 2300$, laminar flow) and that capillary forces overcome inertial forces ($We_G < 2$), so the bubble generation process is driven by the competition between viscous and interfacial forces. Therefore, the capillary number is the most important dimensionless number of this study, since it compares the relative effect of those two mentioned forces. For low capillary numbers (as it is in this case), the flow is dominated by capillary forces, whereas for high capillary numbers the capillary forces are negligible in comparison to viscous forces.

	U_{SG} $[m/s]$	U_{SL} $[m/s]$	Bo	We_G	Re_M	Ca $[\mathrm{x10^{-3}}]$
Min. value	0.2	0.2	0.136	0.003	400	2.78
Max. value	0.6	0.6	0.136	0.034	1200	8.33

Table 2.1 Dimensionless numbers range of values

In relation with CFD, there is another dimensionless number worth to discuss: the Courant number Co. It is defined as the quotient between the time step of a simulation and the residence time of the fluid inside a finite volume (cell volume); see Eq. (2.8).

$$Co = \frac{\Delta t}{\Delta t_{cell}} = \frac{\Delta t}{\Delta x / U} = \frac{U \Delta t}{\Delta x} \quad \textbf{(2.8)}$$

Where Δt is the simulation time step, Δx is the characteristic length of the cell and U is the velocity of the fluid at the given cell.

If $Co > 1$ it means that $\Delta t > \Delta t_{cell}$, that is, a fluid particle moves through two or more cells at each time step, affecting the convergence of the solution since the momentum equations cannot be solved for every cell. However, some robust systems and fine solvers like OpenFOAM can deal with bigger Courant numbers (way into hundreds before diverging), since implicit solvers are less sensitive to numerical instability [19].

The ideal scenario would be to have a $Co < 1$ during the whole simulation. If the mesh is stationary (fixed cell dimensions at every time) and the inlet velocities are also fixed for every simulation, then the only parameter that can be adjusted is the time step. Isolating Δt from Eq. (2.8) we can have a good estimation of how small the time step has to be in order to keep the Courant number below a desired threshold. Nonetheless, this calculation is not always accurate since we do not know the velocity of the fluid at every cell, so the best practice is to run a test simulation with the calculated time step and see if the solution converges or diverges. In this project, the order of magnitude of the time step is μs.

Finally, it is mandatory to discuss the phase fraction α value. This value determines the relative volume fraction of both phases in each computational cell. The liquid phase is defined to have a value of $\alpha = 0$ and the gaseous phase has a value of $\alpha = 1$. The interphase cells can have an alpha ranging from 0 to 1. In order to calculate the mean value of α in a surface, the program does the weighted average shown in Eq. (2.9).

$$\alpha_{avg} = \frac{\sum_i^n A_i \alpha_i}{\sum_i^n A_i} \quad \textbf{(2.9)}$$

Where α_i is the phase fraction value of the cell i and A_i its equivalent surface.

2.3. Contact angle

The contact angle θ is the angle between the surface of a liquid and a solid when they are in physical contact. It is used to quantify the wettability of a solid surface: if the contact angle is smaller than 90º, the solid surface is considered hydrophilic and if the contact angle is bigger than 90º, the solid surface is said to be hydrophobic (see Fig. 2.1).

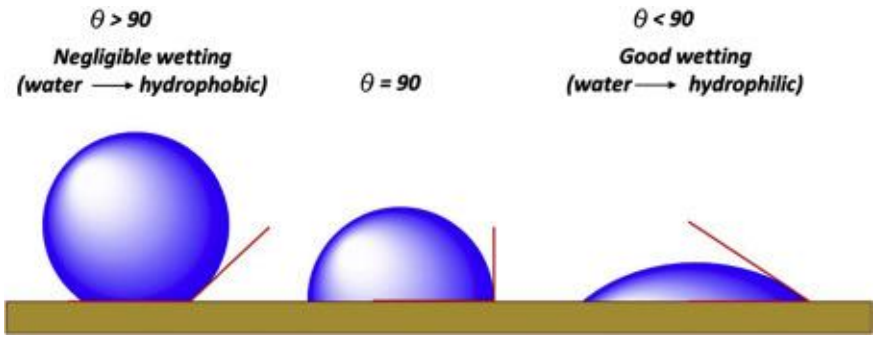

Fig. 2.1 Contact angle examples [20]

As it can be seen, the contact angle is usually defined from the inner part of the bubble. However, in OpenFOAM is defined as the supplementary angle: $180 - \theta$ since it is measured through the gas instead of through the liquid.

To get an idea of the values of the contact angle, highly hydrophobic surfaces such as some polymers exhibit contact angles of around 120º. A natural example of an ultrahydrophobic surface is the lotus leaf, with contact angles up to 150º, as a result of the surface being covered by a waxy material and the irregular structure of the leaf. Thus, water droplets are loosely stuck on the surface and tiny movements of the leaf will cause the water to move easily, taking the dirt particles away [21]. It is a self-cleaning process.

The regular capillary materials are rather hydrophilic than hydrophobic. Therefore, in this project some trials will be done using contact angles smaller than 90º and the value which makes the results closer to reality will be the one chosen to perform the definitive simulations.

CHAPTER 3. METHODOLOGY

In this chapter, the collection of procedures, techniques and tools used in this project are discussed. The three different steps of the simulation process are explained.

3.1. CFD

CFD is the field of fluid mechanics that uses numerical methods and algorithms in order to solve and analyze fluid flow problems. Computers are used to perform millions of calculations required to simulate the interaction of liquids and gases with complex surfaces and structures. Even with simplified equations and the newest high-performance supercomputers, only approximate solutions can be achieved in many cases. This is due to the fact that there is not a general analytic solution to the Navier-Stokes equations, the system of partial differential equations that define the motion of any fluid in space. In fact, some basic properties of these equations have never been proven and we know very little about turbulent fluid motion (which is often present in the solutions of the Navier-Stokes equations). For this reason, the Clay Mathematics Institute made the following problem one of its seven Millennium Prize problems in mathematics [22]:

"Prove or give a counter-example of the following statement:
In three space dimensions and time, given an initial velocity field, there exists a vector velocity and a scalar pressure field, which are both smooth and globally defined, that solve the Navier–Stokes equations."

A one-million-dollar prize is offered to the first person solving this problem. The 2D version was already solved in the 1960s and weak solutions have also been proved. Finding the 3D solution to the Navier-Stokes equations would suppose a huge advance in the CFD field, in fluid mechanics and in mathematics, in general.

Nonetheless, continuous research allows the incorporation of software that increases the calculation speed as well as decreases the error margin of the simulations. At the same time, this research has also led to the analysis of increasingly complex situations such as transonic fluids and turbulent flows (by giving more accurate approximations). The verification of the data obtained by CFD is usually carried out in wind tunnels or other physical scale models. For example, in this project the results obtained via OpenFOAM will be compared with the experiments performed in [1].

Nowadays, CFD is used in almost every branch of science and technology: process industry, construction (i.e. ventilation), health and safety (i.e. virus propagation), motor industry, aerodynamics, electronics, meteorology, biology, etc. It is a good alternative method to experiments that can be highly expensive or difficult to replicate in real life.

The CFD simulation process consists of three primary steps:

- Pre-Processing: describing the geometry and setting up the problem. Generating the mesh. Defining the fluid properties and the initial and boundary conditions.

- Solver: solving the discretized governing equations of the problem choosing the right solver (interFoam in our case).

- Post-Processing: analyzing the results generating contour plots, streamlines, vector plots, data plots, etc. The software that will be used for post-processing is ParaView and MATLAB.

3.1.1 OpenFOAM

The program used to perform the numerical simulations is OpenFOAM, which stands for Open source Field Operation And Manipulation. It is a free open source CFD software that has a large user base from most areas of engineering and science since it can solve complex fluid flows that involve chemical reactions, turbulence, heat transfer, acoustics, solid mechanics, electromagnetics, etc. [23]

The main advantage of OpenFOAM is that users do not have to pay for a license and everyone can contribute to the different libraries since it is an open source program. For the same reason, the code can be customized according to individualized problems of the user. The main drawback is that is not a friendly program for inexperienced users because it has no interface and the simulation parameters have to be changed manually from text files written in C++ language. Some weeks of training are recommended in order to get used to OpenFOAM.

The basic directory structure for an OpenFOAM case is shown in Fig. 3.1.

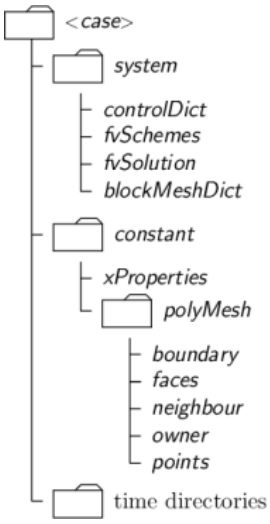

Fig. 3.1 Structure of an OpenFOAM case directory [24]

- The *system* directory contains the files associated with the solution procedure, such as time steps, start/end time, discretization schemes, equation solvers, tolerances, etc.

- The *constant* directory contains the files that fully describe the case mesh (inside *polyMesh*) and the files that specify the physical properties of the problem.

- The time directories contain the results obtained by the program at each writing time. The most important one is the *0* directory, where initial and boundary conditions are imposed.

For more information about OpenFOAM, check the user guide available in [24]. A detailed explanation of an OpenFOAM case is explained in Appendix A.1.

3.2. Pre-Processing

3.2.1 Geometry

The geometry used to perform the simulations is a T-junction, which is the 90 degrees intersection of two different pipelines or capillaries, resulting in one single outlet. The diameter of the capillaries is 1 mm, the same as in [1] to do reliable comparisons, and the length of both inlet and outlet capillaries is 10 mm, as it can be seen in Fig. 3.2. This is a new addition with respect to previous studies such as [25] or [26], where the length of the capillaries was 1 mm for the inlets and 4 mm for the outlet. The elongation of the capillaries has been done in order to stabilize the flux, that is, to ensure that the channel is long enough to allow both fluids to develop properly (this idea will be further discussed in Chapter 4.1).

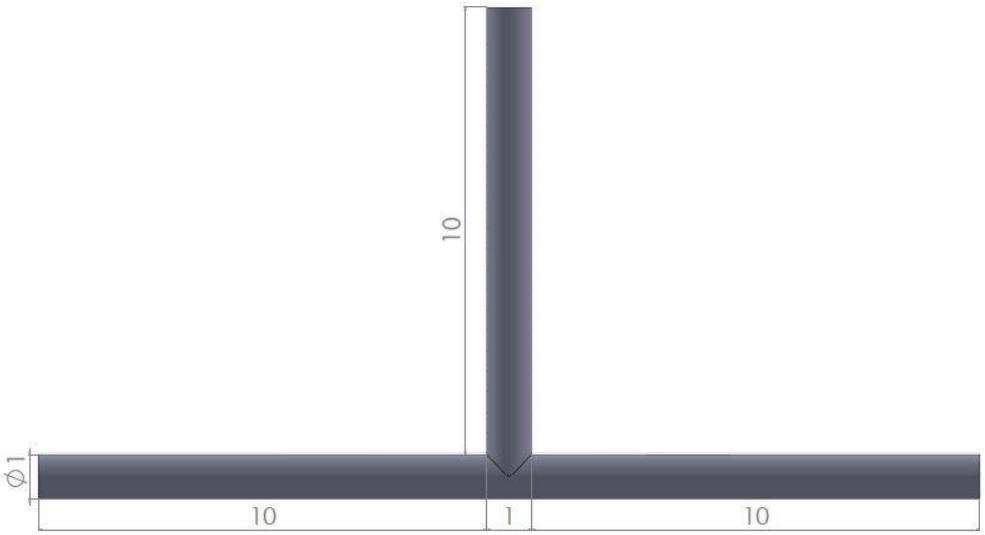

Fig. 3.2 T-junction geometry (dimensions in mm)

When creating the geometry, the most important step is to define the boundaries (if not, OpenFOAM will not be able to set the boundary conditions). They can be seen in Fig. 3.3.

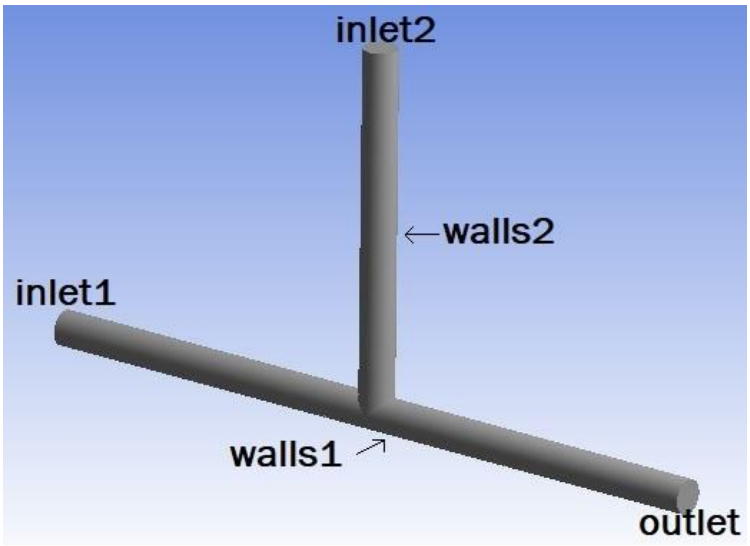

Fig. 3.3 Geometry boundaries

Where *inlet1* corresponds to the gas inlet, *inlet2* to the liquid inlet, *outlet* to the exit of gas-liquid mixture, *walls1* to the horizontal capillary walls and *walls2* to the vertical capillary walls.

The geometry and mesh generation process can be really challenging by just using OpenFOAM's native tools (like BlockMesh or SnappyHexMesh) because the learning curve is highly steep for new users, so it is recommended to use external software. In my case, the program used to generate both the geometry and the mesh is ANSYS Student 2019 R3. This software has been chosen for its simplicity. It is intuitive to use, includes the DesignModeler tool to create the geometry without any problem (it is similar to some other CAD programs like SolidWorks) and it is easy to export the mesh as a *.msh* file to later use it in OpenFOAM.

3.2.2 Mesh

The generated mesh has a total of 683 565 cells and it can be observed in Fig. 3.4. It is important to stablish a compromise between mesh quality and computational time. The ideal case would be to design a mesh with millions of cells and dispose of huge computational power in order to perform fast and accurate simulations. However, CFD simulations are time and resource consuming, so it is not feasible to create a ten million cells mesh. Moreover, a mesh with tiny cells implies big values of the Courant number (see Eq. (2.8)),

leading to a possible non-convergent solution. To give an example, the slowest simulation of this project ran for more than 7 days on the EETAC cluster.

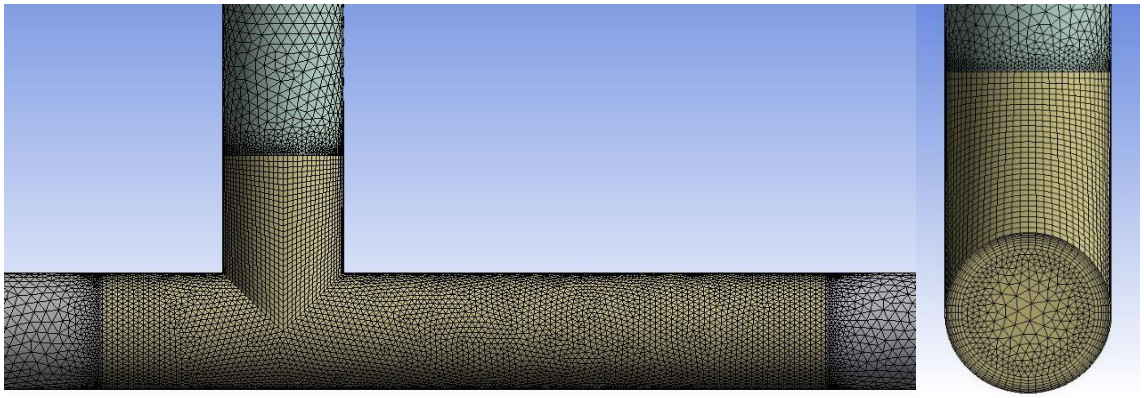

Fig. 3.4 Close-up look of the mesh (frontal and lateral view)

As it can be seen, the mesh has been split into two separate parts. The first one corresponds to the T-junction itself, which has a higher density of cells in order to capture as good as possible the bubble generation phenomenon. The second one corresponds to the entrance and exit capillaries, which can have a coarser mesh since no critical process happens in this part of the geometry.

If this thickening process was not done, that is, if the entire geometry consisted of the same fine mesh, the total number of cells would be 1 351 163 and the computational time of every simulation would increase a 97.7 % (assuming a linear relationship between the number of cells and the clock time). That is, the slowest simulation that took 7 days to finish with the mixed mesh would take approximately 14 days to finish with a regular mesh.

To have a better understanding of the contact angle parameter (adhesion to the walls), an inflation mesh is done near the walls. That means that the mesh is finer close to the walls than close to the capillary centerline.

The main setup parameters introduced in ANSYS are the following:

- Element size: 0.05 mm on the fine part and 0.1 mm on the coarse part.
- Inflation mesh: 8 layers, growth rate equal to 1.2 and maximum thickness equal to 0.1 mm.
- Multizone: *Hexa* mapped mesh type and *Tetra* free mesh type.

The mesh was exported as a *.msh* file and was converted to OpenFOAM using the *fluent3DMeshToFoam* command, which automatically creates the *polyMesh* directory inside *constant* (see Appendix A.1 for further explanation). This directory contains all the files related with the mesh.

3.2.3 Initial conditions

The initial conditions are defined inside *0* folder of the directory case. There are three files: one corresponding to the velocity, another to the pressure and the final one to the air fraction. These files can be found in Appendix A.1.1.

- Velocity file: the internal field value states that the fluid has zero velocity at initial time.
- Pressure file: the internal field is set to zero (that is, gauge pressure equal to zero at initial time). Atmospheric pressure conditions.
- Alpha file: defines $\alpha = 1$ as the gaseous phase and $\alpha = 0$ as the liquid phase. The internal field is set to $\alpha = 0$, that is, everything is filled of water at initial time.

However, the file *setFields* inside *system* directory fills the main channel with air up to $x = 10\,mm$. This is done because the process of filling the main capillary with air is quite simple and it is not necessary to simulate it. In terms of time and resource consumption it is better to use *setFields*. To have a quantitative idea, for the $U_{SG} = U_{SL} = 0.5\,m/s$ case, the gas spends 36 simulation hours filling the main channel until reaching the T-Junction. These one and a half days can be saved by using *setFields*. The initial alpha distribution can be observed in Fig. 3.5 (air in red, water in blue).

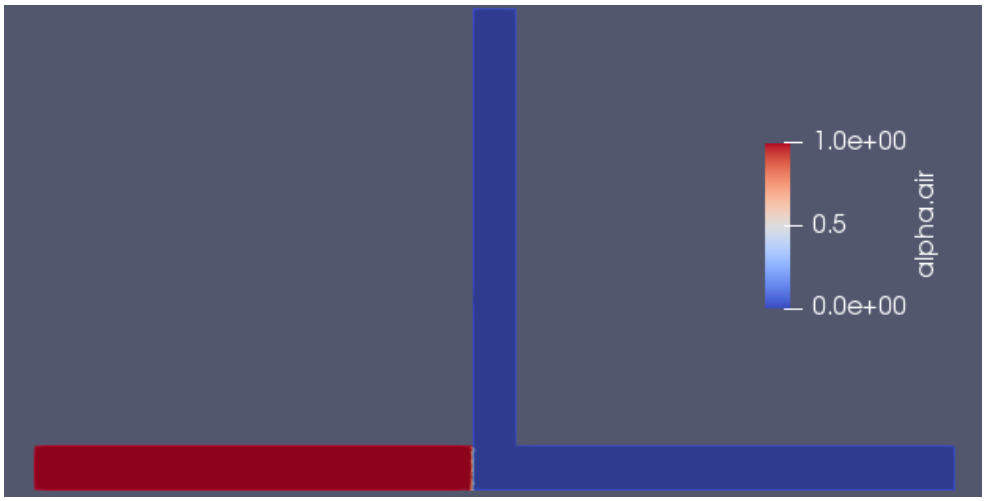

Fig. 3.5 Initial distribution of fluids

3.2.4 Boundary conditions

The boundary conditions used in this project are the same as the ones used by Aboukhedr et al. in [15], where the bubble formation in a T-Junction is simulated

using OpenFOAM. The full list of possible boundary conditions that OpenFOAM can offer can be seen in [27].

Since the geometry has a total of 5 boundaries and there are 3 different fields (velocity, pressure and air fraction), the total number of boundary conditions is 15. In Table 3.1, Table 3.2 and Table 3.3 the boundary conditions are listed and explained.

Boundary	Condition	Value	Meaning
inlet1	fixedValue	$U_{SG} = 0.5\ m/s$	Injection of air at a constant speed of 0.5 m/s in the x-axis direction.
inlet2	fixedValue	$U_{SL} = 0.5\ m/s$	Injection of water at a constant speed of 0.5 m/s in the y-axis direction.
outlet	pressureInlet OutletVelocity	$U = 0\ m/s$	Flow out of the domain: assigns a zeroGradient condition. Flow into the domain: assigns a velocity based on the flux in the patch-normal direction.
walls1 and walls2	fixedValue	$U = 0\ m/s$	The fluid at the walls does not move. The same as noSlip BC.

Table 3.1 Velocity file boundary conditions

Boundary	Condition	Value	Meaning
inlet1 and inlet2	zeroGradient	-	Pressure gradient is 0 at the inlets, that is, it is assumed to be constant.
outlet	fixedValue	uniform 0	At the outlet the gauge pressure is equal to 0 (atmospheric pressure conditions).
walls1 and walls2	fixedFluxPressure	uniform 0	Adjusts the pressure gradient such that the flux on the boundary is specified by the velocity boundary condition.

Table 3.2 Pressure file boundary conditions

Boundary	Condition	Value	Meaning
inlet1	inletOutlet	uniform 1	The same as zeroGradient, but it switches to fixedValue if there is some backward flow. Air enters through this channel.
inlet2	inletOutlet	uniform 0	The same as zeroGradient, but it switches to fixedValue if there is some backward flow. Water enters through this channel.
outlet	zeroGradient	-	Alpha gradient is 0 at the outlet.
walls1	constantAlpha ContactAngle	theta0 155	Sets a contact angle of 25 degrees (180 – 155) at the walls of the main channel.
walls2	fixedValue	uniform 0	The walls of the vertical capillary are always covered by water.

Table 3.3 Alpha file boundary conditions

3.3. Solver

3.3.1 InterFoam

Since our problem deals with two different phase fluids, a multiphase solver must be used. The chosen solver is *InterFoam*, which is a multiphase, transient, incompressible, immiscible and isothermal VOF method solver. VOF stands for *Volume Of Fluid*, a free-surface modelling technique (numerical method that tracks and locates the fluid-fluid interface). The characteristics of the solver are in accordance with the considerations and assumptions of our problem, so it is suitable to use it. The detailed operation of this solver can be found on the OpenFOAM Wiki [28], but in this subchapter a brief summary will be done.

InterFoam solves the Navier-Stokes equations for the two fluids. The material properties are constant everywhere except in the interphase, that is, water behaves as water when is separated from air, which also behaves as air. In the interphase, according to how much volume each phase occupies, a weighted average is done in order to compute the physical properties. The constant-density continuity equation and the momentum equation are shown in Eq. (3.1) and Eq. (3.2), respectively.

$$\frac{\partial u_j}{\partial x_j} = 0 \qquad \textbf{(3.1)}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\tau_{ij} + \tau_{t_{ij}}\right) + f_{\sigma i} \quad \textbf{(3.2)}$$

Where u represents the velocity, ρ the density, p the pressure, τ and τ_t the viscous and turbulent stresses and $f_{\sigma i}$ the surface tension. The density is defined as a function of the air fraction value:

$$\rho = \alpha\rho_G + (1-\alpha)\rho_L \quad \textbf{(3.3)}$$

α is 1 inside the gaseous fluid and 0 inside the liquid fluid. At the interphase between the two fluids α varies between 0 and 1. The surface tension force is calculated as:

$$f_{\sigma i} = \sigma\kappa\frac{\partial\alpha}{\partial x_i} \quad \textbf{(3.4)}$$

Where κ is the curvature of the free surface, that can be approximated as:

$$\kappa = -\frac{\partial}{\partial x_i}\left(\frac{\partial\alpha/\partial x_i}{|\partial\alpha/\partial x_i|}\right) \quad \textbf{(3.5)}$$

3.3.2 Number of cores trial

To minimize the simulations computational time, they were executed in parallel by splitting the cluster in different cores or sub-processors. The *scotch* decomposition method was used (see *decomposeParDict* file in Apendix A.1.3) and it was tried in a short test simulation (it took less than half an hour for the slowest one). Depending on the number of cores in which the processor is divided, the simulation will take shorter or longer. The results of the clock time as a function of the number of cores is shown in Table 3.4.

Number of cores	Clock time [s]
1	2092
2	1331
4	1243
8	1279
16	1449

Table 3.4 Simulation time depending on the number of cores

As it can be seen, the most efficient option is to run the simulation in parallel using a total of 4 sub-processors. This method offers a 68% improvement on computational time with respect to the single-core simulation.

3.4. Post-Processing

Once the simulations are finished, it is mandatory to post-process the data in order to correctly visualize the results, obtain useful information and extract conclusions.

The main software that is used to post-process is ParaView, an open-source application integrated with OpenFOAM that allows to visualize the results. In this project it is used to extract plots over lines (for example, the velocity of the fluid in a given direction, velocity profiles, pressure plots, etc.), visualize the interior of the geometry and generate animations. A more extensive explanation of ParaView can be found in Appendix A.3.

Apart from Paraview, MATLAB is also used to calculate different parameters such as bubble frequency, velocity, length or volume and obtain graphics presenting the information. The full code is shown in Appendix C.

In order to calculate these parameters using MATLAB, first it is needed to define some sampling surfaces and probes in OpenFOAM. This can be done inserting functions in the *controlDict* file inside *System* directory (see Appendix A.1.3). There is a total of two sampling surfaces (located at x = 19 mm and x = 20 mm) and one point probe (located at x = 10.5 mm, y = 0, z = 0); see Fig. 3.6. The surfaces are located far away from the T-junction to give time to the bubbles to stabilize and the probe is located just at the middle of the T-junction to have a better understanding of the forces behind the bubble generation process.

Fig. 3.6 Location of the probe and the sampling surfaces

From the pressure probe we obtain a file showing the gauge pressure at the *PROBE* point for every time step. From each surface we obtain a file showing the average fraction of air for every time step. In Fig. 3.7 the α_{avg} as a function of time for $U_{SL} = U_{SG} = 0.4\, m/s$ at both surfaces can be observed.

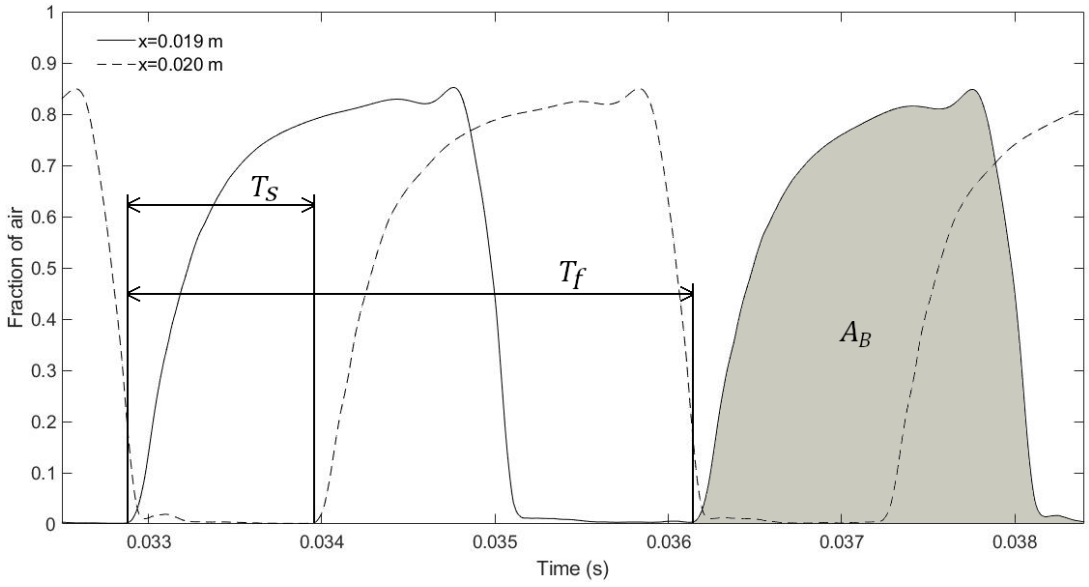

Fig. 3.7 Fraction of air versus time for $U_{SL} = U_{SG} = 0.4\, m/s$

Where T_f is the time between two consecutive bubbles at the same cross section (bubble period) and T_s is the time it takes for the bubble to cross from the first section to the second one. A_B is the area under the alpha curve for a given bubble.

3.4.1 Bubble frequency

The frequency of the bubble can be easily calculated as the inverse of its period, as it can be seen in Eq. (3.6). Note that only one sampling surface is needed in order to obtain the frequency.

$$f = \frac{1}{T_f} \tag{3.6}$$

3.4.2 Bubble velocity

The velocity of the bubble can be calculated dividing the distance between the two surfaces ($d = 1\ mm$) by the bubble crossing time; see Eq. (3.7).

$$U_G = \frac{d}{T_s} \tag{3.7}$$

3.4.3 Bubble length

The length of the bubble can be calculated by multiplying its velocity by the time it takes the bubble to enter and exit the same sampling surface.

3.4.4 Bubble volume

The volume of the bubble can be calculated in two different ways. The first one is to integrate the fraction of air between the surface entering and exiting time and then multiplying it by its velocity and the channel area; see Eq. (3.8) and Eq. (3.9).

$$A_B = \int_{T_0}^{T_1} \alpha(t) dt \tag{3.8}$$

$$V_B = A_B U_G A \tag{3.9}$$

The second strategy is to use the principle of mass conservation. At any section of the capillary the gas volumetric rate (Q_G) is conserved, so the volume can be calculated as shown in Eq. (3.10).

$$Q_G = U_{SG}A = fV_B \longrightarrow V_B = \frac{U_{SG}A}{f} \quad \textbf{(3.10)}$$

3.4.5 Standard deviation

In order to study the dispersion of the different parameters of the bubbles, the standard deviation of the sample can be calculated as shown in Eq. (3.11).

$$\sigma = s_{n-1} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad \textbf{(3.11)}$$

Where n is the number of bubbles of the sample, x_i the parameter value for the i^{th} bubble and $\bar{x}$ the mean value of the sample.

CHAPTER 4. VALIDATIONS

In this chapter, preliminary tests are done in order to set up the definitive case. These tests are very important if we want to obtain accurate and close to reality results.

4.1. Capillary sizing

As said before, the aim of this project is to simulate the development of two fluids injected through a T-junction along a pipe of a given length. In order to determine the most appropriate length of the capillaries, it is mandatory to discuss the flow of a single fluid along a circular pipe (Hagen-Poiseuille flow). If the length of the channel is too short, the flow will not be fully-developed and the results will not be satisfactory.

In this subchapter, a simulation of water entering with velocity $U_{SL} = 0.5\ m/s$ into a channel of diameter $\phi_c = 1\ mm$ is done.

4.1.1 Entrance length

Consider a fluid (water, for example) entering a circular pipe at a constant velocity. Assuming that the fluid has zero velocity with respect to the wall (no-slip condition), the particles in contact with the surface are completely stationary. As a result of friction, the adjacent layers will also slow down and the velocity of the fluid at the midsection of the channel will increase in order to keep the mass flow rate constant. That is, a velocity gradient appears across the section of pipe.

The flow in a pipe is divided into two different regions [29]:

1. Boundary layer region: the effects of the viscous shearing forces caused by fluid viscosity have importance.

2. Irrotational (core) flow region: frictional effects have no importance and the velocity remains constant along the radial direction.

Due to frictional effects, the thickness of the boundary layer increases until reaching the pipe center and filling the entire channel, as can be seen in Fig. 4.1.

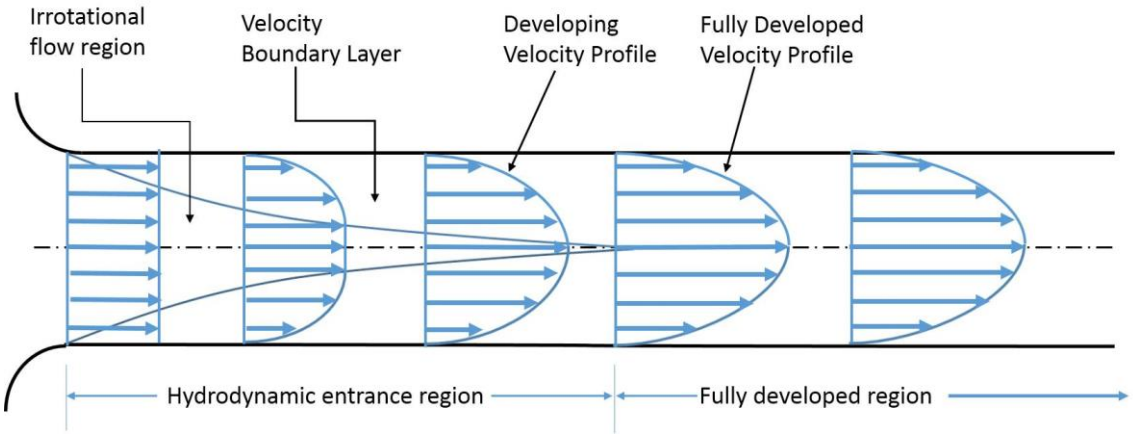

Fig. 4.1 Development of the velocity boundary layer and the velocity profile in a pipe [29]

Thus, the length from the pipe inlet to the point where the boundary layers merge is called the hydrodynamic entrance length. In this region the flow is developing and the velocity profile changes along x-direction. Beyond the entrance region is the fully developed region, where the flow velocity profile remains unchanged along x-axis. Expressing it mathematically:

$$\frac{\partial u(r,x)}{\partial x} = 0 \rightarrow u = u(r) \tag{4.1}$$

As stated in [30], for laminar flow ($Re \leq 2300$), the hydrodynamic entry length can be obtained from an expression of the form:

$$L_{h,lam} \approx 0.05 Re \phi_c \tag{4.2}$$

Note that, the higher the Reynolds number (the more turbulent flow), the longer the entry length must be. The same dependency is observed with the diameter of the channel. In this simulation, $Re = 500$ (laminar flow) and $\phi_c = 1\ mm$, so the approximate entry length that is needed is $25\ mm$.

4.1.2 Analytical solution

In order to compare the results of the simulations with theoretical results, the exact solution of the Navier-Stokes equations for fully developed laminar pipe flow must be obtained. The main calculations and derivations to get the solution are done in Appendix B.1.

The solution states that the maximum velocity that the fluid can achieve (at the capillary centerline) is twice the injection velocity. That is, if we inject water at $0.5\,m/s$ the velocity at the centerline will be $1\,m/s$ when the flow is fully developed. The other interesting result that can be extracted from the solution is that the pressure gradient along the pipe is constant and has a value of $-16\,kPa/m$ in our case.

So, when running the simulations, we expect to get similar results to the ones obtained with the analytical solution.

4.1.3 Simulations

Since this is a validations subchapter where the dimensions of the main channel are discussed, we are not going to go into detail about the mesh nor the simulation parameters.

The generated geometry is a circular pipe of 1 mm diameter and 10 mm length. The mesh has a total of 54 889 cells and it is automatically generated by the program. It can be seen in Fig. 4.2.

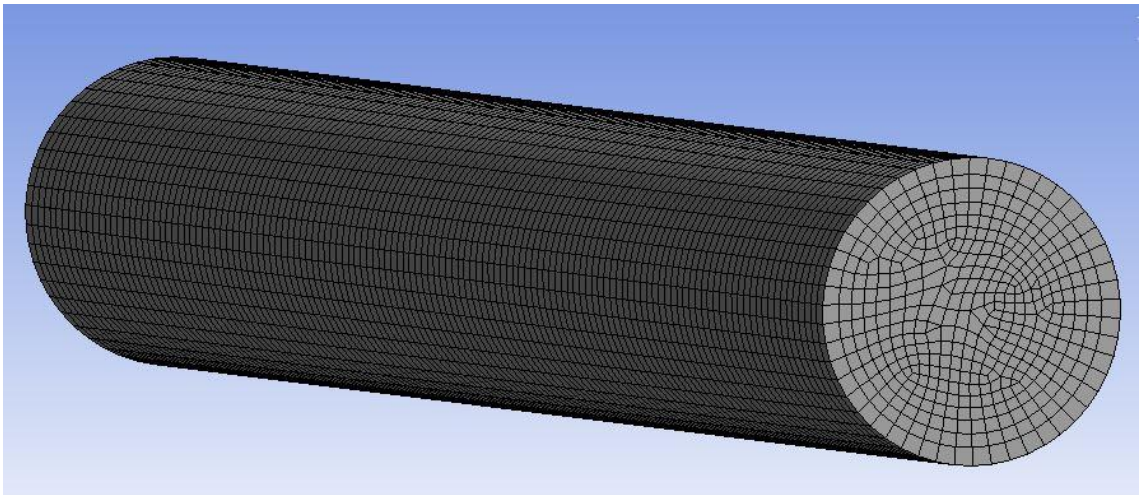

Fig. 4.2 10 mm length pipe mesh

After running the simulation in OpenFOAM and using ParaView, the gauge pressure plot (zero-referenced against ambient air pressure) of Fig. 4.3 is obtained.

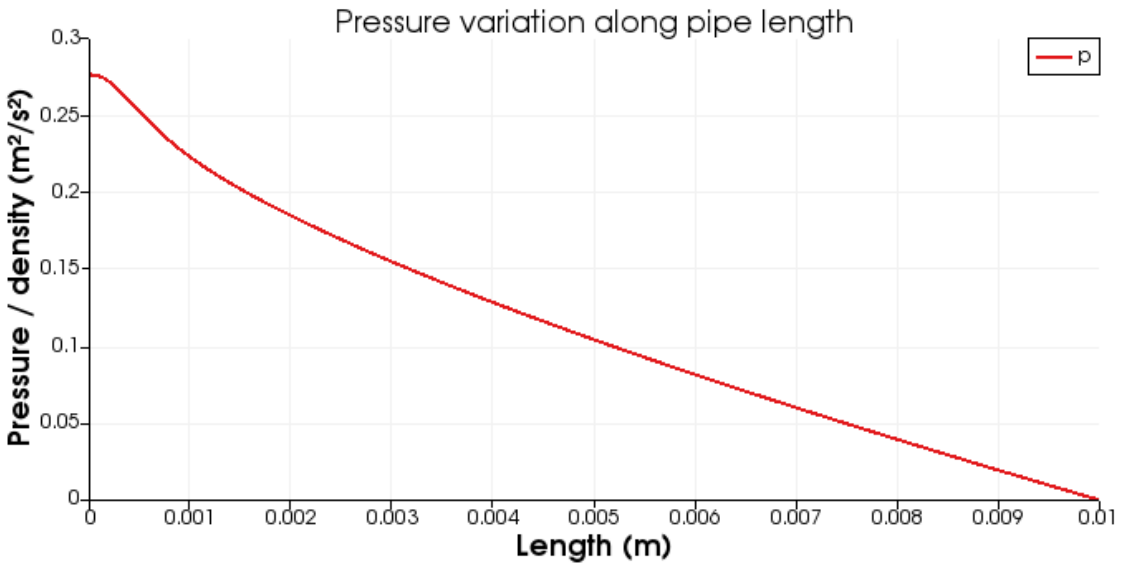

Fig. 4.3 Gauge pressure along pipe length

As can be seen, at the entrance of the pipe, pressure variation is not constant due to the fact that the flow is still developing. At the end of the pipe, the plot is almost linear and the pressure gradient approaches the analytical value shown in Eq. (4.3) (which is derived in Appendix B.1).

$$\frac{dp}{dx} = \frac{-4\mu}{R^2} v_{max} \cong -16\ kPa/m \qquad \textbf{(4.3)}$$

Velocity plots can also be obtained. In Fig. 4.4, the pipe is cut by the centerline along x-axis and the magnitude of the velocity can be observed.

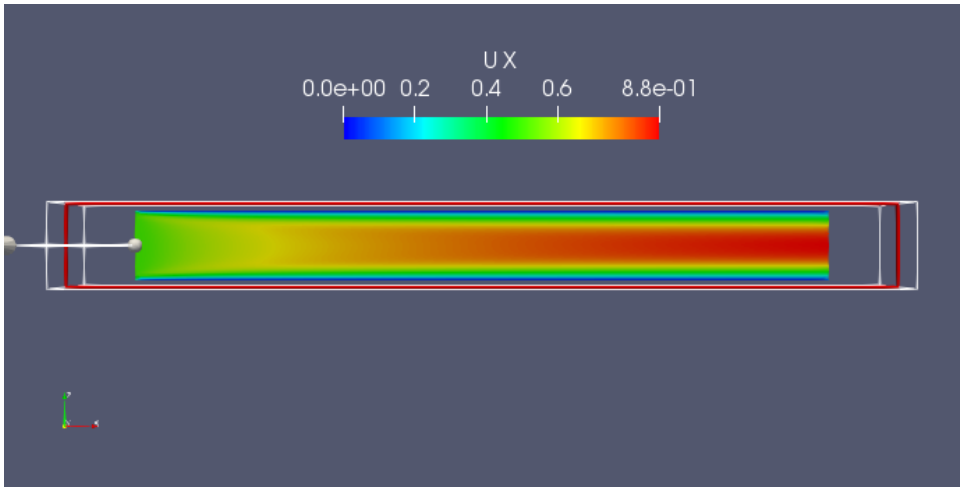

Fig. 4.4 Velocity distribution along the pipe when injecting water at $0.5\ m/s$

Note that the inlet injection velocity is $0.5\,m/s$ and the fluid has zero velocity at the walls, as it should be. Exit velocity is approximately $0.88\,m/s$, which is not exactly $1\,m/s$ due to the reason that the flow is still developing at the end of the channel and a bigger length is needed. Nonetheless, the velocity profile at the outlet is (almost) a paraboloid, as shown in Fig. 4.5, where the velocity profile evolution along the pipe can be observed. Sectional cuts every $1\,mm$ are done. It can be seen how the fluid develops as it advances through the capillary, starting at $x = 0\,mm$ with a $0.5\,m/s$ rectangular velocity profile and ending at $x = 10\,mm$ with a $0.88\,m/s$ peak velocity paraboloid velocity profile. It is the same information presented in Fig. 4.4 but from a different view.

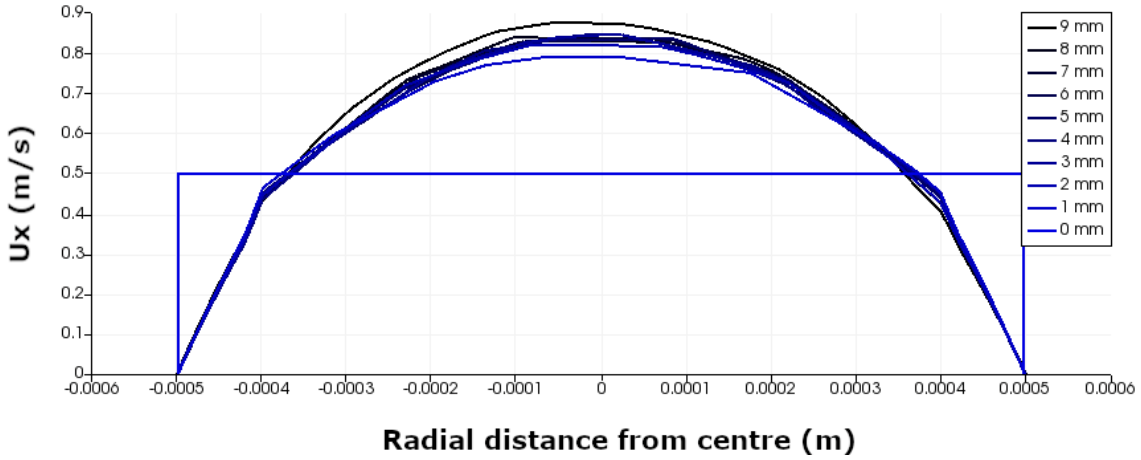

Fig. 4.5 Velocity profile evolution through the capillary

At this point, other simulations were run changing the length of the pipe. In total, five different lengths are tried: $5\,mm$, $10\,mm$, $20\,mm$, $30\,mm$ and $50\,mm$. The results are shown and compared with the analytical solution in Table 4.1.

L $[mm]$	**Cells**	ΔP $[Pa]$	$\Delta P/L$ $[Pa/m]$	v_{max} $[m/s]$	$\varepsilon_{v_{max}}$ [%]
5	27 786	172.5	34 500	0.7749	22.51
10	54 889	276.5	27 650	0.8826	11.74
20	115 018	453.3	22 665	0.9615	3.85
30	163 072	617.8	20 593	0.9781	2.19
50	276 872	939.2	18 784	0.9866	1.34

Table 4.1 Simulation results for different channel lengths

Where L is the length of the channel, ΔP the absolute difference of pressure between the inlet and the outlet and $\Delta P/L$ the pressure gradient along the channel (in absolute value and assuming that $p(x)$ is linear). $\varepsilon_{v_{max}}$ refers to the percentage error of the maximum velocity achieved at the outlet with respect to the velocity that would be obtained if the flow was fully developed ($1\ m/s$, as stated in 4.1.2).

Note that, as the channel length is increased, the simulation results get closer to the analytical solution: $\Delta P/L = 16\ kPa/m$ and $v_{max} = 1\ m/s$. For $L = 20\ mm$, a good result is already obtained and we can assume that the flow is almost fully developed at the exit of the channel (just $3.85\ \%$ error in v_{max}). However, the approximate Eq. (4.2) suggests that a minimum length of $25\ mm$ should be used.

In order to study the influence of the mesh (specifically the number of cells), another simulation is run for $L = 10\ mm$. The new mesh has a total of 742 352 cells and the new value for maximum velocity at the outlet is $v_{max} = 0.8882\ m/s$, that is, just a $0.63\ \%$ improvement with respect to the older mesh of 54 889 cells. However, computational time raises from $20\ s$ to $1044\ s$ (fifty times slower), so increasing the number of cells is not worthy.

The same process is done for every case: generate a fine and a coarse mesh and compare it with the original mesh. The fine mesh has approximately 10 times more cells than the standard mesh and the coarse mesh has 10 times less cells than the standard mesh. The results are plotted in Fig. 4.6.

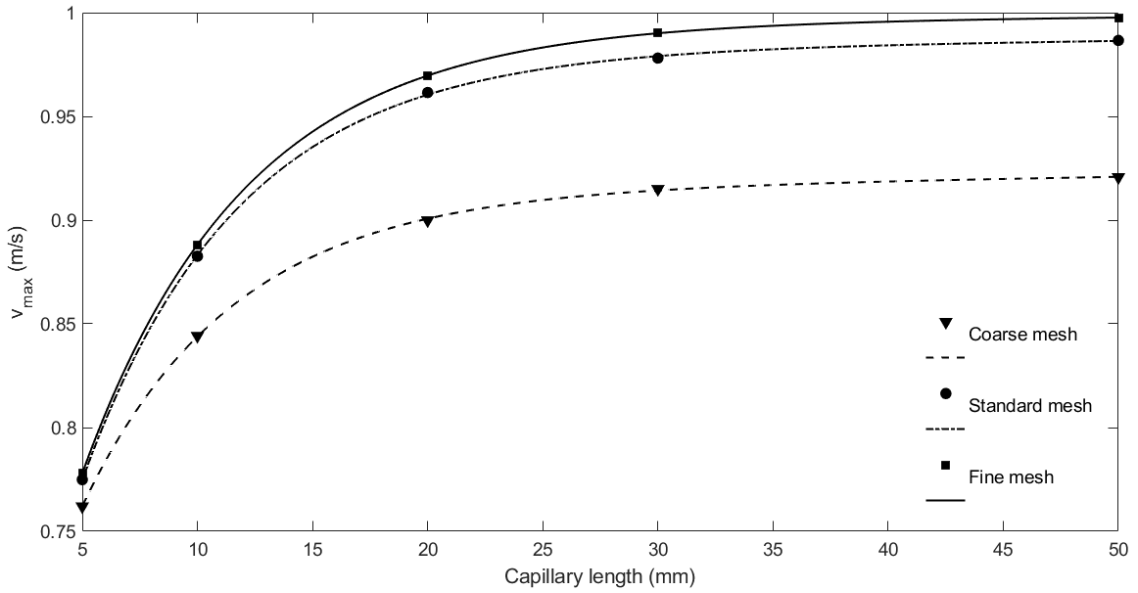

Fig. 4.6 Maximum velocity as a function of capillary length and mesh quality (lines are guide for the eye)

There is not much difference between standard and fine mesh: just a maximum of 1.1% for $L = 50\,mm$. However, the difference is bigger between coarse and standard mesh: a 6.6% for $L = 50\,mm$. In the light of the results, it has been decided that for the final case T-junction, a channel length of $10\,mm$ is chosen for both air inlets and the outlet, using a standard mesh. Longer capillaries would help to get better results, but the increase of mesh cells and computational complexity are not acceptable for this project.

4.2. Contact angle tests

In this subchapter, some test simulations are done in order to understand the contact angle boundary condition and to choose the most adequate value for the definitive cases. Numerical results are very sensitive to the gas-liquid-wall contact angle boundary condition and it can even cause the solution to diverge. For this reason, it is very important to study the influence of this parameter, although it is a challenging and open problem which is still being studied at present; e.g. in [2].

The geometry is a reduced version of the one presented in Chapter 3.2.1 and it corresponds to the finer part of the mesh. This finer segment of the full geometry has been chosen because, as said before, the results are highly sensitive to the contact angle boundary condition and this condition is directly related to the bubble generation process, so it is mandatory to perform the tests with a good quality mesh. Both the water and air inlet are $1\,mm$ long and the outlet is $4\,mm$ long. Two sampling surfaces are placed at $x = 4\,mm$ and $x = 5\,mm$; see Fig. 4.7. The time step is $1\,\mu s$, final time is $0.0256\,s$ and the boundary conditions are the same as in Chapter 3.2.4. Both water and air are injected at $0.5\,m/s$.

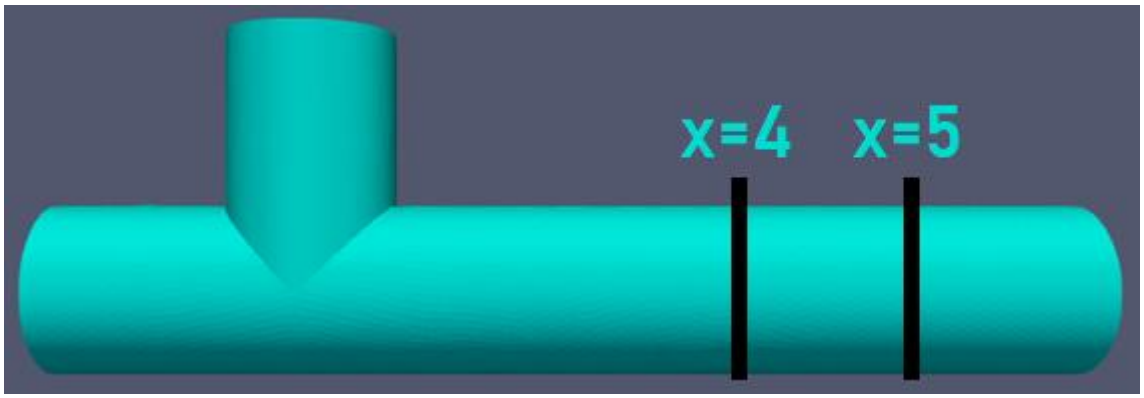

Fig. 4.7 Reduced geometry for the contact angle tests

The tested contact angles are: $0º, 15º, 25º, 45º$ and $90º$. These values are in the range of $0 - 90º$ and sampled around $25º$, since values in the range of $25 - 30º$ are optimal for most of the injection velocities [2]. The comparison of the bubble generation process for the different contact angles can be seen in Fig. 4.8.

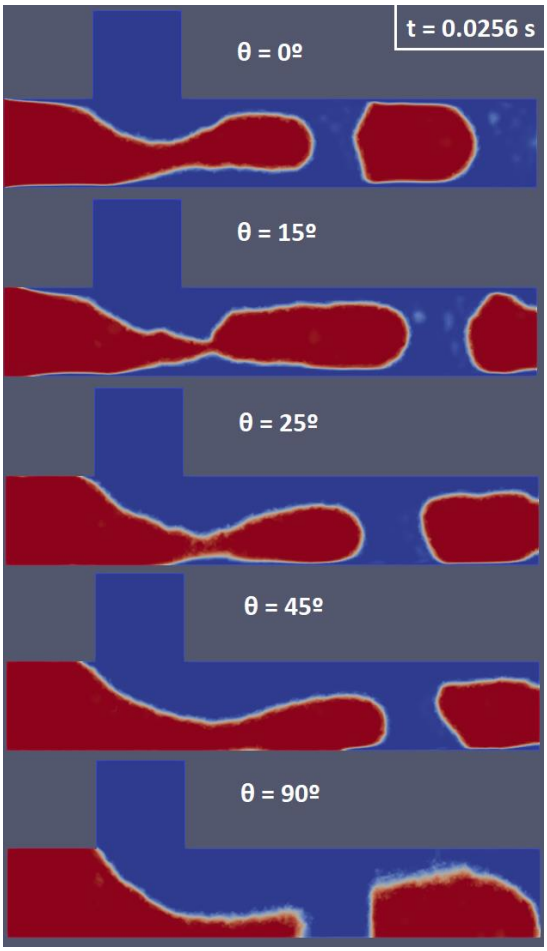

Fig. 4.8 Bubble generation at $t = 0.0256\ s$ for different contact angles

The mean values of frequency, velocity, volume and length for a sample of the last 6 bubbles are shown in Table 4.2, along with its corresponding standard deviations.

$\boldsymbol{\theta}$ [º]	$\boldsymbol{f}$ $[\boldsymbol{Hz}]$	$\boldsymbol{\sigma_f}$ $[\boldsymbol{Hz}]$	$\boldsymbol{U_G}$ $[\boldsymbol{m/s}]$	$\boldsymbol{\sigma_{U_G}}$ $[\boldsymbol{m/s}]$	$\boldsymbol{V_B}$ $[\mathbf{x10^{-10}}\boldsymbol{m^3}]$	$\boldsymbol{\sigma_{V_B}}$ $[\mathbf{x10^{-10}}\boldsymbol{m^3}]$	$\boldsymbol{L_B}$ $[\boldsymbol{mm}]$	$\boldsymbol{\sigma_{L_B}}$ $[\boldsymbol{mm}]$
0	632.0	69.4	1.098	0.016	4.275	0.598	1.568	0.167
15	602.5	55.6	1.144	0.043	4.608	0.788	1.682	0.231
25	574.6	59.2	1.129	0.042	4.888	0.681	1.735	0.194
45	461.6	40.4	1.053	0.011	6.153	0.378	1.984	0.136
90	370.7	20.1	1.031	0.012	9.049	0.717	2.369	0.154

Table 4.2 Contact angle tests results on bubble frequency, velocity, volume and length for a sample of the last 6 bubbles generated

As the contact angle value increases, the gas sticks more to the walls since the surface becomes more adherent. This causes the frequency of the bubbles to decrease, because the walls slow down the gas movement. These results are in agreement with similar studies [2], where the bubble generation frequency also decreased when increasing the contact angle value. Moreover, for $\theta = 45º$ and $\theta = 90º$, the gas is not developing in a smooth way, that is, bubbles that are forming at present time collide with previous bubbles, causing bubble coalescence and other undesirable effects. For this reason, high values of the contact angle are discarded for the final simulations. Therefore, a value of $\theta = 25º$ is chosen, since it causes a reliable (and close to reality) bubble generation process. It must be said that this is a complex problem and further work in this subject is required.

4.3. Measuring surfaces location

Locating the sampling surfaces very close to the T-junction can be a cause of imprecision in the measuring process, since in this region bubbles are still developing and stabilizing. On the other hand, the mesh gets coarser from the point $x = 15\,mm$ to the end of the horizontal capillary in order to reduce computational time and complexity (as it has been discussed in Chapter 3.2.2). So, a reasonable question that arises is where to locate the measuring surfaces in order to get good results. Another question one could ask is if the coarse mesh has a negative impact on the measuring process. In this subchapter these two questions will be addressed.

A test simulation is run placing the sampling surfaces at $x = 13\,mm, 14\,mm$ (inside the fine mesh region) and at $x = 19\,mm, 20\,mm$ (inside the coarse mesh region, but far away from the T-junction and close to the outlet), as it is shown in Fig. 4.9. Air is injected at $U_{SG} = 0.5m/s$ and water at $U_{SL} = 0.5m/s$.

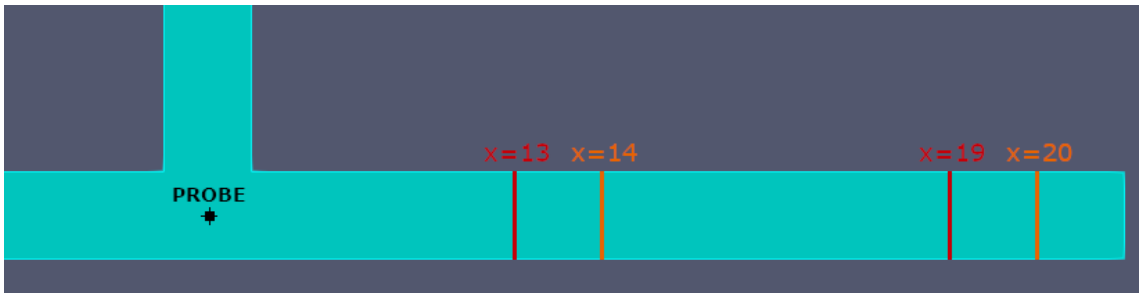

Fig. 4.9 Location of the test sampling surfaces

In Fig. 4.10 the average air fraction at the first and the last surfaces can be observed. Note that bubbles are way more regular and stable at $x = 20\,mm$ surface since the maximum air fraction value of every bubble ranges from 75% to 82%, while at $x = 13\,mm$ the maximum value ranges from 55% to 75%. This

is an indicator that the bubbles have not yet stabilized when they cross the first sampling surface.

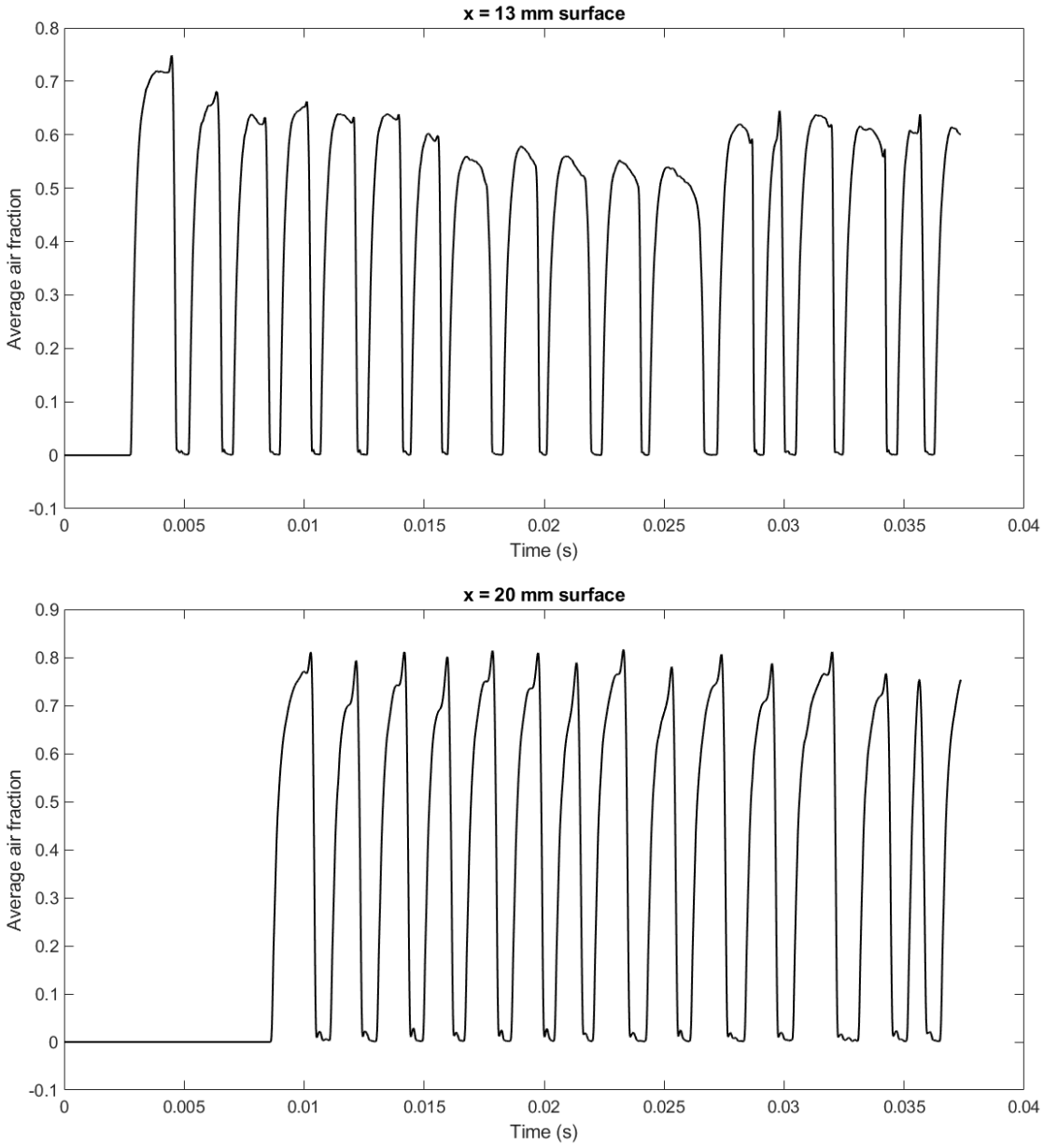

Fig. 4.10 Average air fraction at $x = 13\ mm$ and $x = 20\ mm$ surfaces

In Fig. 4.11 the frequency of the first 11 bubbles at every sampling surface is plotted. There is not a huge difference between surfaces when calculating the frequency, they follow the same tendency.

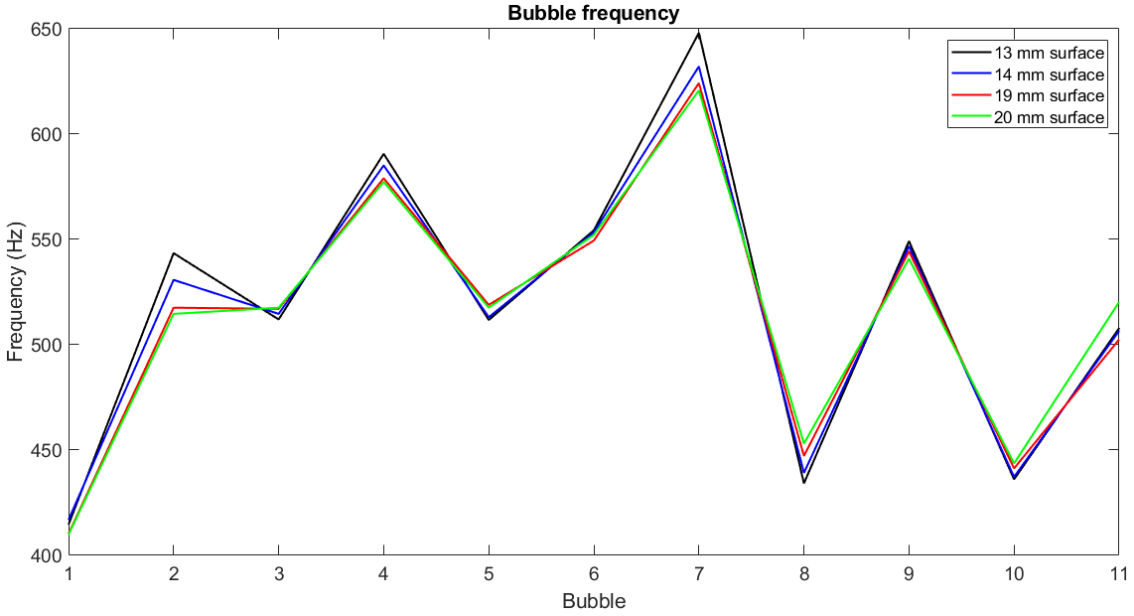

Fig. 4.11 Frequency of the first 11 bubbles for every sampling surface

The difference can be observed when plotting the velocity and length of every bubble; see Fig. 4.12 and Fig. 4.13. The shape of the curves is almost the same but the values are higher for the last two surfaces. The bubbles are $0.1\ m/s$ faster and $0.2\ mm$ longer in $x = 19\ mm, 20\ mm$ surfaces than in $x = 13\ mm, 14\ mm$ surfaces. This happens due to the reason that at the first two surfaces the bubbles are not fully grown and they continue to form as they advance through the capillary.

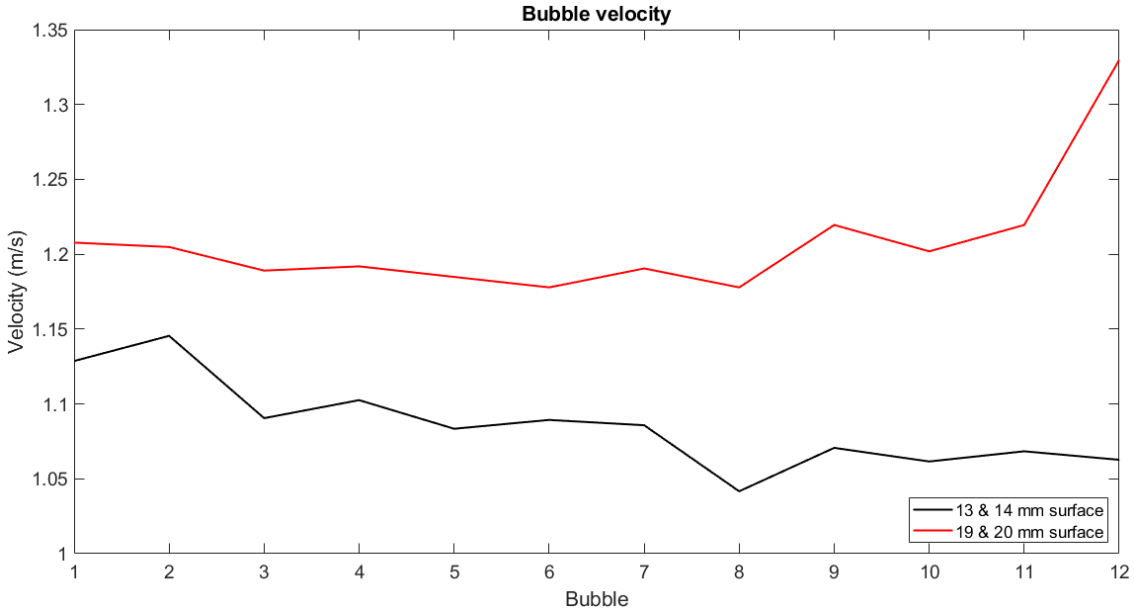

Fig. 4.12 Velocity of the first 12 bubbles at both surface doublets

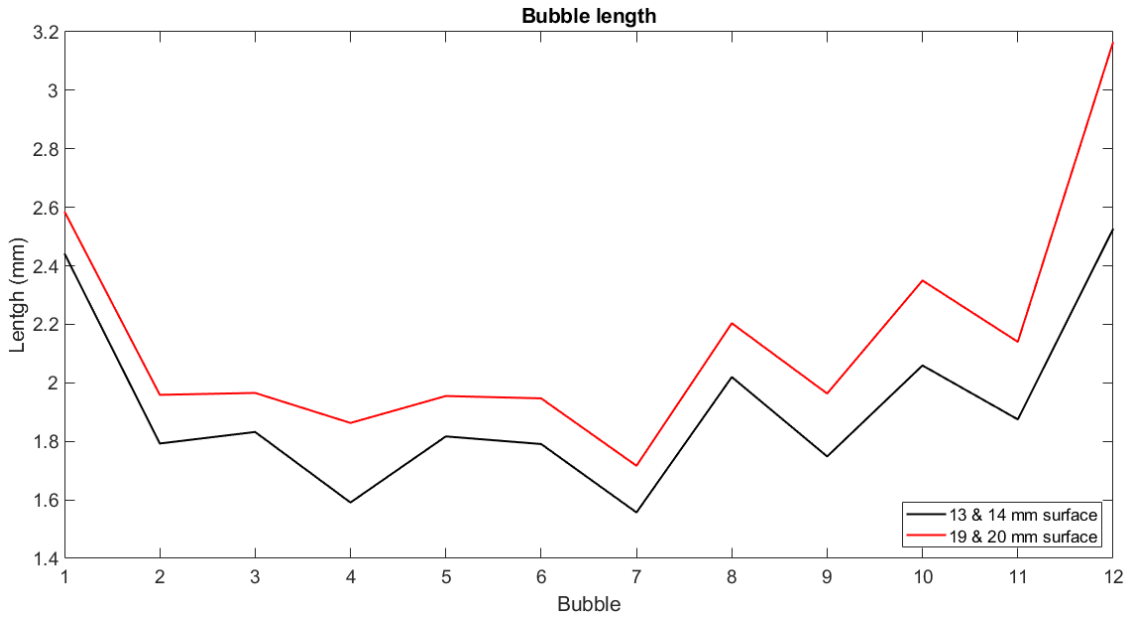

Fig. 4.13 Length of the first 12 bubbles at both surface doublets

Finally, the bubble volume has been calculated using Eq. (3.10) for the two pair of surfaces (13 & 14; 19 & 20) and using Eq. (3.11) for $x = 14\ mm$ and $x = 19\ mm$ surfaces; see Fig. 4.14 (method 2 in blue and green colors). It can be seen that the volume is conserved for the two different methods: red, blue and green curves are almost the same and black curve has lower values because the bubbles are not fully developed at the first two surfaces.

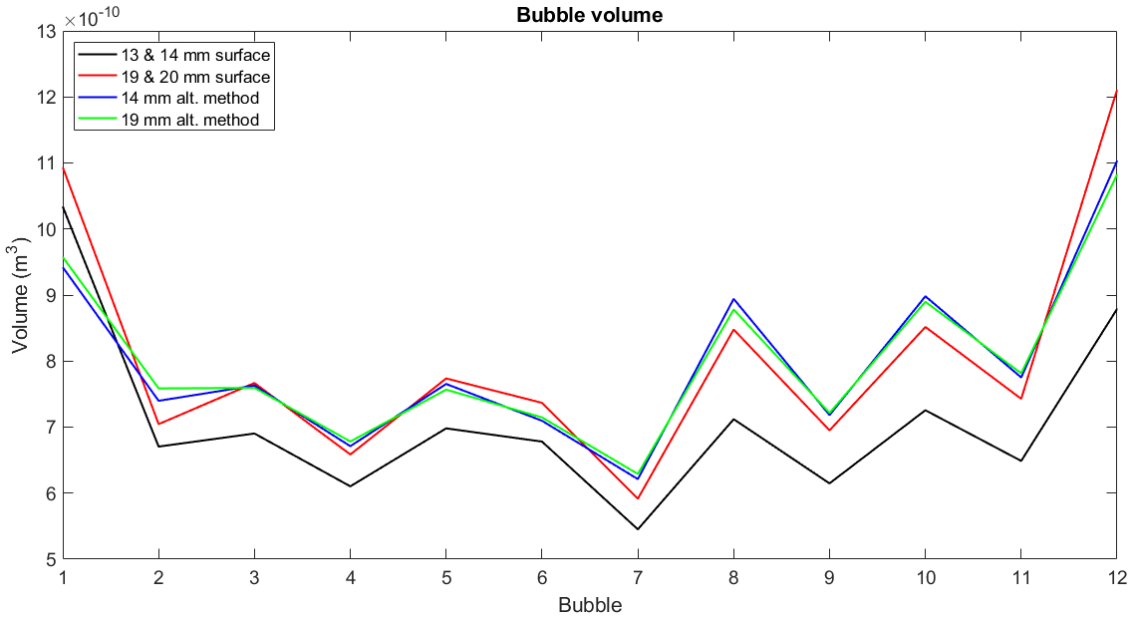

Fig. 4.14 Volume of the first 12 bubbles using two different methods

Therefore, in the light of the results, sampling surfaces will be located at $x = 19\,mm$ and $x = 20\,mm$. The coarse mesh has no significant impact on the measuring process, so it is a better option to place the surfaces as far as possible from the T-junction.

4.4. Velocity profiles

In this subchapter, different velocity profiles and velocity evolution plots are shown and discussed, corresponding to the $U_{SG} = U_{SL} = 0.5\,m/s$ case.

In Fig. 4.15 the horizontal velocity of the fluid (at the capillary centerline) versus the distance to the air inlet is shown. This information is plotted for four different instants of time. As it can be seen, the initial velocity of the fluid is $0.5\,m/s$ (the inlet velocity) and, as air travels through the capillary, this value grows up to $0.88\,m/s$ (the terminal velocity discussed in Chapter 4.1). From the point $x = 9.5\,mm$ onwards, the T-junction effects start to be noticeable and the velocity evolution becomes complex and unpredictable.

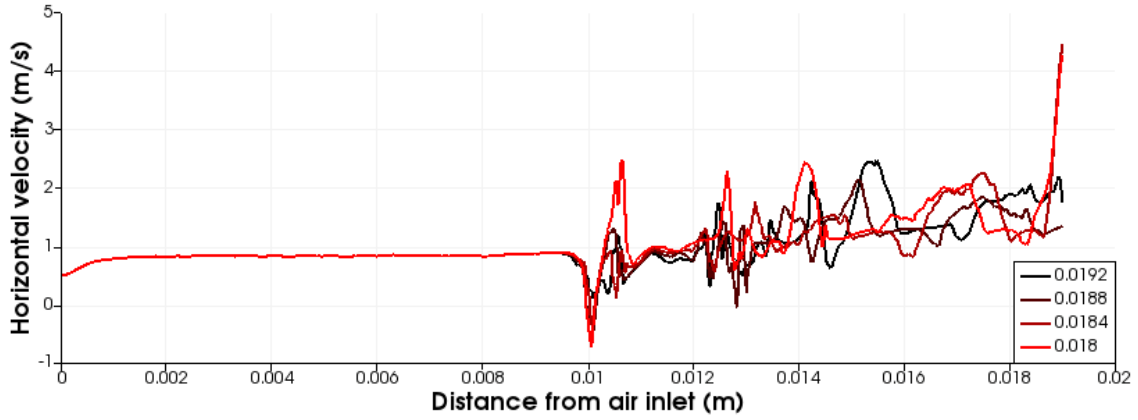

Fig. 4.15 Horizontal component of the velocity of the fluid at the capillary centerline versus horizontal distance to the air inlet for four different times

To have a better understanding of the velocity evolution after the mixing process of air and water at the T-junction, the velocity profiles from $x = 10\,mm$ to $x = 19\,mm$ at time $t = 0.0192\,s$ are shown in Fig. 4.16. In $x = 11\,mm$ and $x = 12\,mm$ surfaces, the peak of velocity is located below the centerline (negative values on the x-axis). That is, the flow has a higher velocity in the lower part of the capillary, which is in accordance with the internal view of the geometry. Bubbles are being generated at the lower part of the T-junction since the water pushes the air down in the T-junction. Therefore, the fluid in the lower part of the capillary moves at a higher velocity than the fluid at the upper part.

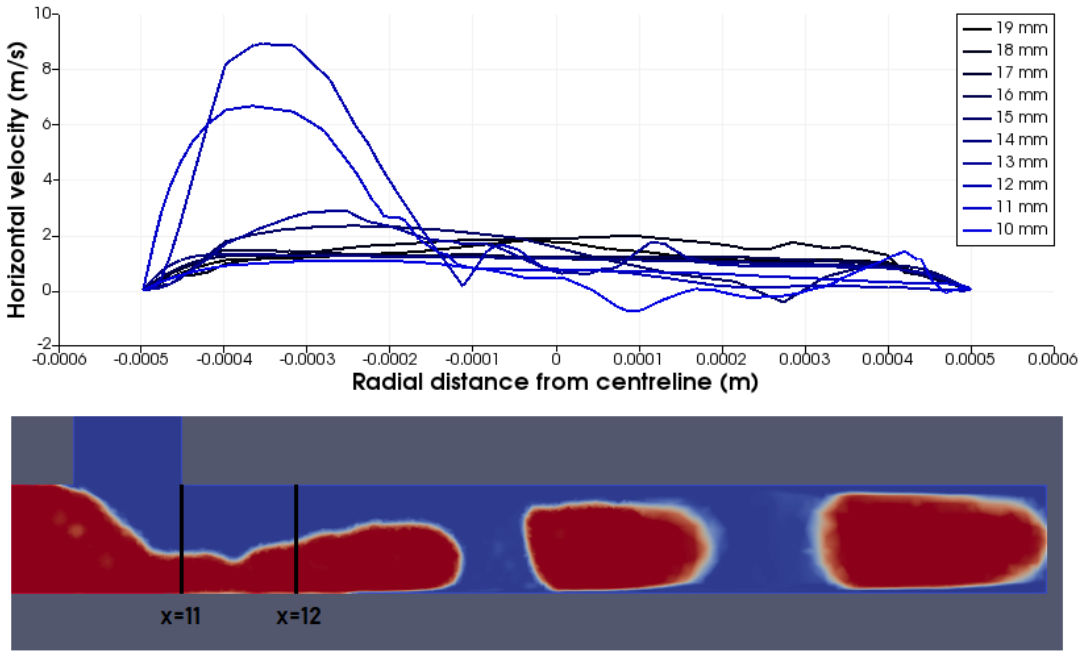

Fig. 4.16 Horizontal velocity profiles for 10 different sections and internal view of the geometry at $t = 0.0192\ s$

Finally, in Fig. 4.17 the vertical component of the velocity of water at $t = 0.0192\ s$ is shown along the vertical capillary. The values are negative since the water is injected downwards (negative Y-axis). In $x = 0\ mm$, it enters the geometry at $0.5\ m/s$ and accelerates up to a terminal velocity of $0.71\ m/s$ (not in accordance with analytical results), which is reached in $x = 4\ mm$, approximately. This velocity is maintained in the segment of $4 - 8\ mm$ and then has a small increase up to $0.74\ m/s$ just before entering the T-junction, because at this point the liquid encounters the gas and it has a smaller section to go through, so it accelerates.

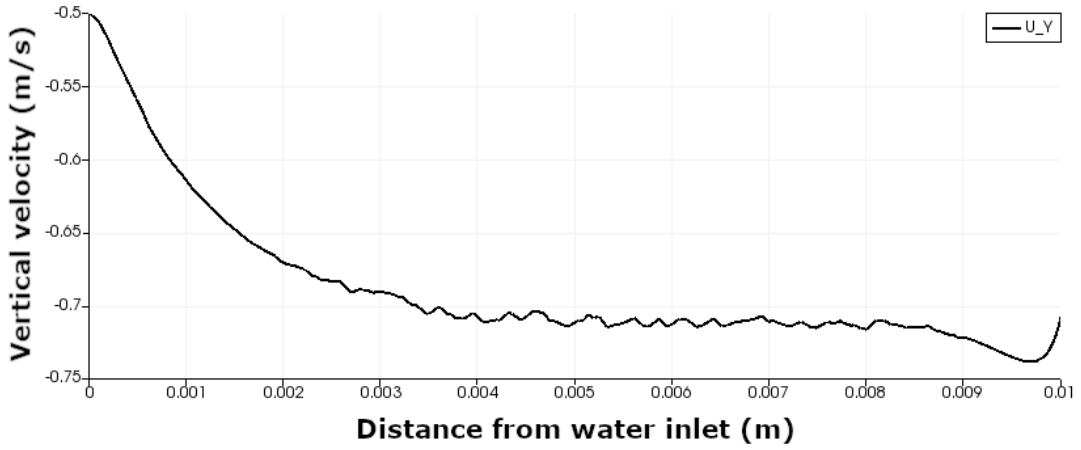

Fig. 4.17 Vertical component of the velocity of the fluid at the capillary centerline versus vertical distance to the water inlet at $t = 0.0192\ s$

CHAPTER 5. RESULTS

In this chapter, an analysis of the definitive simulations is done. All simulations are performed using the 680k cells T-junction mesh, a time step of 1 μs and a contact angle condition of 25 degrees. A total of 5 simulations are performed in which the injection velocity of air and water is the same, that is, $U_{SG} = U_{SL}$, and they range from $0.2\,m/s$ to $0.6\,m/s$ in intervals of $0.1\,m/s$. 12 bubbles are generated for the faster cases ($0.4\,m/s$, $0.5\,m/s$ and $0.6\,m/s$) and 5 bubbles are generated for the slower cases ($0.2\,m/s$ and $0.3\,m/s$), since these simulations are time consuming. In order to calculate the different parameters (frequency, volume, length, velocity), the last 8 bubbles are taken as a sample for the faster cases and the last 4 bubbles for the slower cases, because the results related to the first bubbles of the simulations are not representative.

Fig. 5.1 shows the comparison between cases of the bubble generation process. For the sake of clarity, all screenshots were taken at the same moment: just after the bubble breakup. As it can be seen, the size of the bubbles is smaller as the injection velocity increases, that is, both the volume and the length decrease. At low velocities (low Ca), the gas tends to fill completely the main channel, that is, the maximum average value of alpha α_{avg} in the transversal section of the bubble is almost 1. This causes a blockage of the main channel and a pressure buildup upstream the bubble, which generates a force that pushes the bubble downstream and provokes its breakup. This process is known as the squeezing mechanism, which is mainly driven by interfacial forces [31, 32]. As the injection velocity increases (higher Ca), the gas does not block the main channel anymore because the viscous forces exerted by the liquid have a predominant role and shear off the emerging bubble [1]. This process is known as the dripping mechanism, which is dominated by the liquid shearing forces. Moreover, as Ca increases, the bubble breakup region moves downstream and the liquid forces are responsible for cutting the bubbles off. Note also that the generation frequency and the irregularity of the bubbles increase and the separation between bubbles decreases. Regarding the form of the bubbles, they have a bullet shape: rounded at the leading edge and almost straight at the trailing edge; except for high Ca, where some bubbles have a spherical shape, and for low Ca, where bubbles are rounded both at the anterior and the posterior parts. This happens because, for the slower cases, the conditions are close to the quasi-stationary situation and the bubbles have enough time to adapt their front and back form to a semi-spherical shape, becoming almost symmetric.

Fig. 5.2 shows the comparison between the experiment at $U_{SG} = 0.5\,m/s$, $U_{SL} = 0.53\,m/s$ and the simulation at $U_{SG} = U_{SL} = 0.5\,m/s$. The shape of the bubbles is quite similar comparing the cases, but the bubbles of the experiments are way more regular than the simulated bubbles. This is due to the fact that in the laboratory it is easy to generate trains of hundreds of bubbles (in less than a second) and they have time to stabilize. In the simulations, generating hundreds of bubbles is totally unfeasible since it would take several weeks of computational time.

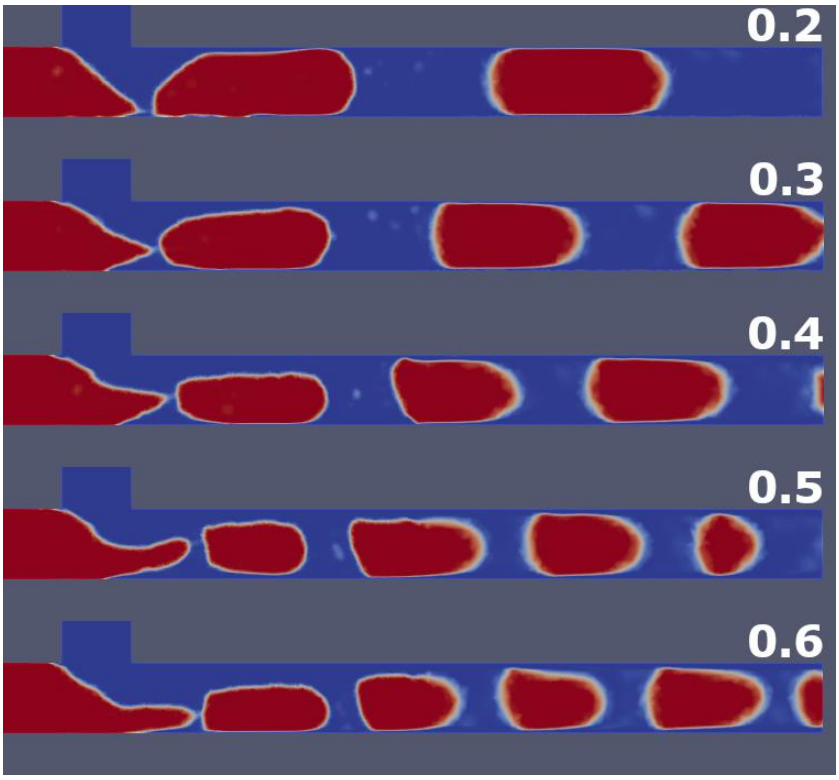

Fig. 5.1 Bubble generation comparison between five different cases

Fig. 5.2 Bubble generation comparison between the experiments and the simulations for the $U_{SG} = U_{SL} = 0.5\ m/s$ case

5.1. Pressure probe

From the pressure probe, located at the very center of the T-junction (see Chapter 3.4), the gauge pressure over time can be obtained for every case. For example, for $U_{SG} = U_{SL} = 0.4\ m/s$ case, the graph that is obtained is shown in Fig. 5.3.

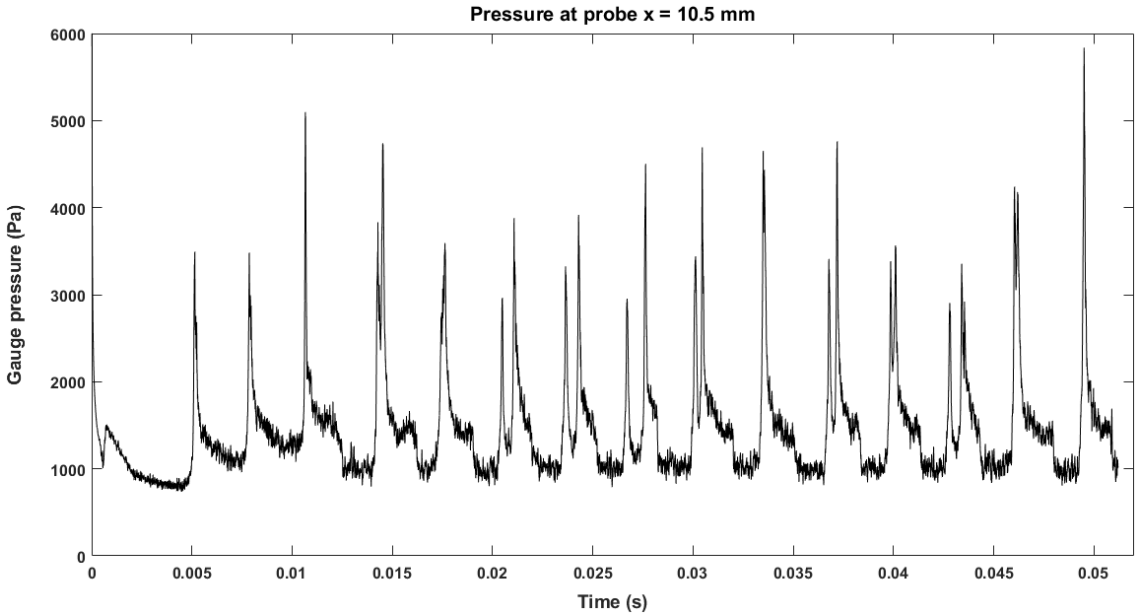

Fig. 5.3 Pressure evolution as a function of time for $U_{SG} = U_{SL} = 0.4\, m/s$ case

As it can be seen, there are cycles of pressure corresponding to certain points of the bubble formation process. To properly analyze one cycle of pressure, the following procedure is followed:

1. Select a pressure cycle and obtain its initial and final time.

2. Extract the pressure data on this interval and nondimensionalize the time axis with the duration of the cycle time. The initial time corresponds to $\bar{t} = 0$ and the final time corresponds to $\bar{t} = 1$.

The results are shown in Fig. 5.4. The selected pressure cycle corresponds to the 5th bubble of the $U_{SG} = U_{SL} = 0.4\, m/s$ case. The initial time of formation is $t = 0.0144\, s$ and the time when the bubble separates is $t = 0.0172\, s$, so it takes $2.8\, ms$ to generate. As it can be seen, the peak of pressure is located just after the breakup of the bubble. At this time, the liquid mostly fills the section and the gas must exert a bigger force to break the liquid resistance. Then, the pressure remains almost constant at $1500\, Pa$, since the gas has opened a channel to circulate through the liquid and this channel has a constant cross-sectional width. Finally, the breakup process of the forming bubble begins: the channel gas width shortens and the gas velocity increases, causing a decrease in pressure to a constant value of $1000\, Pa$, approximately.

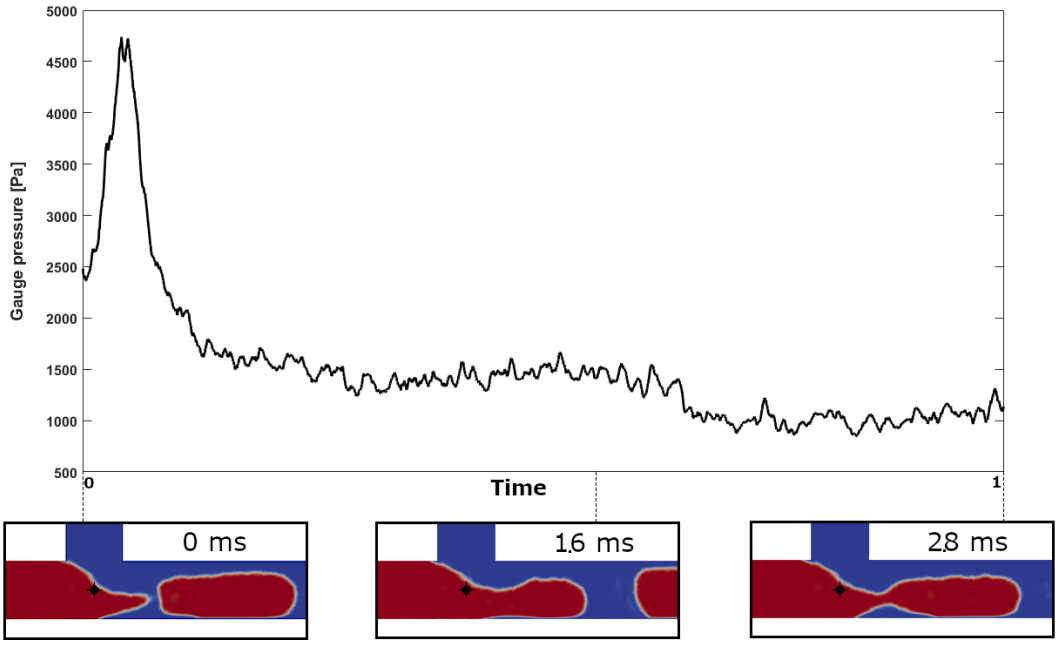

Fig. 5.4 Pressure cycle for the $U_{SG} = U_{SL} = 0.4\,m/s$ case

In order to compare the five different cases, a pressure cycle has been plotted for every case in Fig. 5.5. These cycles do not correspond exactly to a bubble generation cycle, but they show the peaks of pressure, which is the most important information extracted from the pressure probes. All the different cycles have a similar shape but a different peak pressure value. The maximum value of pressure for the $U_{SG} = U_{SL} = 0.6\,m/s$ case is roughly $6500\,Pa$, where for the $U_{SG} = U_{SL} = 0.2\,m/s$ case is $3500\,Pa$. The dynamic pressure inside a pipe depends on the square of the velocity, so it makes sense that an increase in the injection velocity causes an increase in the gauge pressure.

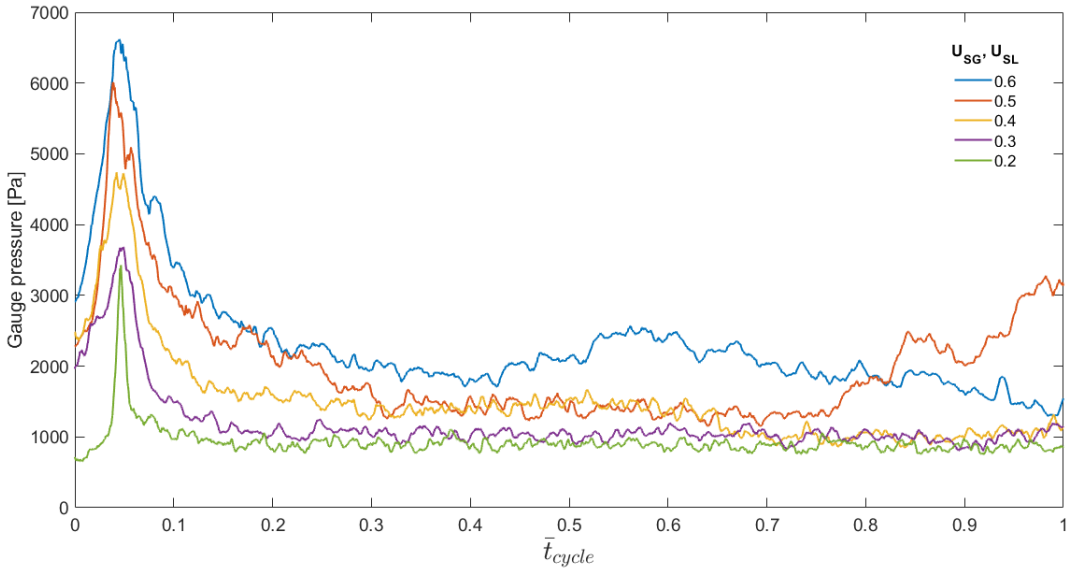

Fig. 5.5 Pressure cycle for five different superficial velocities

5.2. Bubble generation frequency

We discuss here the formation of bubbles by analyzing the bubble generation frequency results obtained from the sampling surfaces and comparing them to experimental data. As said before, one of the objectives of this study is to compare the values obtained via OpenFOAM with the experimental results obtained in [1].

First of all, Fig. 5.6 shows the frequency value for every bubble of the five different cases. As It can be seen, there are 11 values of frequency for the faster cases and 4 values for the slower cases (remember that 12 bubbles were generated for the faster cases and 5 for the slower ones). This is due to the reason that the frequency calculation requires two consecutive bubbles passing through the same sampling surface, so it is impossible to calculate the generation frequency for the last bubble.

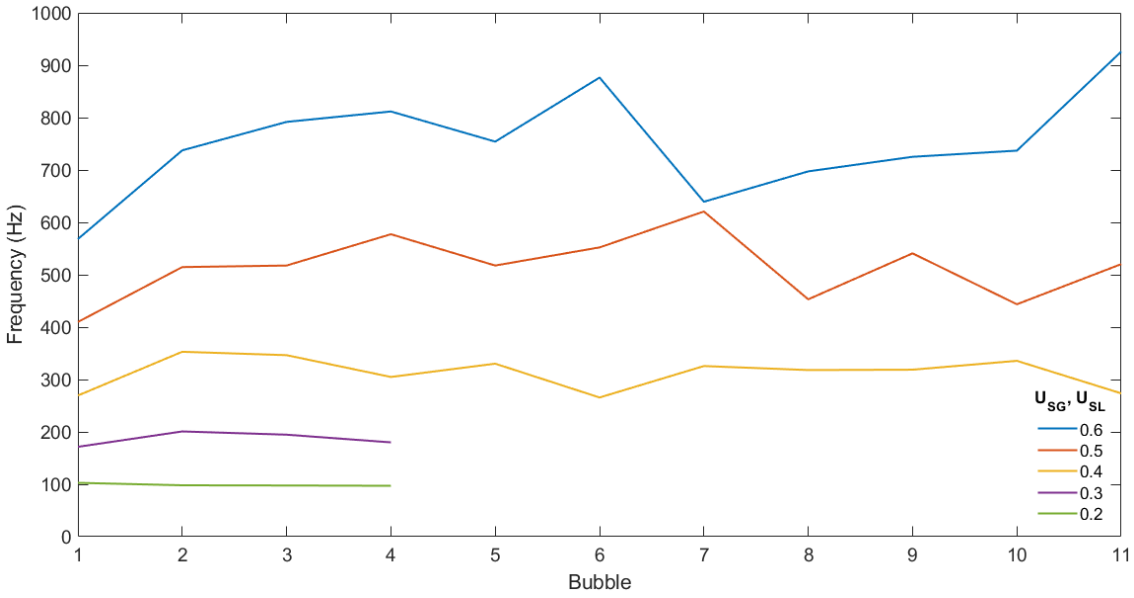

Fig. 5.6 Frequency of every bubble for the five different cases

Note that, as the injection velocity of both gas and liquid increases, the frequency of the bubbles increases too, along with the irregularity of the process. That is, there is a positive correlation between bubble frequency and injection velocity. Arias and Montlaur proposed in [3] a fitting exponential function to describe the behavior of the bubble generation frequency; see Eq. (5.1).

$$f = f_{sat}\left(1 - e^{-(a_0/f_{sat})U_{SG}}\right) \qquad \textbf{(5.1)}$$

Where f_{sat} is the saturation frequency in Hz and a_0 is the initial slope of the linear regime in m^{-1}. More information about these two parameters can be found in [3].

In order to compare the OpenFOAM case result with its corresponding experimental frequency curve, it is necessary to obtain the values of f_{sat} and a_0 for the capillary numbers of this study from [1]. To obtain the values of the initial slope, Fig. 5.7 can be used.

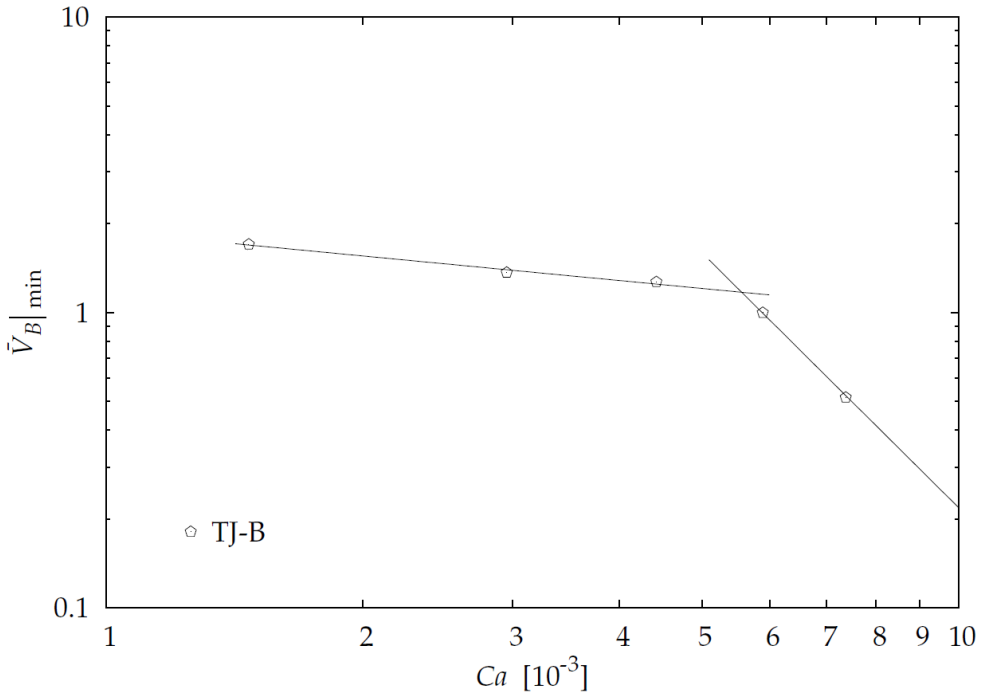

Fig. 5.7 Experimental normalized minimum bubble volume as a function of the capillary number [1]

It shows the normalized minimum bubble volume for some capillary number values. This volume is related with the initial slope of the linear regime in the following way: $\bar{V}_B|_{min} = 1/\bar{a}_0$ (see [1] for further explanation). As it can be seen, two different trends appear in this graph: the first one corresponds to the squeezing regime and the second one to the dripping regime. Eq.(5.2) shows the fit for the squeezing regime and Eq. (5.3) for the dripping regime.

$$\bar{V}_B|_{min} = 0.28\left(\frac{Ca}{1000}\right)^{-0.28} \quad \textbf{(5.2)}$$

$$\bar{V}_B|_{min} = 4.08 \cdot 10^{-7}\left(\frac{Ca}{1000}\right)^{-2.86} \quad \textbf{(5.3)}$$

Using these last two equations, it is possible to extract the values of a_0 for the capillary numbers used in the simulations. These values are shown in Table 5.1.

To obtain the experimental values of the saturation frequency, Fig. 5.8 can be used. This figure shows the saturation frequency value as a function of the Capillary number for the experiments performed in [1].

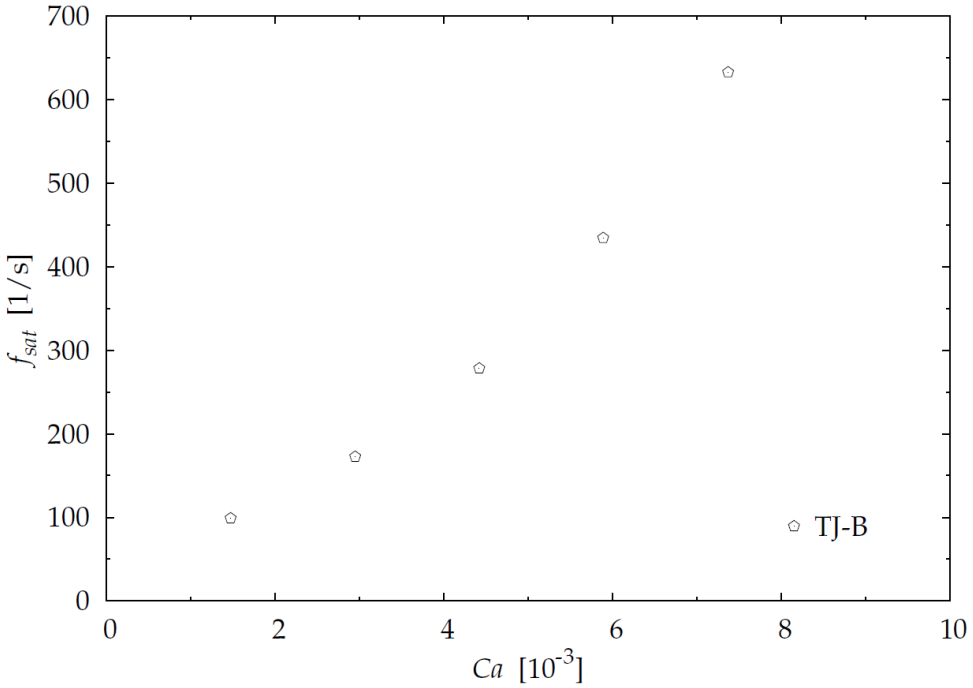

Fig. 5.8 Experimental saturation frequency as a function of the capillary number [1]

An interpolation between values is made in order to extract the saturation frequency for the capillary values of this study. The values that are obtained are shown in Table 5.1.

U_{SL}, U_{SG} $[m/s]$	Ca $[\mathbf{x10^{-3}}]$	a_0 $[m^{-1}]$	f_{sat} $[Hz]$
0.6	8.33	2714	850
0.5	6.94	1610	600
0.4	5.56	854	400
0.3	4.17	788	275
0.2	2.78	705	175

Table 5.1 Experimental values of a_0 and f_{sat} for the five simulated cases

Once the experimental data are obtained, it is possible to plot the simulated frequency results over the experimental frequency curves; see Fig. 5.9. The value of a_0 corresponds to the slope of the curve near the origin and the value of f_{sat} corresponds to the horizontal asymptote.

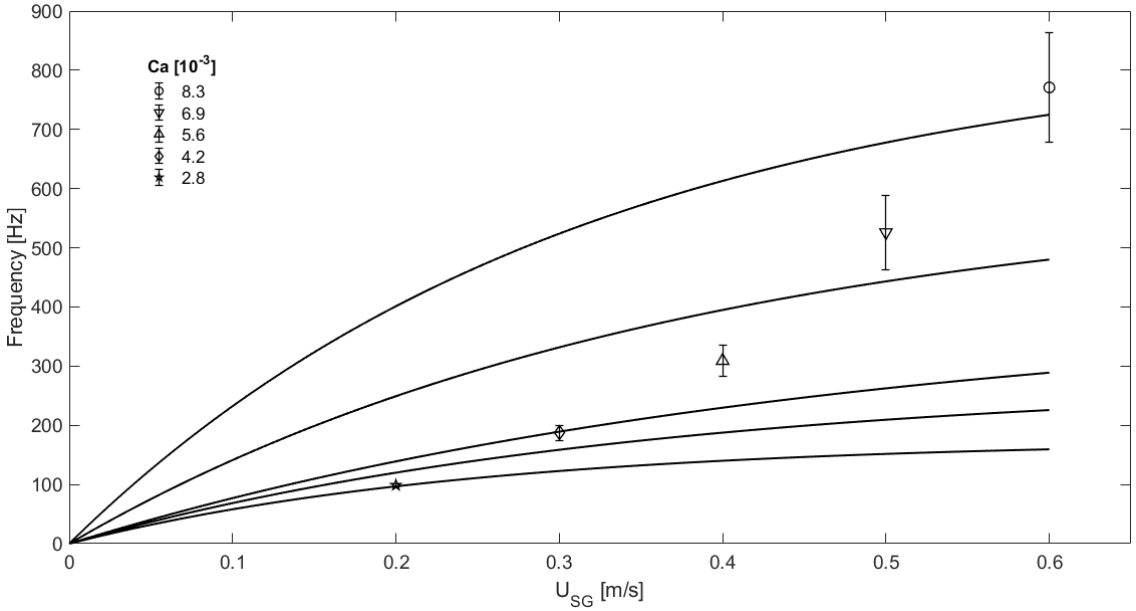

Fig. 5.9 Bubble frequency as a function of the gas superficial velocity for five different capillary numbers. Lines correspond to Eq. (5.1) and are shown in the same order as the capillary numbers of the legend

The points correspond to the mean value of the generation frequency for every case. The error bars correspond to the standard deviation of the bubble samples. The solid lines represent the experimental curves of Eq. (5.1) using the f_{sat} and a_0 corresponding to each capillary number. As it can be seen, the $U_{SG} = U_{SL} = 0.2\,m/s$ $(Ca = 2.8 \cdot 10^{-3})$ case and the $U_{SG} = U_{SL} = 0.6\,m/s$ $(Ca = 8.3 \cdot 10^{-3})$ case are the cases which get closer to the experimental values in terms of frequency. But, in general, the bubble frequency of the simulations is higher than the experimental bubble frequency. It can also be observed how the standard deviation increases as the superficial velocities increase, that is, faster bubbles are more irregular than slower bubbles.

Another interesting way to present the results is to nondimensionalize both the vertical and horizontal axes. The generation frequency of each case can be normalized with its corresponding experimental saturation frequency f_{sat} and the gas superficial velocity can be normalized with the crossover point between the linear and the saturation regimes, calculated in the following way: $U_{SG,CP} = f_{sat}/a_0$. For a more detailed explanation, see [1]. Thus, the resulting normalized equation can be written as seen in Eq. (5.4).

$$\bar{f} = 1 - e^{-\bar{U}_{SG}} \quad \textbf{(5.4)}$$

The points shown in Fig. 5.9 have been normalized and included in Fig. 5.10 along with Eq. (5.4) plot.

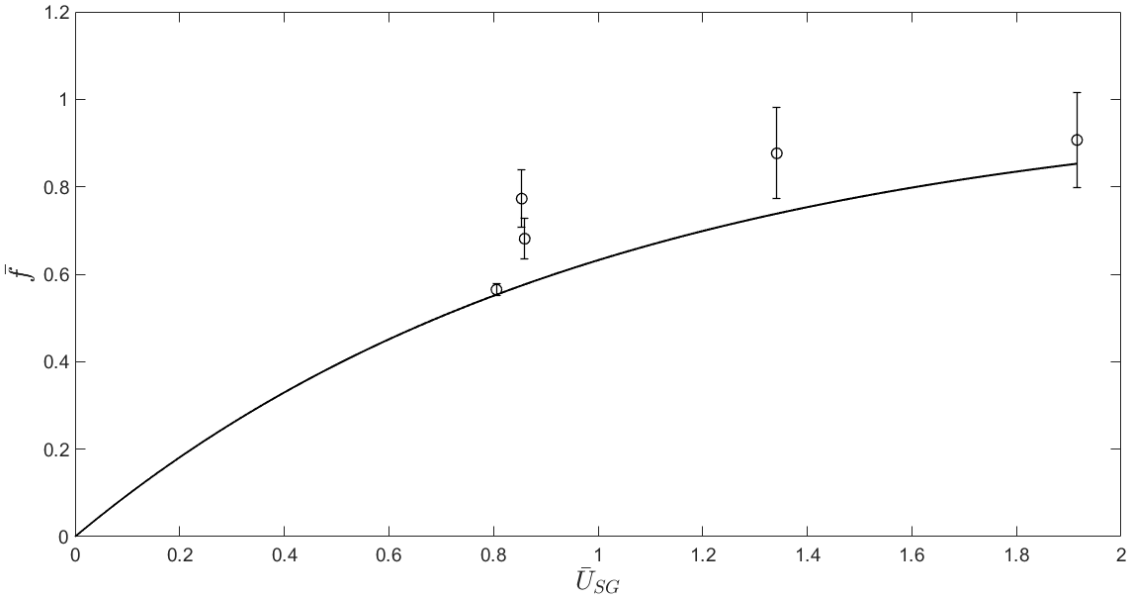

Fig. 5.10 Normalized frequency as a function of the normalized gas superficial velocity. Line corresponds to Eq. (5.4)

The simulated results adjust to the theoretical curve quite well. As said before, 0.2 and 0.6 cases are the most accurate ones, but qualitatively all the values are satisfactory. Note that the simulations results are plotted in terms of the experimental results, so it is natural that they do not adjust perfectly. However, the tendency of (5.4) is followed, which is an excellent result.

5.3. Bubble volume

The volume of every bubble for all different cases is shown in Fig. 5.11. As it can be observed, a different tendency is followed compared to the generation frequency. The volume of the bubbles is quite small for high injection velocities and it gets bigger as the superficial velocities decrease. That is, volume and injection velocity have an inverse correlation.

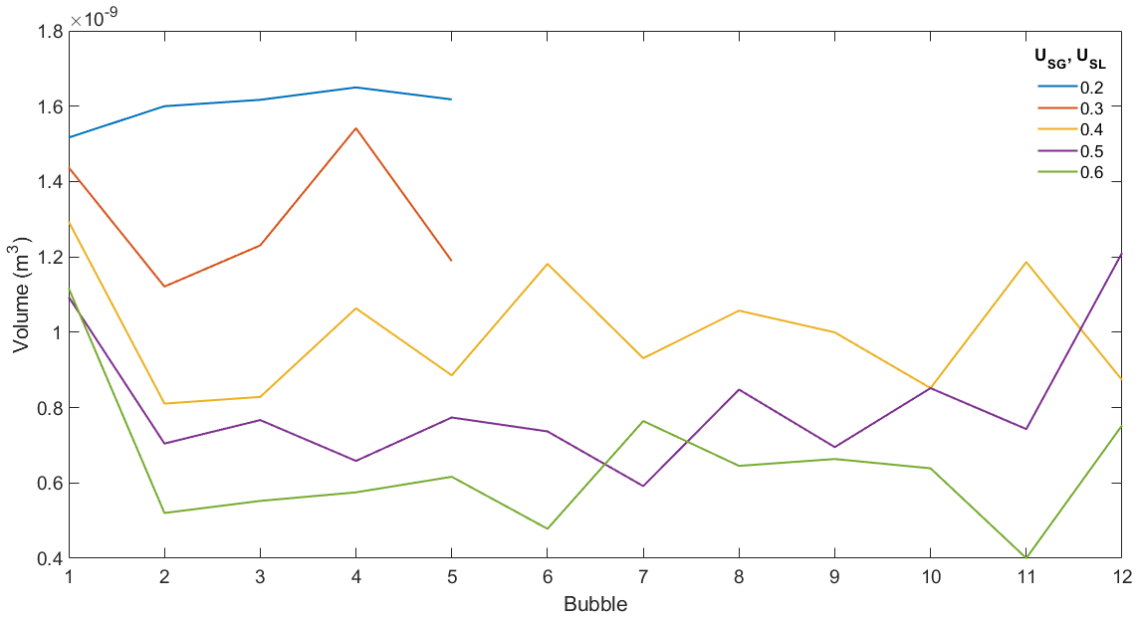

Fig. 5.11 Volume of every bubble for the five different cases

To have a better understanding of this dependency, the mean value of the volume and its corresponding standard deviation can be calculated for each case and plotted in a separate graph; see Fig. 5.12. The volume has been normalized with the channel equivalent volume $V_c = A_c\phi_c = \frac{\pi}{4}{\phi_c}^3$. The straight line represents the exponential fit of the last four points and it has the form of Eq. (5.5). A double logarithmic scale is used to represent the data.

$$\bar{V}_B = 7.99 \cdot 10^{-3} \left(\frac{Ca}{1000}\right)^{-0.97} \quad \textbf{(5.5)}$$

As said before, there are two different trends of gas breakup mechanism for low capillary numbers: the squeezing and the dripping regimes, already reported in the related literature [1, 31]. In the squeezing regime, interfacial forces dominate the shear stresses and the dynamics of bubble formation are directly connected to the confined geometry [32]. Consequently, the bubble blocks almost the entire cross-section of the main channel and confines the flow of water to thin wetting films on the walls of the microchannel. At a critical value of the capillary number, the system transits into a shear-dominated or dripping mechanism of bubble formation [31]. The squeezing-to-dripping transition takes place within the range of $Ca = [2 - 5.9] \cdot 10^{-3}$ in the related literature [1, 33, 34]; where similar T-junction two-phase flows experiments were performed.

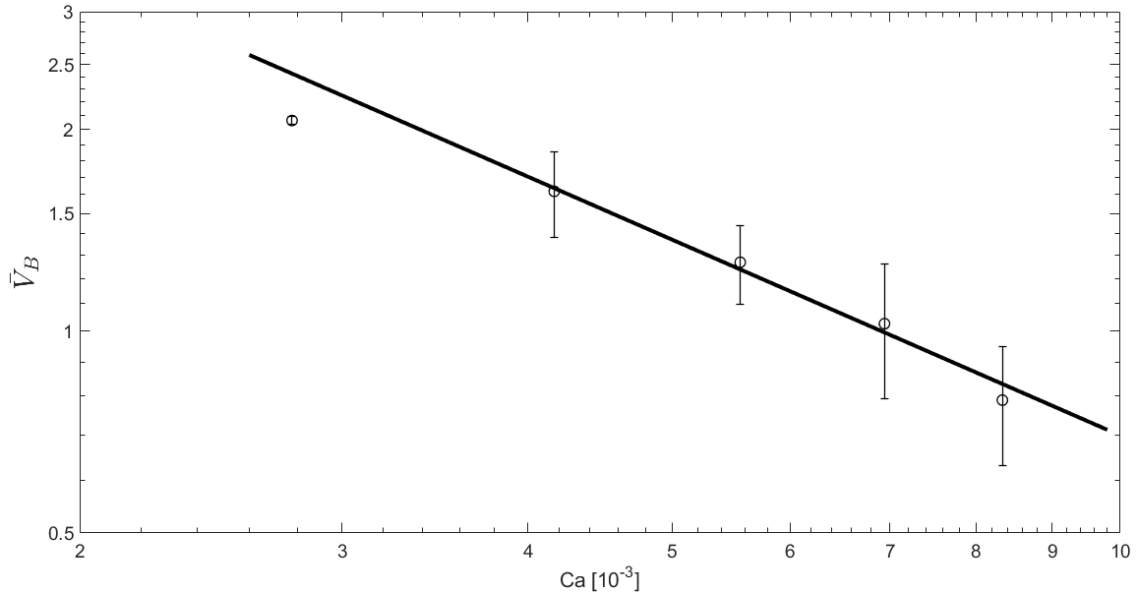

Fig. 5.12 Normalized bubble volume as a function of the capillary number. Line corresponds to Eq. (5.5)

By looking at Fig. 5.12, the linear trend of the dripping regime is clearly visible. Nonetheless, note that the volume corresponding to $Ca = 2.8 \cdot 10^{-3}$ is much lower than the estimated volume using the dripping line, which means that the squeezing-to-dripping transition is most likely located in the range of $Ca = [2.8 - 4.2] \cdot 10^{-3}$. In order to confirm that, an extra simulation at $U_{SG} = U_{SL} = 0.1\ m/s$ (corresponding to $Ca = 1.39 \cdot 10^{-3}$) was tried, but it diverged, unfortunately. These simulations are complex and time consuming, so this low-capillary case is left as future work.

In order to study the regularity of the bubble generation process, the volume polydispersity index (I_p) can be used, which is defined in Eq. (5.6).

$$I_p\ [\%] = \left(\frac{\sigma_{V_B}}{V_B}\right) \cdot 100 \qquad \textbf{(5.6)}$$

Where σ_{V_B} is the bubble volume standard deviation. Thus, the bubble volume variability is studied as a relative change concerning the V_B of each case. In Fig. 5.13 the volume polydispersity index as a function of the capillary number can be seen.

In general, there is a positive correlation between the polydispersity index and the capillary number. That is, the higher the injection velocity of the fluids, the higher the irregularity of the bubbles. This trend is coherent with the experimental results [1], but, quantitatively, the dispersity of the simulated bubbles is way bigger than the experimental ones, where $I_p < 2\%$ for every $Ca < 8.8 \cdot 10^{-3}$, and from this point on it begins to increase sharply.

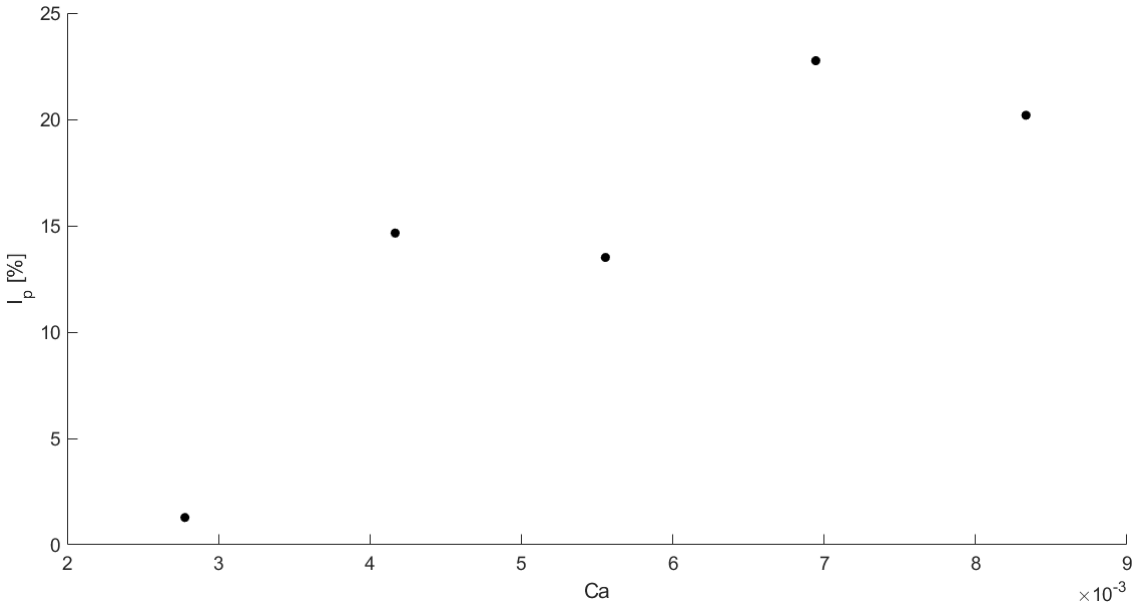

Fig. 5.13 Volume polydispersity index as a function of the capillary number

5.4. Bubble velocity

The bubble velocity is shown in Fig. 5.14 as a function of the average mixture superficial velocity U_M.

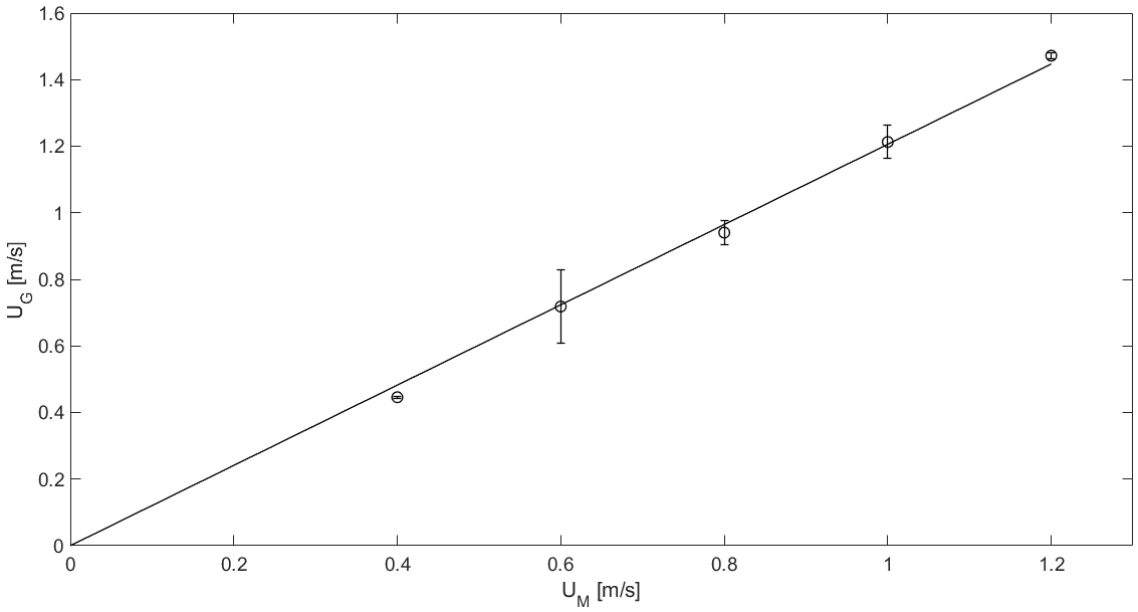

Fig. 5.14 Bubble velocity as a function of the average mixture superficial velocity. Line corresponds to Eq. (2.5)

The bubble velocity increases as the mixture superficial velocity increases (as expected), following a linear tendency: $U_G = C_0 U_M$. This equation is derived from the drift-flux model under the hypothesis of nondominant gravitational forces [35]. The parameter that relates both velocities is the void fraction distribution coefficient C_0, which is the slope of the line. Using a linear fit over the data, the coefficient happens to be $C_0 = 1.21$, which is coherent with the range of $1.1 - 1.2$ reported in the related literature for two-phase flows [2, 3]. Since it has a value bigger than 1, it means that the bubbles move faster than the mixture. This makes sense, since the bubbles are located at the capillary centerline, where the highest velocities are found. The standard deviation is quite small for all the cases, except for $U_{SG} = U_{SL} = 0.3\ m/s$, where the velocity of the bubbles is more irregular.

5.5. Bubble length

In Fig. 5.15 the bubble length has been nondimensionalized with the capillary diameter ϕ_c and has been plotted as a function of the capillary number.

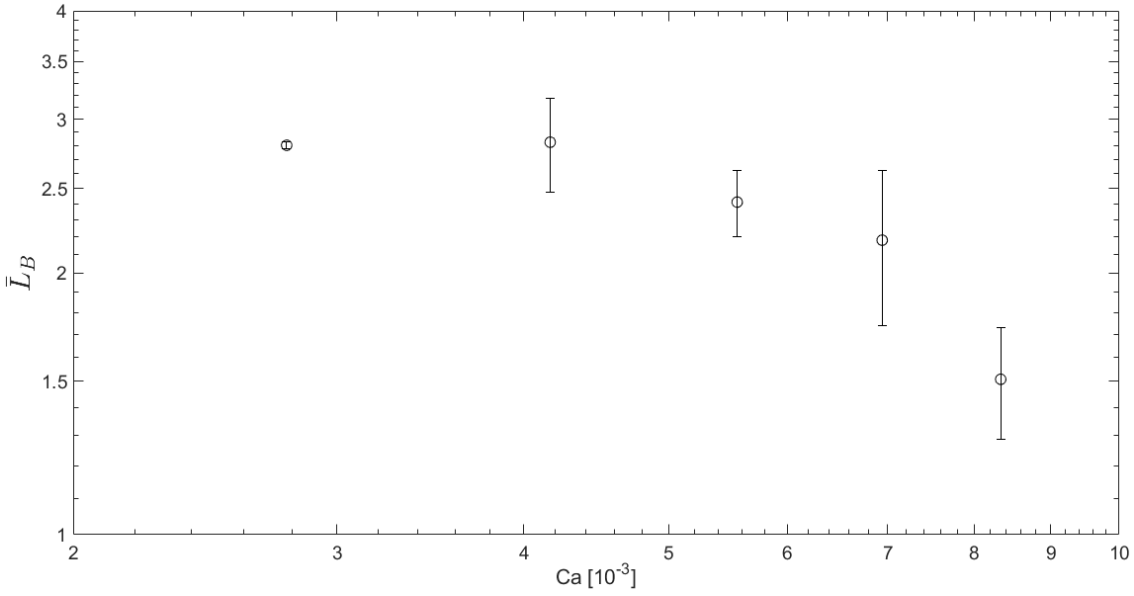

Fig. 5.15 Normalized bubble length as a function of the capillary number

It can be observed how the bubble length is almost constant up to $Ca = 4.2 \cdot 10^{-3}$, that is, the average length of the bubbles is roughly the same for the $U_{SG} = U_{SL} = 0.2\ m/s$ case and the $U_{SG} = U_{SL} = 0.3\ m/s$ case. Then, the length of the bubbles decreases with a higher slope. This could be an indication that the squeezing-to-dripping transition is located in the range of $Ca = [4.2 \cdot 10^{-3}, 5.4 \cdot 10^{-3}]$ since before the first value the liquid shearing forces do not have a significant influence over the bubble length (squeezing regime) and, after the second value, the length of the bubbles decreases sharply, which means that the liquid shearing forces start to be important (dripping regime).

CHAPTER 6. CONCLUSIONS

This project presents a study of the bubble generation process inside a T-junction capillary in microgravity conditions. Numerical simulations were run using OpenFOAM v6.0 and the data was post-processed using ParaView and MATLAB. Both channels of the T-junction had a diameter of $1\ mm$, so the Bond number was small enough to neglect the effects of gravity. The bubble generation process was mainly driven by the interaction between interfacial and viscous forces. Thus, the Capillary number became the most important dimensionless number of this study.

First of all, the setup of the definitive cases was done by trying different boundary conditions and by doing certain validations which ensured a good accuracy in the results. It was proven that a longer length of the capillaries improved the accuracy of the results, since the flow became fully-developed. $10\ mm$ long inlet and outlet capillaries were chosen for the final simulations. Tests on the contact angle boundary condition showed that high values of this parameter ($\theta > 45º$) resulted in an unsatisfactory bubble generation process since the bubbles adhered too much to the walls. A value of $\theta = 25º$ was chosen, but this is a complex problem where there might be a different optimal contact angle for each velocity case, so a separate study should be done for every simulation. Finally, the sampling surfaces were located at the end of the exit capillary ($x = 19\ mm$, $x = 20\ mm$) since placing them too close to the T-junction showed that the flow was not yet stabilized.

Five different simulations were run where $U_{SL} = U_{SG}$ and the values ranged from $0.2\ m/s$ to $0.6\ m/s$ in $0.1\ m/s$ increments. The results were analyzed in terms of the bubble frequency, volume, length, velocity and the gauge pressure at the center of the T-junction. The formation of the bubbles was affected by the injection gas and liquid flow rates. For instance, the dispersity on the bubble volume increased as the superficial velocities increased (that is, more irregular bubbles as Ca grew). Qualitatively, this is in accordance with the experiments performed in [1]. The results on the generation frequency approached the experimental curves, but the values were higher for certain simulation cases (a maximum of $+34\%$ for $U_{SL} = U_{SG} = 0.4\ m/s$ case). The results on the bubble volume and bubble length showed that the squeezing-to-dripping transition might be located in the range of $Ca = [2.8 - 5.6] \cdot 10^{-3}$, but extra simulations and further analysis are needed in order to assure this statement. The velocity of the bubbles increased linearly with the mixture superficial velocity and a distribution coefficient $C_0 = 1.21$ was obtained. The analysis of the pressure probe showed that higher injection velocities provoked higher peaks of pressure, as expected.

For future projects, the exit boundary conditions should be studied in more detail, since the gas is being absorbed at the outlet, exiting the geometry in an unrealistic way. An extensive study of the contact angle boundary condition should also be done, since the bubbles stuck too much to the walls in some cases, causing some simulations to diverge.

BIBLIOGRAPHY

[1] Arias S., “Comparison of Two Gas Injection Methods for Generating Bubbles in a T-Junction” (2020).

[2] Arias S. and Montlaur A., “Influence of Contact Angle Boundary Condition on CFD Simulation of T-Junction”, *Microgravity-Science and Technology*, pp.1-9 (2018).

[3] Arias S. and Montlaur A., “Numerical Study and Experimental Comparison of Two-Phase Flow Generation in a T-Junction”, *AAIA Journal*, 55 (5), pp. 1565-1574 (2017).

[4] What Are Zero Gravity and Microgravity – Geekswipe. https://geekswipe.net/science/physics/what-are-zero-gravity-and-microgravity-and-what-are-the-sources-of-microgravity/. On-line (03/2020).

[5] Top Fuel Racing Class – NHRA. https://www.nhra.com/nhra-101/drag-racing-classes. On-line (03/2020).

[6] The Bremen Drop Tower (ZARM) – Universität Bremen. https://www.zarm.uni-bremen.de/en/drop-tower/general-information.html. On-line (04/2020).

[7] How a Parabolic Flight Works – Zero Gravity Corporation. http://gozerog.com/index.cfm?fuseaction=Experience.How_it_Works. On-line (04/2020).

[8] Parabolic Flight Image. https://www.2luxury2.com/wp-content/uploads/S3-Zero-G-Flight-Parabola-950x633.png. On-line (04/2020)

[9] What is a Sounding Rocket? – NASA. https://www.nasa.gov/missions/research/f_sounding.html. On-line (04/2020).

[10] Hestroffer D. et al., “Small Solar System Bodies as Granular Systems”, *EPJ Web of Conferences*, Vol. 140, p.4, Fig. 2 (2017).

[11] McQuillen J., “Two-phase Flow and Space-based Applications”, NASA technical report (1999).

[12] Basic Microfluidic Concepts – University of Washington. https://faculty.washington.edu/yagerp/microfluidicstutorial/basicconcepts/basicconcepts.htm. On-line (04/2020).

[13] Herbig B. A., Yu X. and Diamond S. L., “Using Microfluidic Devices to Study Thrombosis in Pathological Blood Flows”, *Biomicrofluidics*, Vol. 12, 042201 (2018).

[14] Neumann C. and Rapp B. E., “Fluidic Platforms and Components of Lab-on-a-Chip Devices”, Ch. 5*, Lab-on-a-Chip Devices and Micro-Total Analysis Systems*, Springer, p. 89, Fig. 5.2 (2015).

[15] Aboukhedr M., Georgoulas A., Marengo M., Gavaises M. and Vogiatzaki K., “Simulation of Micro-Flow Dynamics at Low Capillary Numbers Using Adaptive Interface Compression”, *Computers and Fluids Journal*, pp. 35-40 (2018).

[16] Suo M. and Griffith P., “Two-Phase Flow in Capillary Tubes”, *Journal of Basic Engineering*, Vol. 86, No.3, pp. 576-582, (1964).

[17] Rezkallah K. S., “Weber Number Based Flow-Pattern Maps for Liquid-Gas Flows at Microgravity”, *International Journal of Multiphase Flow*, Vol. 22, No. 6, pp. 1265-1270 (1996).

[18] Schlichting H., “Origin of Turbulence I”, Ch. 16, *Boundary-Layer Theory 7th edition*, McGraw-Hill, pp. 449-452 (1979).

[19] Courant-Friedrichs-Lewy condition – CFD Online. https://www.cfd-online.com/Wiki/Courant%E2%80%93Friedrichs%E2%80%93Lewy_condition. On-line (04/2020).

[20] Hebbar R. S., Isloor A. M. and Ismail A. F., “Contact Angle Measurements”, Ch. 12, *Membrane Characterization*, Elsevier, p. 233, Fig. 12.10 (2017).

[21] Mechanisms behind the Lotus effect – Atomic World. http://www.hk-phy.org/atomic_world/lotus/lotus02_e.html. On-line (05/2020).

[22] Navier-Stokes equation Millennium Problem – Clay Mathematics Institute. https://www.claymath.org/millennium-problems/navier-stokes-equation. On-line (05/2020).

[23] About OpenFOAM – OpenFOAM, the open source CFD toolbox. https://www.openfoam.com/. On-line (05/2020).

[24] OpenFOAM User Guide – OpenFOAM, the open source CFD toolbox. https://www.openfoam.com/documentation/user-guide/. On-line (05/2020).

[25] Dalfó B., “CFD Study of the Bubble Generation Process in a T-Junction with Inversed Flows”, Treball de fi de grau: http://hdl.handle.net/2117/121361 (2018).

[26] Rosado G., “Numerical Study with OpenFOAM of the Bubble Generation Process in an Inverse T-Junction in microgravity conditions”, Treball de fi de grau: http://hdl.handle.net/2117/167251 (2019).

[27] OpenFOAM Boundary Conditions – OpenFOAM, the open source CFD toolbox. https://www.openfoam.com/documentation/user-guide/standard-boundaryconditions.php. On-line (05/2020).

[28] InterFoam solver – Unofficial OpenFOAM Wiki. https://openfoamwiki.net/index.php/InterFoam. On-line (05/2020).

[29] Çengel Y.A. and Cimbala J.M., “Flow in Pipes”, Ch. 8, *Fluid Mechanics: Fundamentals and Applications*, McGraw-Hill, pp. 325-329 (2006).

[30] Bergman T.L. and Incropera F.P., “Internal Flow”, Ch. 8, *Fundamentals of Heat and Mass Transfer*, John Wiley & Sons, pp. 519-522 (2002).

[31] De Menech M., Garstecki P., Jousse F. and Stone H., “Transition from Squeezing to Dripping in a Microfluidic T-Shaped Junction”, *Journal of Fluid Mechanics*, Vol. 595, pp. 141-161 (2008).

[32] Garstecki P., Fuerstman M., Stone H. and Whitesides G., “Formation of Droplets and Bubbles in a Microfluidic T-Junction – Scaling and Mechanism of Break-up”, *Lab Chip Journal*, Vol. 6, pp. 437-446 (2006).

[33] Oishi M., Kinoshita H., Fujii T. and Oshima M., “Confocal Micro-PIV Measurement of Droplet Formation in a T-shaped Micro-Junction”, *Journal of Physics: Conference Series*, Vol. 147, pp. 1-9 (2009).

[34] Guo F. and Chen B., “Numerical Study on Taylor Bubble Formation in a Micro-channel T-Junction Using VOF Method”, *Microgravity Science and Technology*, Vol. 21, pp. 51-58 (2009).

[35] Zuber N. and Findlay J., “Average Volumetric Concentration in Two-Phase Systems”, *Journal of Heat Transfer*, Vol. 87, pp. 453-468 (1965).

[36] White F. M., “Differential Relations for Fluid Flow”, Ch. 4, *Fluid Mechanics 7th edition*, McGraw-Hill, pp. 272-273 (2011).

APPENDICES

APPENDIX A. OpenFOAM

In this appendix, a detailed discussion of the OpenFOAM case of this project is done. It is also explained how to set up the case and run the simulation in the EETAC cluster. Lastly, a brief ParaView tutorial is done.

A.1. Case folder

We are going to discuss the ***case4*** case folder, which corresponds to the $U_{SL} = U_{SG} = 0.4\ m/s$ case. As it can be seen on the header of every file, the OpenFOAM version used for this project is OpenFOAM v6.

A.1.1 0 folder

This folder contains the ***alpha.air*** (Fig. A.1), ***p_rgh*** (Fig. A.2) and ***U*** (Fig. A.3) files. Every OpenFOAM file starts with a header displaying the information of the installed version. Then, the line *dimensions* defines the dimensions of the file, where each vector position corresponds to: $[\mathrm{kg},\ \mathrm{m},\ \mathrm{s},\ \mathrm{K},\ \mathrm{mol},\ \mathrm{A},\ \mathrm{cd}]$. For example, the pressure has units of $\frac{kg}{m\ s^2}$, so the *dimensions* vector is: $[1, -1, -2, 0, 0, 0, 0]$. The line *internalField* defines the value of the parameter in the entire domain at the initial time. The *boundaryField* section defines the boundary conditions with its corresponding value for every boundary defined in the geometry. As said in Chapter 3.2.1, our geometry has 5 different boundaries: *inlet1* (air inlet), *inlet2* (water inlet), *walls1* (horizontal capillary walls), *walls2* (vertical capillary walls) and *outlet* (mixture outlet).

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alpha.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet1
    {
      type        inletOutlet;
      inletValue  uniform 1;
    }

    inlet2
    {
      type        inletOutlet;
      inletValue  uniform 0;
    }

    walls1
    {
      type        constantAlphaContactAngle;
      theta0      155;
      limit       gradient;
      value       uniform 0;
    }

    walls2
    {
      type        fixedValue;
      value       uniform 0;
    }

    outlet
    {
      type        zeroGradient;
    }
}

// ************************************************************************* //
```

Fig. A.1 Alpha file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet1
    {
      type        zeroGradient;
    }

    inlet2
    {
      type        zeroGradient;
    }

    walls1
    {
      type        fixedFluxPressure;
      value       $internalField;
    }

    walls2
    {
      type        fixedFluxPressure;
      value       $internalField;
    }

    outlet
    {
      type        fixedValue;
      value       $internalField;
    }
}

// ************************************************************************* //
```

Fig. A.2 Pressure file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet1
    {
        type            fixedValue;
        value           uniform (0.4 0 0);
    }

    inlet2
    {
        type            fixedValue;
        value           uniform (0 -0.4 0);
    }

    walls1
    {
        type            fixedValue;
        value           $internalField;
    }

    walls2
    {
        type            fixedValue;
        value           $internalField;
    }

    outlet
    {
        type            pressureInletOutletVelocity;
        value           $internalField;
    }
}

// ************************************************************************* //
```

Fig. A.3 Velocity file

A.1.2 Constant folder

In the ***g*** file (Fig. A.4) the microgravity conditions are set.

In the ***transportProperties*** file (Fig. A.5) the two phases are defined, writing its corresponding values of kinematic viscosity and density and telling the program that both fluids are Newtonian.

In the ***turbulenceProperties*** file (Fig. A.6) the type of simulation is set to laminar, since the Reynolds number is always smaller than 2300 in this project.

Inside *constant* folder there is also the ***polymesh*** folder, which is automatically created by the command *fluent3DMeshToFoam*. Inside *polymesh* there is all the information related with the mesh created in Ansys, but translated to OpenFOAM language.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];

value           (0 0 0); //Microgravity conditions

// ************************************************************************* //
```

Fig. A.4 Gravity file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (air water);

air
{
    transportModel  Newtonian;
    nu              8.163e-06;
    rho             1.225;
}

water
{
    transportModel  Newtonian;
    nu              1e-06;
    rho             1000;
}

sigma           0.072;

// ************************************************************************* //
```

Fig. A.5 Transport properties file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;

// ************************************************************************* //
```

Fig. A.6 Turbulence properties file

A.1.3 System folder

In the ***controlDict*** file (Fig. A.7) you can define the running parameters of the simulations. For example, you can select the start and end times and the time step, which is called *deltaT*. Setting the field *writeControl* to *adjustableRunTime*, a results folder will be saved each *writeInterval* time (in my case I save the results each 0.0004 s in order to have a smooth representation of the simulation in ParaView).

At the end of the file, the different functions to obtain the post-processing data are written. In my case I use two different functions, one for the pressure probe and another one for the sampling surfaces. First, you have to specify the type of function and the library to which it corresponds. You can also select the write interval in the *writeControl* field. For the probe, you have to specify its location in *probeLocations*. For the sampling surfaces, it is important to set the *operation* field to *areaAverage* if you want to measure the average air fraction value of the surface. Another option is to use *areaIntegrate*, which gives you the equivalent area of air on the surface. The location and direction of the surface can be written in the *pointAndNormalDict* field.

In the ***decomposeParDict*** file (Fig. A.8) you can select the decomposition method of the processor if you want to run the simulation in parallel. In my case, I use the scotch decomposition method with 4 subdomains. For more information about this method, check the source code file:

$FOAM_SRC/parallel/decompose/scotchDecomp/scotchDecomp.C

In the ***setFieldsDict*** file (Fig. A.9) the first half of the horizontal capillary is filled with air. Inside *regions*, the *cylinderToCell* function is used because the geometry is a cylinder (if it was a rectangular cuboid, the function *boxToCell* should be used). The points *p1* and p2 specify the centers of the two circular faces of the cylinder and the scalar *radius* defines its radius.

In the ***fvSchemes*** file (Fig. A.10) the numerical schemes of the simulation are specified.

In the ***fvSolution*** file (Fig. A.11) the solvers, PIMPLE algorithm parameters and relaxation factors are controlled. These last two files are the same as the ones used by Blanca Dalfó in [25]. To know more information about numerical schemes, solvers and algorithms, check the OpenFOAM user guide [24].

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.0512;

deltaT          0.000001;

writeControl    adjustableRunTime;

writeInterval   0.0004;      //save a folder interval

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  no;

maxCo           2;
maxAlphaCo      1;

functions
{
      probes
      {
            type        probes;
            libs        ("libfieldFunctionObjects.so");
            log         true;
            writeControl      timeStep;
            name        probes;
            fields
            (
                  p_rgh
```

```
      );
      interpolationScheme     cellPoint;
      probeLocations
      (
            (0.0105 0 0)
      );
}

surface13
{
      type        surfaceFieldValue;
      libs        ("libfieldFunctionObjects.so");
      log         true;
      writeControl      timeStep;
      writeFields false;
      regionType  sampledSurface;
      name        surface13;
      operation   areaAverage;            //Air fraction average value
      fields            (alpha.air);
      surfaceFormat     dx;
      sampledSurfaceDict
      {
            type        cuttingPlane;
            planeType   pointAndNormal;
            pointAndNormalDict
            {
                  basePoint   (0.013 0 0);
                  normalVector    (1 0 0);
            }
      source            cells;
      interpolate       false;
      triangulate false;
      }
}

surface14
{
      type        surfaceFieldValue;
      libs        ("libfieldFunctionObjects.so");
      log         true;
      writeControl      timeStep;
      writeFields false;
      regionType  sampledSurface;
      name        surface14;
      operation   areaAverage;
      fields            (alpha.air);
      surfaceFormat     dx;
      sampledSurfaceDict
      {
            type        cuttingPlane;
            planeType   pointAndNormal;
            pointAndNormalDict
            {
                  basePoint   (0.014 0 0);
                  normalVector    (1 0 0);
            }
      source            cells;
      interpolate       false;
      triangulate false;
      }
}

surface19
```

```
    {
        type        surfaceFieldValue;
        libs        ("libfieldFunctionObjects.so");
        log         true;
        writeControl      timeStep;
        writeFields false;
        regionType  sampledSurface;
        name        surface19;
        operation   areaAverage;
        fields            (alpha.air);
        surfaceFormat     dx;
        sampledSurfaceDict
        {
            type        cuttingPlane;
            planeType   pointAndNormal;
            pointAndNormalDict
            {
                basePoint   (0.019 0 0);
                normalVector    (1 0 0);
            }
        source            cells;
        interpolate       false;
        triangulate false;
        }
    }

    surface20
    {
        type        surfaceFieldValue;
        libs        ("libfieldFunctionObjects.so");
        log         true;
        writeControl      timeStep;
        writeFields false;
        regionType  sampledSurface;
        name        surface20;
        operation   areaAverage;
        fields            (alpha.air);
        surfaceFormat     dx;
        sampledSurfaceDict
        {
            type        cuttingPlane;
            planeType   pointAndNormal;
            pointAndNormalDict
            {
                basePoint   (0.02 0 0);
                normalVector    (1 0 0);
            }
        source            cells;
        interpolate       false;
        triangulate       false;
        }
    }
}

// ************************************************************************* //
```

Fig. A.7 ControlDict file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 4;   //number of cores of the cluster

method          scotch;

// ************************************************************************* //
```

Fig. A.8 DecomposeParDict file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
      volScalarFieldValue    alpha.air 0
);

regions
(
      cylinderToCell
      {
            p1 (0 0 0);
            p2 (0.01 0 0);
            radius 0.0005;
            fieldValues
            (
                  volScalarFieldValue    alpha.air 1
            );
      }
);


// ************************************************************************* //
```

Fig. A.9 SetFieldsDict file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)  Gauss upwind;
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p_rgh;
    pcorr;
    alpha.air;
}

// ************************************************************************* //
```

Fig. A.10 FvSchemes file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.air.*"
    {
        nAlphaCorr      2;
        nAlphaSubCycles 1;
        cAlpha          1;

        MULESCorr       yes;
        nLimiterIter    5;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }

    "pcorr.*"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-05;
        relTol          0;
    }

    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0.05;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0;
    }
}
```

```
PIMPLE
{
    momentumPredictor no;
    nOuterCorrectors    1;
    nCorrectors         3;
    nNonOrthogonalCorrectors 0;
    cAlpha        1;
    nAlphaCorr          2;
    nAlphaSubcycles     5;
}

relaxationFactors
{
      "U.*"       1;
      "pcorr.*"   1;
}

// ************************************************************************* //
```

Fig. A.11 FvSolution file

A.2. How to run a simulation

In this subchapter, the process of running a simulation in the EETAC cluster is explained. The steps are the following:

1. Copy the case file from the local PC to the cluster using the command:
   scp -r ./OpenFOAM/carlos-6/run/case2 cmoreno@147.83.7.212:/cluster/users/students/cmoreno

2. Run the *setFieldsDict* file using the command:
   setFields

3. Decompose the case in order to run the simulation in parallel:
   decomposePar

4. Run the simulation in parallel using the *nohup* command and save a *.log* file so you can close the simulation window while the process is being executed (the number 4 refers to the number of subprocessors):
   nohup mpirun -np 4 interFoam -parallel > std.log &

5. If you want to stop the simulation before *endTime*, you have to search the *nohup* process id in the list and then kill it:
   ps -ef
   kill <pid number>

6. Reconstruct the subprocessors using the command:
   reconstructPar

7. Delete the subprocessors folders:
   rm -r processor0; rm -r processor1; rm -r processor2; rm -r processor3

8. Copy the case file from the cluster to the local PC using the command:
   scp -r cmoreno@147.83.7.212:/cluster/users/students/cmoreno/case2 ./OpenFOAM/carlos-6/run

A.3. ParaView

In this subchapter, a brief tutorial on how to use ParaView 5.6.0 is done. Once the program is installed, the only thing you have to do in order to visualize the case in ParaView is to type *paraFoam* in the terminal. Then, a window like Fig. A.12 will open.

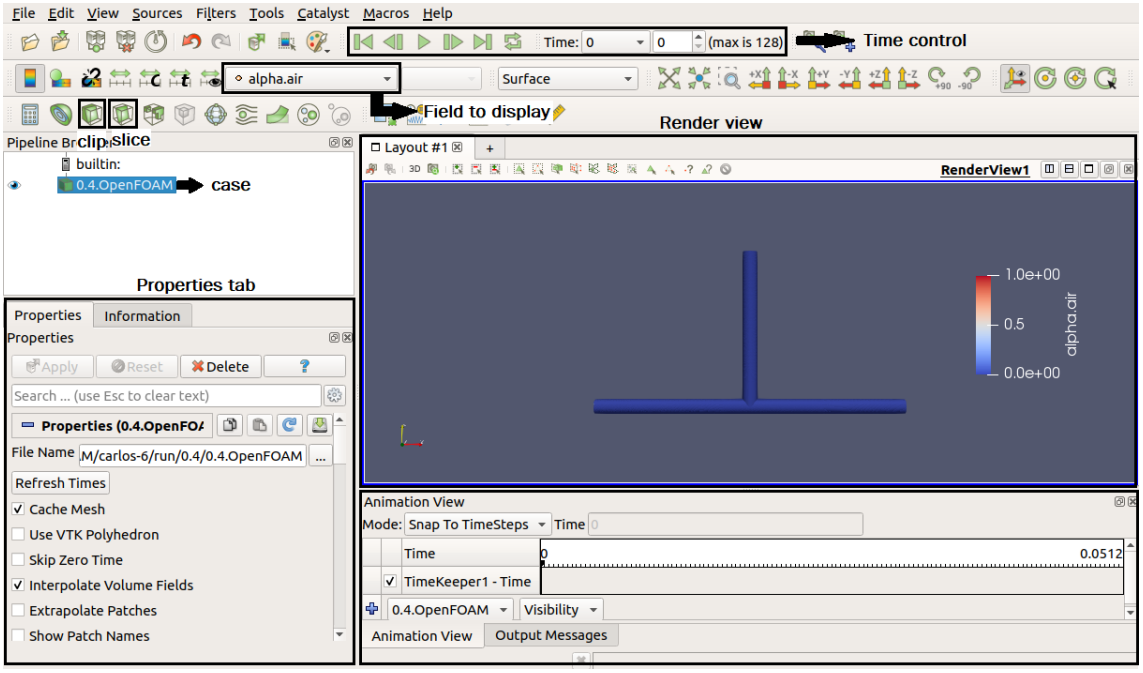

Fig. A.12 ParaView window

In the time control tab, you can select the frame to visualize and also press the play button to run the whole simulation. In this case, a total of 128 frames were saved in steps of 0.0004 s. Below this tab you can select the field to display: alpha, velocity, pressure…

Now, the slice and clip options are going to be explained since they are the tools that I used the most in ParaView. First, to use the slice tool just click on the slice button. Then, on the properties tab you have to select the origin and normal direction of the plane. Since we want to see the inside of the geometry, a cut with origin (0, 0, 0) and normal (0, 0, -1) will be done (-z axis direction); see Fig. A.13. Now you can see the interior of the geometry and analyze the bubble generation process.

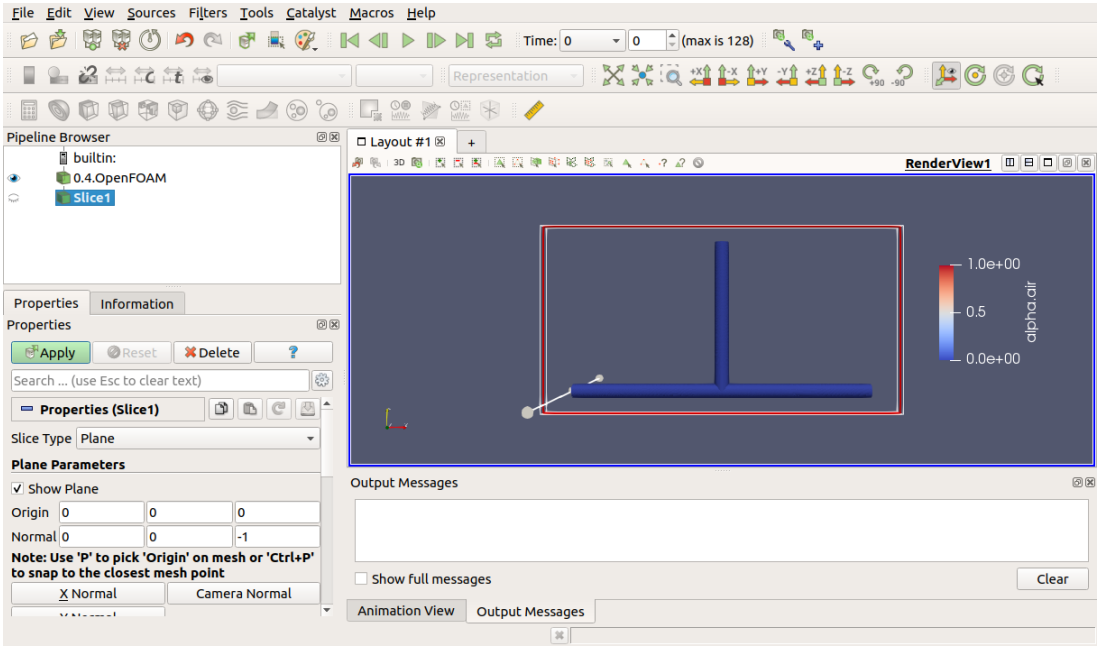

Fig. A.13 Slice tool

To use the clip tool, click the clip button and then, in the properties tab, select the "Scalar" clip type with "alpha.air" as scalar and a value of 0.5. Disable the "invert" option. Then, visualize the original geometry, but set the opacity to 0.2 approximately to "see through" the walls of the T-junction. The result will be something like Fig. A.14. It is the same as the "slice" view but in 3D, which can be useful to generate a video animation.

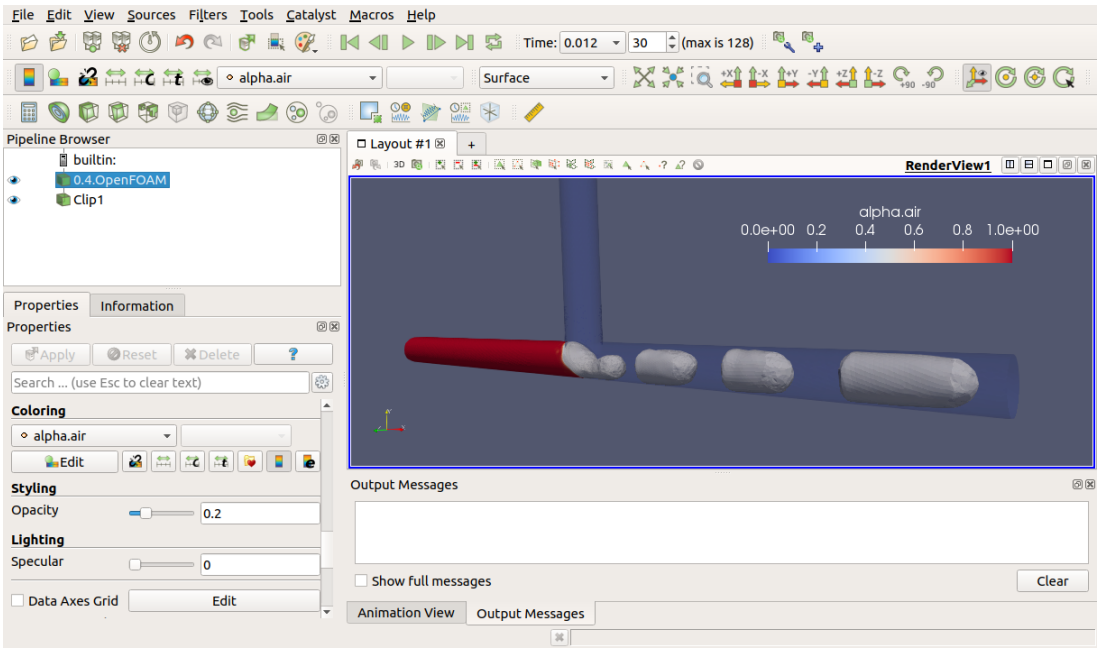

Fig. A.14 Clip tool

To generate the animation, just go to the left upper corner and click “File” → “Save Animation”. Select the save directory for the animation. Then, an options window will show up; see Fig. A.15.

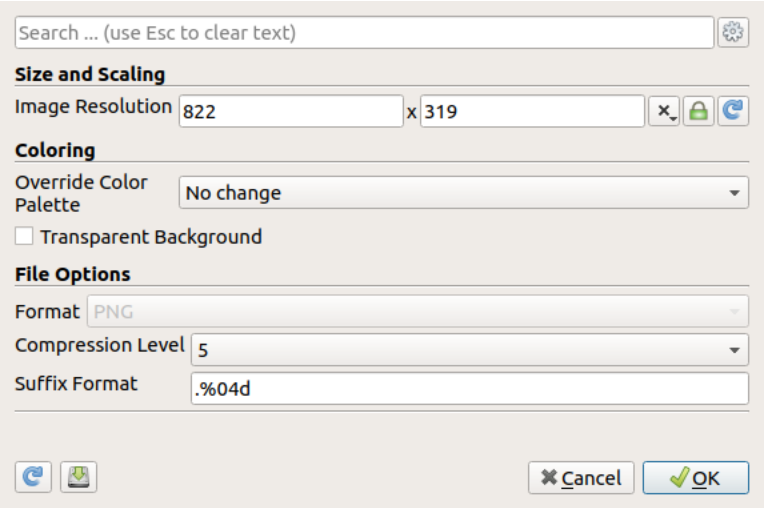

Fig. A.15 Animation options

There you can choose the resolution, the color of the background (it can also be a transparent .png) and the compression level of the images (less compression, better quality). Finally, ParaView will automatically save one image for every frame of the simulation. To create the video, use a video editor like VirtualDub, FFMPEG or OpenShot to merge the images at your desired frame rate and resolution.

APPENDIX B. CALCULATIONS

In this appendix, the analytical solution of the Hagen-Poiseuille flow is derived, since it is necessary to compare it with the simulations of Chapter 4.1.

B.1. Hagen-Poiseuille flow analytical solution

Consider the incompressible flow through a straight circular pipe of radius R. Assuming the fully developed situation, the flow is purely axial, that is, $v_r = v_\theta = 0$ and $v_z \neq 0$ [36]. Axial symmetry is also assumed: $\frac{\partial}{\partial \theta} = 0$. Then, the continuity equation in cylindrical coordinates (Eq. (B.1)) reduces to $\frac{\partial v_z}{\partial z} = 0$, so $v_z = v_z(r)$ only.

$$\frac{1}{r}\frac{\partial}{\partial r}(r v_r) + \frac{1}{r}\frac{\partial v_\theta}{\partial \theta} + \frac{\partial v_z}{\partial z} = 0 \qquad \textbf{(B.1)}$$

Eq. (B.2) shows the Navier-Stokes equation in cylindrical coordinates (z-axis).

$$\rho\left(\frac{\partial v_z}{\partial t} + v_r\frac{\partial v_z}{\partial r} + \frac{v_\theta}{r}\frac{\partial v_z}{\partial \theta} + v_z\frac{\partial v_z}{\partial z}\right) = \rho g - \frac{\partial p}{\partial z} + \mu\nabla^2 v_z \qquad \textbf{(B.2)}$$

The majority of the terms disappear when applying the considered assumptions, resulting in the following linear differential equation:

$$\frac{\mu}{r}\frac{d}{dr}\left(r\frac{dv_z}{dr}\right) = \frac{dp}{dz} = const < 0 \qquad \textbf{(B.3)}$$

Integrating twice and applying the no-slip boundary condition at $r = R$ and finite velocity at $r = 0$, the final solution is obtained:

$$v_z = -\left(\frac{dp}{dz}\right)\frac{1}{4\mu}(R^2 - r^2) \qquad \textbf{(B.4)}$$

As it can be seen, the velocity profile is a paraboloid with maximum at $r = 0$ with value $v_{max} = -\left(\frac{dp}{dz}\right)\frac{R^2}{4\mu}$. A parameter that is important to calculate is the average velocity, which happens to be the injection velocity of the fluid into the pipe. This value can be obtained from the conservation of mass principle [29]:

$$\dot{m} = \rho v_{avg} A = \int_A \rho v_z(r) dA \qquad \textbf{(B.5)}$$

Plugging the expression of the velocity profile into Eq. (B.5) and integrating, the value of the average velocity happens to be half of the maximum velocity:

$$v_{avg} = -\left(\frac{dp}{dz}\right)\frac{R^2}{8\mu} = \frac{v_{max}}{2} \qquad \textbf{(B.6)}$$

So, if the injection velocity of the fluid is $0.5\, m/s$, then the theoretical maximum velocity that can be achieved (at the channel centerline) is $1\, m/s$. This is the value taken as reference to compare it with the exit velocity of the pipe when running the simulations. If the length and the mesh of the cylinder are good enough, then it is expected to get similar values to $1\, m/s$ as maximum velocity.

Another interesting parameter that can be extracted from the analytical solution is the pressure gradient along the pipe, that is, how much the pressure drops for every meter of channel. Isolating $\frac{dp}{dz}$ from Eq. (B.6):

$$\frac{dp}{dz} = \frac{-4\mu}{R^2} v_{max} \qquad \textbf{(B.7)}$$

Substituting water viscosity, channel radius and maximum velocity into Eq. (B.7), a pressure gradient of $-16\, kPa/m$ is obtained. We expect to get a similar value to this one when running the simulations.

APPENDIX C. MATLAB CODE

In this appendix, the full MATLAB code used in this project is shown. First, the code for a single case is presented. This code is the same as the one used by Blanca Dalfó in [25], but with some modifications in order to adjust it to my project scenario. Finally, the code used for extracting the graphs shown in CHAPTER 5 is presented.

C.1. Single case code

```
close all;
clear all;

%% Analysis of the data in surface x = 19 mm
s1data = importdata('surface19.dat');
time1 = cellfun(@str2double, s1data.textdata(6:end, 1));
airfrac1 = cellfun(@str2double, s1data.textdata(6:end, 2));
watfrac1 = 1 - airfrac1;
area = pi*((1e-3)/2)^2;

% Find initial and final points of every bubble
i = 1;
k = 1;
init = true;
val = 0.0035;   % Minimum value of airfrac to consider a bubble
while (i <= length(airfrac1))
    done = false;
    if (airfrac1(i) >= val && init == true)
        points1(k) = i;
        k = k + 1;
        init = false;
        done = true;
    end
    if (airfrac1(i) <= val && init == false && done == false)
        points1(k) = i;
        k = k + 1;
        init = true;
    end
    i = i + 1;
end

% Remove odd numbers
if (rem(length(points1),2) == 1)
    points1(k-1) = [];
end

% Separation between initial and final points
i = 1;
k = 1;
while (i <= length(points1))
    ipoints1(k) = points1(i);
    i = i + 1;
    fpoints1(k) = points1(i);
    i = i + 1;
```

```
    k = k + 1;
end

% Bubble area
i = 1;
while (i <= length(ipoints1))
    a1(i) = trapz(time1(ipoints1(i):fpoints1(i)),
airfrac1(ipoints1(i):fpoints1(i)));
    av1(i) = a1(i)*area;
    i = i + 1;
end

% Remove little bubbles (simulation errors)
i = 1;
while (i <= length(av1))
    if (av1(i) <= 10e-11)
        ipoints1(i) = [];
        fpoints1(i) = [];
        av1(i) = [];
        a1(i) = [];
        i = i - 1;
    end
    i = i + 1;
end

% Bubble entering and exit time
i = 1;
while (i <= length(fpoints1))
    bubbletime1(i,1) = time1(ipoints1(i));
    bubbletime1(i,2) = time1(fpoints1(i));
    i = i + 1;
end

% Bubble frequency
i = 1;
while (i <= length(ipoints1) - 1)
    periode1(i) = time1(ipoints1(i+1)) - time1(ipoints1(i));
    fi1(i) = 1/periode1(i);
    i = i + 1;
end

%% Analysis of the data in surface x = 20 mm
s2data = importdata('surface20.dat');
time2 = cellfun(@str2double, s2data.textdata(6:end, 1));
airfrac2 = cellfun(@str2double, s2data.textdata(6:end, 2));
watfrac2 = 1 - airfrac2;

% Find initial and final points of every bubble
i = 1;
k = 1;
init = true;
while (i <= length(airfrac2))
    done = false;
    if (airfrac2(i) >= val && init == true)
        points2(k) = i;
        k = k + 1;
        init = false;
        done = true;
    end
    if (airfrac2(i) <= val && init == false && done == false)
```

```
        points2(k) = i;
        k = k + 1;
        init = true;
    end
    i = i + 1;
end

% Remove odd numbers
if (rem(length(points2),2) == 1)
    points2(k-1) = [];
end

% Separation between initial and final points
i = 1;
k = 1;
while (i <= length(points2))
    ipoints2(k) = points2(i);
    i = i + 1;
    fpoints2(k) = points2(i);
    i = i + 1;
    k = k + 1;
end

% Bubble area
i = 1;
while (i <= length(ipoints2))
    a2(i) = trapz(time2(ipoints2(i):fpoints2(i)),
airfrac2(ipoints2(i):fpoints2(i)));
    av2(i) = a2(i)*area;
    i = i + 1;
end

% Remove little bubbles (simulation errors)
i = 1;
while (i <= length(av2))
    if (av2(i) <= 10e-11)
        ipoints2(i) = [];
        fpoints2(i) = [];
        av2(i) = [];
        a2(i) = [];
        i = i - 1;
    end
    i = i + 1;
end

% Bubble entering and exit time
i = 1;
while (i <= length(fpoints2))
    bubbletime2(i,1) = time2(ipoints2(i));
    bubbletime2(i,2) = time2(fpoints2(i));
    i = i + 1;
end

% Bubble frequency
i = 1;
while (i <= length(ipoints2) - 1)
    periode2(i) = time2(ipoints2(i+1)) - time2(ipoints2(i));
    fi2(i) = 1/periode2(i);
    i = i + 1;
end
```

```
%% Parameter calculations
% Bubble velocity and volume
i = 1;
while (i <= length(ipoints2))
    tbetween(i) = bubbletime2(i,1) - bubbletime1(i,1);    %time from
surf1 to surf2
    Ug(i) = 1e-3/tbetween(i);
    if (Ug(i) < 0)
        Ug(i) = [];
    end
    volume(i) = av1(i) * Ug(i);
    i = i + 1;
end

% Alternative volume calculation
Usg = 0.4;  %air injection velocity
Qg = area*Usg;  %volumetric flow rate (constant)
i = 1;
while (i < length(ipoints2))
    volumeAlt1(i) = Qg / fi1(i);
    volumeAlt2(i) = Qg / fi2(i);
    i = i + 1;
end

% Bubble length
i = 1;
while (i <= length(Ug))
    tbubble(i) = bubbletime1(i,2) - bubbletime1(i,1);     %time from
entering surf1 to exiting surf1
    L(i) = Ug(i)*tbubble(i);
    i = i + 1;
end

% Arrays of the selected final num bubbles to do the mean and standard
% deviation calculations
num = 8;    % last number of bubbles
i = 1;
while i <= num
    fi1new(i) = fi1(end-(i-1));
    fi2new(i) = fi2(end-(i-1));
    Ugnew(i) = Ug(end-(i-1));
    Lnew(i) = L(end-(i-1));
    Vnew(i) = volume(end-(i-1));
    i = i + 1;
end
fi1new = fliplr(fi1new);
fi1mean = mean(fi1new);
fi2new = fliplr(fi2new);
fi2mean = mean(fi2new);
Ugnew = fliplr(Ugnew);
Ugmean = mean(Ugnew);
Lnew = fliplr(Lnew);
Lmean = mean(Lnew);
Vnew = fliplr(Vnew);
Vmean = mean(Vnew);

% Standard deviation calculation
for i = 1:length(fi1new)
    f1(i) = (fi1new(i) - fi1mean)^2;
    f2(i) = (fi2new(i) - fi2mean)^2;
    u(i) = (Ugnew(i) - Ugmean)^2;
```

```
    l(i) = (Lnew(i) - Lmean)^2;
    v(i) = (Vnew(i) - Vmean)^2;
end
df1 = sqrt(sum(f1)/(num - 1));
df2 = sqrt(sum(f2)/(num - 1));
du = sqrt(sum(u)/(num - 1));
dl = sqrt(sum(l)/(num - 1));
dv = sqrt(sum(v)/(num - 1));

% Save the parameters in a file
case4_mean = [fi1mean, fi2mean, Ugmean, Lmean, Vmean];
case4_dev = [df1, df2, du, dl, dv];
save('case4_mean', 'case4_mean');
save('case4_dev', 'case4_dev');

%% Analysis of probe data
probedata = importdata('pressure');
timep = cellfun(@str2double, probedata.textdata(5:end, 1));
pressure = cellfun(@str2double, probedata.textdata(5:end, 2));

% Extract the data of one pressure cycle
case4_tcycle = timep(14399:17199);
case4_pcycle = pressure(14399:17199);

% Save it
save('case4_tcycle', 'case4_tcycle')
save('case4_pcycle', 'case4_pcycle')

%% PLOTS

limit = val*ones(length(time1),1);

%19 mm surface
figure
plot(time1, airfrac1, 'black')
hold on
plot(time1, limit, 'red')
title('x = 19 mm surface')
xlabel('Time (s)')
ylabel('Air fraction')

% 20 mm surface
figure
plot(time2, airfrac2, 'black')
hold on
plot(time1, limit, 'red')
title('x = 20 mm surface')
xlabel('Time (s)')
ylabel('Air fraction')

% Pressure
figure
plot(timep, pressure, 'k')
title('Pressure at probe x = 10.5 mm')
xlabel('Time (s)')
ylabel('Gauge pressure (Pa)')

% Frequency
figure
```

```
plot(fi1, 'black')
hold on
plot(fi2, 'blue')
title('Bubble frequency')
xlabel('Bubble')
ylabel('Frequency (Hz)')
legend('19 mm surface', '20 mm surface')
xlim([1 length(fi1)])

% Length
figure
plot(L*1000, 'black')
title('Bubble length')
xlabel('Bubble')
ylabel('Lentgh (mm)')
xlim([1 length(L)])

% Velocity
figure
plot(Ug, 'black')
title('Bubble velocity')
xlabel('Bubble')
ylabel('Velocity (m/s)')
xlim([1 length(Ug)])

% Volume
figure
plot(volume, 'black')
hold on
plot(volumeAlt1, 'blue')
hold on
plot(volumeAlt2, 'green')
title('Bubble volume')
xlabel('Bubble')
ylabel('Volume (m^3)')
legend('19 & 20 mm surface', '19 mm alt. method', '20 mm alt. method',
'Location', 'southeast')
xlim([1 length(volume)])
```

C.2. Results code

```
clear all;
close all;

% Load files
load('case6_mean.mat')
load('case6_dev.mat')
load('case6_tcycle.mat')
load('case6_pcycle.mat')
load('case5_mean.mat')
load('case5_dev.mat')
load('case5_tcycle.mat')
load('case5_pcycle.mat')
load('case4_mean.mat')
load('case4_dev.mat')
load('case4_tcycle.mat')
load('case4_pcycle.mat')
```

```
load('case3_mean.mat')
load('case3_dev.mat')
load('case3_tcycle.mat')
load('case3_pcycle.mat')
load('case2_mean.mat')
load('case2_dev.mat')
load('case2_tcycle.mat')
load('case2_pcycle.mat')

% Create parameters arrays
f = [0, case2_mean(1), case3_mean(1), case4_mean(1), case5_mean(1),
case6_mean(1)];
u = [0, case2_mean(3), case3_mean(3), case4_mean(3), case5_mean(3),
case6_mean(3)];
l = [0, case2_mean(4), case3_mean(4), case4_mean(4), case5_mean(4),
case6_mean(4)];
v = [0, case2_mean(5), case3_mean(5), case4_mean(5), case5_mean(5),
case6_mean(5)];

df = [case2_dev(1), case3_dev(1), case4_dev(1), case5_dev(1),
case6_dev(1)];
du = [case2_dev(3), case3_dev(3), case4_dev(3), case5_dev(3),
case6_dev(3)];
dl = [case2_dev(4), case3_dev(4), case4_dev(4), case5_dev(4),
case6_dev(4)];
dv = [case2_dev(5), case3_dev(5), case4_dev(5), case5_dev(5),
case6_dev(5)];

% Data to create dimensionless numbers
Usg = [0, 0.2, 0.3, 0.4, 0.5, 0.6];
Usl = [0, 0.2, 0.3, 0.4, 0.5, 0.6];
Ca = Usl*1e-3/0.072;
diam = 1e-3;
area = pi*(diam^2)/4;

%% Frequency plot
% Dimensionless frequency (divide by experimental fsat) and
dimensionless
% Usg (divide by experimental fsat/ao)
f_adim = [0];
df_adim = [];
Usg_adim = [0];

% Experimental data
fsat = [0, 175, 275, 400, 600, 850];
ao = [0, 705, 788, 854, 1610, 2714];

f_adim(2) = f(2)/fsat(2);
f_adim(3) = f(3)/fsat(3);
f_adim(4) = f(4)/fsat(4);
f_adim(5) = f(5)/fsat(5);
f_adim(6) = f(6)/fsat(6);

df_adim(1) = df(1)/fsat(2);
df_adim(2) = df(2)/fsat(3);
df_adim(3) = df(3)/fsat(4);
df_adim(4) = df(4)/fsat(5);
df_adim(5) = df(5)/fsat(6);

Usg_adim(2) = Usg(2)/(fsat(2)/ao(2));
```

```
Usg_adim(3) = Usg(3)/(fsat(3)/ao(3));
Usg_adim(4) = Usg(4)/(fsat(4)/ao(4));
Usg_adim(5) = Usg(5)/(fsat(5)/ao(5));
Usg_adim(6) = Usg(6)/(fsat(6)/ao(6));

% Create the function f_adim = 1 - exp(-Usg_adim)
Usg_adim_fun = 0:0.001:Usg_adim(end);
for i = 1:length(Usg_adim_fun)
    f_adim_fun(i) = 1 - exp(-Usg_adim_fun(i));
end

figure()
plot(Usg_adim_fun, f_adim_fun, 'k')
hold on
errorbar(Usg_adim(2:6), f_adim(2:6), df_adim, 'o', 'MarkerEdgeColor',
'k', 'Color', 'k');
xlabel('$\bar{U}_{SG}$','Interpreter','Latex','fontsize',14)
ylabel('$\bar{f}$','Interpreter','Latex','fontsize',14)

%% Extra frequency plot
Usg_fun = 0:0.0005:Usg(end);
for i = 1:length(Usg_fun)
    f_fun2(i) = fsat(2)*(1-exp(-ao(2)*Usg_fun(i)/fsat(2)));
    f_fun3(i) = fsat(3)*(1-exp(-ao(3)*Usg_fun(i)/fsat(3)));
    f_fun4(i) = fsat(4)*(1-exp(-ao(4)*Usg_fun(i)/fsat(4)));
    f_fun5(i) = fsat(5)*(1-exp(-ao(5)*Usg_fun(i)/fsat(5)));
    f_fun6(i) = fsat(6)*(1-exp(-ao(6)*Usg_fun(i)/fsat(6)));
end

figure()
p2 = plot(Usg_fun, f_fun2, 'k');
hold on
p3 = plot(Usg_fun, f_fun3, 'k');
hold on
p4 = plot(Usg_fun, f_fun4, 'k');
hold on
p5 = plot(Usg_fun, f_fun5, 'k');
hold on
p6 = plot(Usg_fun, f_fun6, 'k');
hold on
e6 = errorbar(Usg(6), f(6), df(5), 'o', 'MarkerEdgeColor', 'k',
'Color', 'k');
hold on
e5 = errorbar(Usg(5), f(5), df(4), 'v', 'MarkerEdgeColor', 'k',
'Color', 'k');
hold on
e4 = errorbar(Usg(4), f(4), df(3), '^', 'MarkerEdgeColor', 'k',
'Color', 'k');
hold on
e3 = errorbar(Usg(3), f(3), df(2), 'd', 'MarkerEdgeColor', 'k',
'Color', 'k');
hold on
e2 = errorbar(Usg(2), f(2), df(1), 'p', 'MarkerEdgeColor', 'k',
'Color', 'k');
xlim([0 0.65])
xlabel('{U}_{SG} [m/s]', 'fontsize', 12)
ylabel('Frequency [Hz]', 'fontsize', 12)
leg = legend([e6 e5 e4 e3 e2], '8.3', '6.9', '5.6', '4.2', '2.8',
'location', 'northwest');
legend boxoff
title(leg, 'Ca [10^{-3}]');
```

```
%% Velocity plot
Um = Usg + Usl;
velocity_eq =
fittype('Co*Um','coefficients',{'Co'},'dependent',{'u'},'independent',
{'Um'});
u_fit = fit(Um', u', velocity_eq);

figure()
plot(u_fit, '-k')
hold on
errorbar(Um(2:6), u(2:6), du, 'o', 'MarkerEdgeColor', 'k', 'Color',
'k');
xlim([0 1.3])
ylim([0 1.6])
xlabel('U_{M} [m/s]')
ylabel('U_{G} [m/s]')
legend('off')

%% Length plot
Lb = l/diam;    % non dimensional length
Lb_dev = dl/diam;   % non dimensional length std. deviation

figure()
errorbar(Ca(2:6), Lb(2:6), Lb_dev, 'o', 'MarkerEdgeColor', 'k',
'Color', 'k');
set(gca, 'Xscale', 'log', 'Yscale', 'log', 'Fontsize', 14)
xlabel('Ca [10^{-3}]')
xlim([2e-3, 10e-3])
xticks([2 3 4 5 6 7 8 9 10]*1e-3)
xticklabels({'2', '3', '4', '5', '6', '7', '8', '9', '10'})
ylabel('$\bar{L}_{B}$','Interpreter','Latex','fontsize',14)
ylim([1, 4])

%% Volume plot
Vb = v/(area*diam);    % non dimensional volume
Vb_dev = dv/(area*diam);    % non dimensional volume std. deviation

figure()
errorbar(Ca(2:6), Vb(2:6), Vb_dev, 'o', 'MarkerEdgeColor', 'k',
'Color', 'k');
set(gca, 'Xscale', 'log', 'Yscale', 'log')
xlabel('Ca [10^{-3}]')
xlim([2e-3, 10e-3])
xticks([2 3 4 5 6 7 8 9 10]*1e-3)
xticklabels({'2', '3', '4', '5', '6', '7', '8', '9', '10'})
ylabel('$\bar{V}_{B}$','Interpreter','Latex','fontsize',14)
ylim([0.5, 3])

%% Volume polydispersity index
Ip = [];
for i = 1:length(dv)
    Ip(i) = 100*(dv(i)/v(i+1));
end

figure()
scatter(Ca(2:6), Ip, 'filled', 'k')
title('Volume polydispersity index', 'fontsize', 12)
xlabel('Ca', 'fontsize', 12)
ylabel('{I}_{p} [%]', 'fontsize', 14)
```

```
%% Frequency polydispersity index
Ip = [];
for i = 1:length(df)
    Ip(i) = 100*(df(i)/f(i+1));
end

figure()
scatter(Ca(2:6), Ip, 'filled', 'k')
title('Frequency polydispersity index', 'fontsize', 12)
xlabel('Ca', 'fontsize', 12)
ylabel('{I}_{p} [%]', 'fontsize', 14)

%% Pressure plot
timep6 = (0:1:length(case6_pcycle)-1)/1200;  % non dimensional cycle
time
timep5 = (0:1:length(case5_pcycle)-1)/2030;
timep4 = (0:1:length(case4_pcycle)-1)/2800;
timep3 = (0:1:length(case3_pcycle)-1)/3000;
timep2 = (0:1:length(case2_pcycle)-1)/5650;

figure()
plot(timep6, case6_pcycle, 'Linewidth', 1.5)
hold on
plot(timep5, case5_pcycle, 'Linewidth', 1.5)
hold on
plot(timep4, case4_pcycle, 'Linewidth', 1.5)
hold on
plot(timep3, case3_pcycle, 'Linewidth', 1.5)
hold on
plot(timep2, case2_pcycle, 'Linewidth', 1.5)
xlabel('$\bar{t}_{cycle}$','Interpreter','Latex','fontsize',14)
ylabel('Gauge pressure [Pa]')
leg = legend('0.6', '0.5', '0.4', '0.3', '0.2');
legend boxoff
title(leg, 'U_{SG}, U_{SL}');
```