Dispersed Two-Phase Flow Simulation and Parameter Optimisation

by

Yapi Donatien Achou

Thesis

for the degree of

Master in Applied Mathematics

(Master i anvendt Matematikk)

Faculty of Mathematics and Natural Sciences
University of Oslo

March 2016

Det matematisk-naturvitenskapelige fakultet
Universitetet i Oslo

Acknowledgments

This Master's thesis was worked on during the course of an exciting year at Simula Research Laboratory. I would like to thank the staff and fellow students for creating a productive and enjoyable working environment. I would especially like to thank my two thesis supervisors Xing Cai and Knut Morken for their guidance along the way. There were many questions and technical issues, and I greatly appreciate the generosity which Xing Cai showed me with regards to his availability and time.
The process of writing this thesis has been a journey which allowed me to discover myself. I am grateful for the opportunity and the challenges that I have encountered along the way.

Yapi Donatien Achou, Oslo, March 2016

Contents

Chapter 1

Introduction

Modeling of multiphase flow is of great importance in industry. The mixed flow of oil, gas and water in a system of pipes has to be well-understood to avoid rupture of the pipes due to pressure fluctuation. The proportion of steam in a two-phase flow of steam and water in a geothermal well determines the amount of electrical energy that can be generated. Multiphase flow is also ubiquitous among the various features of our environment, whether one considers sediment transport, pollutant transport in air or in a river. In addition, critical medical and biological flow such as blood flow is multiphase flow.

The ability to predict the behaviour of multiphase flow requires a good understanding of the physical processes involved. There are two different ways of predicting the behaviour of multiphase flow: through experimental analysis or mathematical models. A mathematical model of multiphase flow is expressed as a set of differential equations. One such model is the two-fluid model derived by conditionally averaging the single-phase flow equation. The averaging process introduces extra terms that account for the complexity of the flow and the loss of information induced by this process. For example, the Reynolds stress is an extra term introduced for each phase during the averaging process, and it accounts for turbulence shear stress in the fluids. Besides averaging a model equation for a two-phase flow, there are other methods that directly solve the differential equation(s) without extra assumptions or extra terms. These methods are the so-called interface capturing methods or free surface methods. In this approach the interface between the two phases is part of the solution process. The shape and position of the interface are tracked, and topological changes such as merging, breaking etc. are captured during the solution process.

The solution process of a mathematical equation representing a multiphase flow process is an algorithmic procedure. First the flow domain is discretised, meaning decomposed into discrete points, areas or volumes for one,

two or three-dimensional problem respectively. The discretisation of the flow domain is followed by the discretisation of the continuous differential equations into a discrete set of algebraic equations. This process is achieved by approximating the derivatives of the continuous equations through numerical methods such as finite volume, finite element, or finite difference methods. Once the form of the discretised equation is known, one can either use direct or iterative methods to derive the solutions of the differential equations.
Various numerical parameters are needed to control the accuracy and stability of the discretised equation solutions. One such parameter is the underrelaxation factor, which is used to improve the solution of pressure, velocity and other fields. Other parameters include the mesh size, the time step, the Courant number, etc. It is important to note that there is a trade-off between accuracy and the time it takes to compute the solution(s). For example, by refining the mesh for accuracy, the size of the algebraic equations increases and the computational cost increases: this is a trade-off that must be done.

In computational fluid dynamics (CFD), the finite volume method is widely used as discretisation tool. One of its advantages is that it preserves the conservation of flow quantities, which is a cornerstone of fluid mechanics. One CFD tool that uses the finite volume method is OpenFoam, which is open-source and widely used both in research and industry. In recent years OpenFoam has become more popular than ever. Its capability can be for example compared to its commercial counterpart ANSYS Fluent. It has an extensive range of features to solve complex fluid flow involving multiphase flow, chemical reaction, turbulence, heat transfer, solid mechanics, electromagnetism [20], to only cite a few. Popular Computational Fluid Mechanics algorithms such as SIMPLE and PISO are implemented in OpenFoam. These algorithms are equipped with numerous parameters in order to control the accuracy of the solution. The solver also provides utilities to control and analyse the solution process. Residual information can be accessed from a log file generated from the solver.
The questions a modeler must ask are: can we solely rely on the residuals provided by the solver to judge the accuracy of the solution? How can one choose the numerical parameters involved in the PISO and SIMPLE algorithm to effectively harness their power? The answers to these questions are not trivial. An understanding of the implemented algorithm in terms of its strengths and weaknesses is required and assessing numerical errors with a reliable method is necessary.

To that effect, the objective of this thesis is to use OpenFoam to model a dispersed two-phase flow. The two-fluid model for fully dispersed turbulence flow and the volume of fluid method applied to a single rising bubble

in liquid water will be applied to:

- Identify the numerical parameters affecting the solution of a two-phase flow process. Both laminar and turbulent flow are considered.
- Once the parameters are identified, explore how they affect the solution in terms of accuracy and CPU time. The accuracy of the solutions is assessed by using a variant of the Richardson extrapolation method: the Grid Convergence Index (GCI) method.
- Derive an optimisation strategy to effectively choose the numerical parameters based on accuracy and CPU time.
- Test the result for two different applications: a laminar model for rising of a single air bubble in liquid water and turbulence model of a fully dispersed two-phase flow.

The rest of this thesis is divided as follows: chapter Two gives an introduction to multiphase flow in general and two-phase flow in particular. Direct numerical simulation and the two-fluid method are briefly presented. In chapter Three a detailed exposition of the two-fluid equation is given. The latter is used in chapter Five to model a fully dispersed two-phase flow. Chapter Four gives an exposition of the volume of fluid method, which is used in chapter Five to model the rising of a single air bubble in liquid water. Chapter Five is devoted to the parameter optimisation strategy.

Chapter 2

Introduction To Multiphase Flow

A multiphase flow can be defined as a flow in which more than one phase occurs. A phase is a state of matter and can be solid, gas or liquid. From sediment displacement in a river, pollutant particles flowing in air, flow of oil and gas in a pipeline, to combustion reaction in an engine, multiphase flow is all around us. When two phases are involved we refer to the flow as two-phase flow. In multiphase flow in general and two-phase flow in particular there is a strong influence or interaction between the phases separated by an interface. The interface, defined as the boundary separating the phases, captures the physics of the flow. For example, in liquid-gas flow where a single bubble of air is rising in liquid water, the interface undergoes large deformations. The mathematical modelling approach must therefore consider the interaction and influence of each phase and take the deformation of the interface into account. In this chapter we present basic concepts involved in two-phase flow and present two modelling approaches.

This chapter is organized as follows: we start by defining basic concepts of two-phase flow in section One. Section Two and Three deal with modelling approaches: in section Two we present the direct numerical simulation approach and in section Three the average modelling approach.

2.1 Introduction to two-phase flow

The most common two-phase flows involve liquid-liquid, liquid-gas, solid-liquid, solid-gas and solid-solid. In a two-phase flow involving solid-liquid/ solid-gas, the solid phase is in the form of particles, such as dust particles in air. The particles are commonly called the dispersed phase while the liquid/air is called the continuous phase. Although not treated in this thesis, pipe flow is fundamental in understanding multiphase flow. In fact, pipe flow can be used to define some basic concepts that can be generalised to all multiphase flow processes. To this end we introduce some two-phase flow terminology taken from multiphase pipe flow.

Given a pipe with cross-sectional area A and a flow rate Q we define the superficial velocity of phase k as the velocity of that phase as if it was the only phase present in the pipe:

$$U_{kS} = \frac{Q_k}{A} \tag{2.1}$$

where U_{sk} is the superficial velocity of phase k and Q_k is the mass flow rate of phase k. The bulk velocity U_k of phase k is the actual velocity of that phase in the pipe:

$$U_k = \frac{Q_k}{A_k} \tag{2.2}$$

where A_k is the cross-sectional area of phase k. The mixture velocity is the sum of the superficial velocity of the two phases. Assuming a simultaneous flow of gas and liquid, $k = L, G$ and the mixture velocity would be:

$$U_{mix} = U_{LS} + U_{GS}. \tag{2.3}$$

The slip ratio is the ratio of the bulk velocity of the phases U_G, U_L :

$$S = \frac{U_G}{U_L} \tag{2.4}$$

and the slip velocity is the difference velocity of the phases:

$$U_S = |U_G - U_L|. \tag{2.5}$$

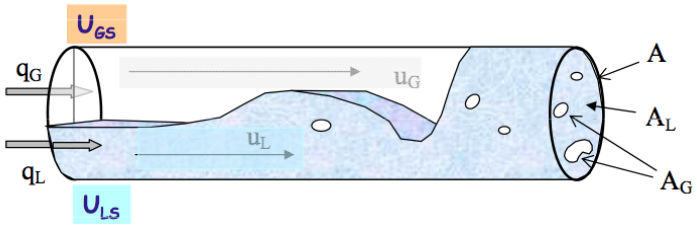

Figure 2.1: Vertical pipe flow of gas and liquid

The void fraction or phase fraction α_k for phase k is one of the most important parameters used to characterize two-phase flows. The void fraction is defined as the proportion of phase k in the two-phase flow:

$$\alpha_k = \frac{A_k}{A} \tag{2.6}$$

The void fraction is sometimes interpreted as a probabilistic quantity: the probability of finding phase k in a given region of space at a given time. Numerous important parameters such as the two-phase mixture density,

the relative average velocity of the phases, and the two-phase viscosity are defined based on the void fraction.
The changes in topology of the flow structure caused by the variation of the phase velocities are inherent to two-phase flow. This topology changes are called flow regimes. The main flow regimes for horizontal pipe flow are: stratified, dispersed bubble, annular (wavy), slug and elongated bubble. For vertical flow we have the same flow regime exempt that there is no stratified flow regime; instead we have an additional churn flow regime. The different flow regimes for pipe flow are illustrated in Figures 2.2 and 2.3.

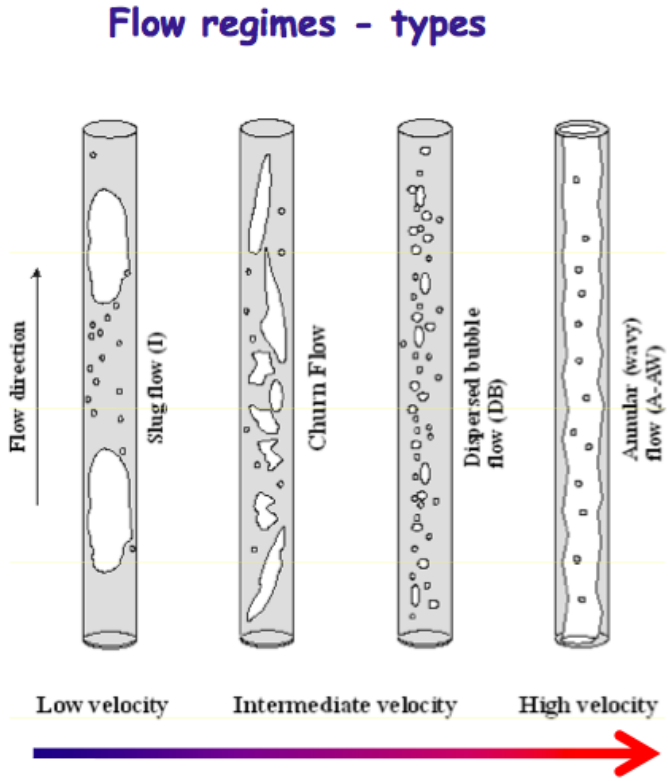

Figure 2.2: Vertical pipe flow regime

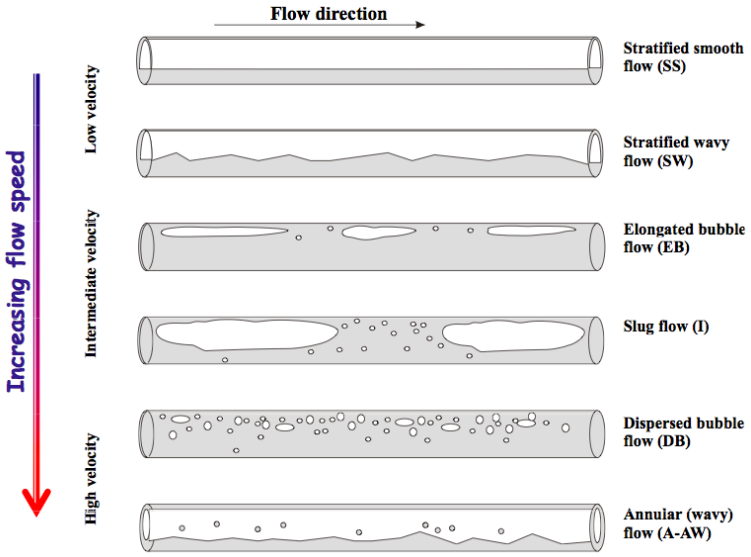

Figure 2.3: horizontal pipe flow regime

The flow regimes encountered in two-phase pipe flow can also be observed in nature. For example, dispersed two-phase flow can be seen in pollutant transport in rivers. In this case the solid pollutant particles represent the dispersed phase, while liquid water represents the continuous phase. The same is true for pollutant transport in the atmosphere, where air represents the continuous phase.

Another important type of two-phase flow is the so-called free surface flow, which can be seen as a separate flow regime, illustrated in Figure 2.4.
A free surface is the surface of a fluid that is subject to both zero perpen-

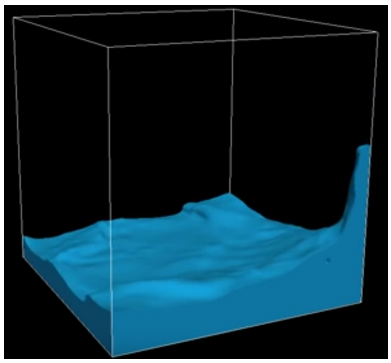

Figure 2.4: Example of free surface

dicular normal stress and parallel shear stress [33]. Surface waves on the ocean generated by the interaction between the ocean and air currents, or a water droplet falling in air are examples of free surface flows. Surface waves

on the ocean can be seen as an example of stratified wavy flow. In such flows the interface or boundary between the flows plays an important role in the physics of the flow. The interface, also called free surface boundary, is a part of the solution process. The topology and the position of the free surface boundary must be known during the flow process. Depending on the types of two-phase flow we can apply different modelling approaches. In the next section we cover different types of modelling methods for the two-phase flow phenomenon.

2.2 Direct Numerical Simulation (DNS)

For reference of this section see [4]. Explicit references will be used otherwise. In the Direct Numerical Simulation approach the Navier-Stokes equations are solved and the topology of the interface between the flows is treated as part of the solution. In practical simulations, the mesh size must be fine enough to resolve a large temporal and spatial scale associated to the interface such as turbulence length, time scale, or the size of each phase involved in the two-phase flow. This implies huge amounts of computational power for large Reynolds numbers, which is the prevalent case for turbulence effects. In DNS, the position of the interface must be tracked. The different techniques used to this aim are collected under the umbrella name interface capturing methodology or free surface solution procedure. Figures 5.10 and 2.6 show the simulation of a breaking column of water using an interface capturing methodology implemented in the InterFoam solver in OpenFoam.

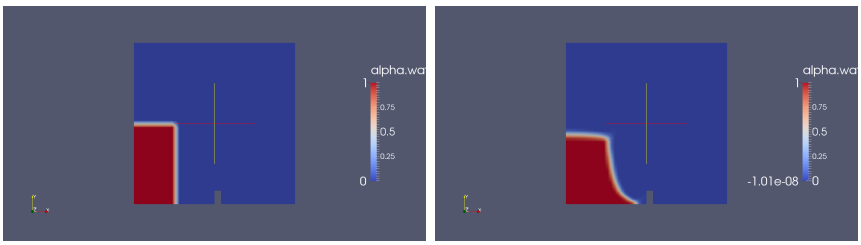

Figure 2.5: Interface capturing method implemented in InterFoam solver in OpenFoam. Simulation of a column of water breaking at $t = 0, 0.1s$

The DNS approach is mainly used in the research community and deals in general with laminar flow. Since the Navier-Stokes equations are solved directly without additional modelling assumptions, DNS allows a deep understanding and insight to physical phenomena such as breaking, merging of air bubbles or water droplets, surface tension effects in the flow, deformation of fluid particles in shear flow etc. It is therefore a great research tool to understand and validate the existing models or develop new models.

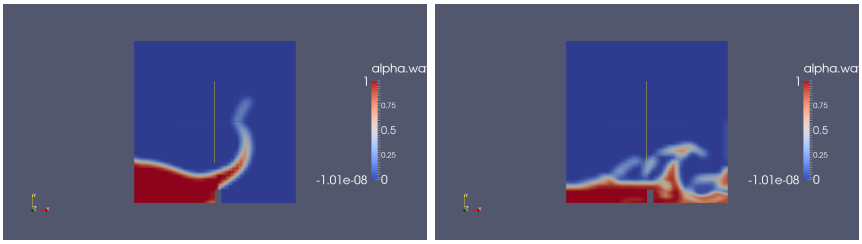

Figure 2.6: Interface capturing method implemented in InterFoam solver in OpenFoam. Simulation of a column of water breaking at $t = 0.25, 0.8s$

In engineering applications however, most practical two-phase flows are turbulence flows. In these flows, the DNS is not widely used due to the large computational power necessary to resolve the wide range of scales.
Turbulence flows are dominated by eddies with macro scales and micro scales. The macro scales are associated to the boundary of the flow. For example, in practical industrial flows the length scale can be of the order of kilometres. The micro scale is called the Kolmogorov scale and it is universal: it only depends on the kinematic viscosity ν and the average rate of dissipation of turbulence kinetic energy ε. These scales (time, length and velocity) are very small and the kinetic turbulence energy dissipates into heat at these scales. The resolution of the small scales associated to turbulence effects involves finer resolutions, which in turn requires huge computational power. For practical engineering problems, this is expensive. In such a case an alternative approach is to model the two-phase flow by computing average properties of the flow such as pressure, velocity, phase fraction, turbulence kinetic energy, rate of dissipation etc. This modelling approach is based on the fact that turbulence fluctuation and the two-phase flow effect affect the mean flow properties.

2.3 Average modelling approach

In the averaging approach, the microscopic conservation equations (mass, momentum and energy) are averaged by using different averaging procedures. Three averaging procedures can be distinguished: time, volume and ensemble averaging, defined respectively by

$$\phi = \frac{1}{T}\int_T \phi \mathrm{d}t \tag{2.7}$$

$$\phi = \frac{1}{V}\int_V \phi \mathrm{d}V \tag{2.8}$$

$$\phi = \frac{1}{N}\sum \phi_r \tag{2.9}$$

where ϕ is a given scalar or vector quantity. V, T are the volume and time domains respectively and N is the number of realization in the domain.

The computational power needed in the DNS approach can be reduced greatly. However this comes with a price: the averaging process induces a loss of information. This lost information must therefore be recovered by introducing additional terms in the averaged conservation equations. These additional terms are expressed in terms of known mean properties or quantities involved in the flow. For example, averaging the Navier-Stokes equations produces the Reynolds stress which appears in the turbulence model.

There are two main approaches when it comes to modelling two-phase flow with averaging techniques: the Dispersed Phase Element (DPE) approach (Euler-Lagrange) and the two-fluid approach (Euler-Euler).

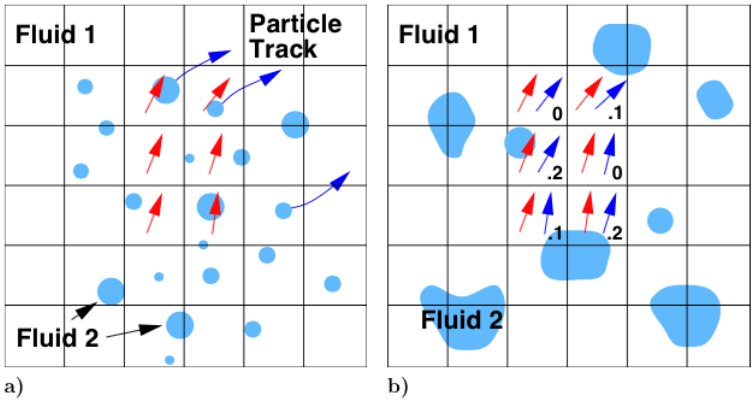

Figure 2.7: a) DPE approach and b) two-fluid approach

In the DPE approach one phase is dispersed into the other. For example, in gas-liquid two-phase flow where gas bubbles are mixed with liquid water, the continuous phase is the water, while the dispersed phase is represented by the gas bubble particles (the Dispersed Phase Element). In this formulation, each dispersed phase element is tracked through the conservation equation (momentum equation) expressed in a Lagrangian frame of reference, where each dispersed phase element requires its own conservation equation.
On the other hand, the conservation equations for the continuous phase are expressed in an Eulerian frame where the fluid properties are given as a function of space and time in an absolute inertia frame. This can seen in Figure 2.7a.
Since each DPE requires it own conservation equations, these give an accurate description of all the properties of the DPEs as an ensemble. However, a high phase fraction of the DPEs increases the computational power. Therefore the DPE approach is well suited for low phase fractions of DPEs where

an accurate description of the DPEs properties is required.

In the two-fluid model both phases are described in an Eulerian frame of reference where each phase is treated as a continuum. Each phase possesses its own conservation equation. The amount of each phase in space and time is represented by the phase fraction, which expresses the probability that a certain phase can be found at a given position in time. Since each phase is treated as a continuum, the transfer of momentum between the phases is accounted for by an extra term called the interphase momentum transfer term, which represents the forces that each phase exerts on the other. These forces are the drag force, lift and virtual mass. The averaging process of the momentum equation introduces Reynolds stress, which is resolved by introducing a turbulence model.

Chapter 3

Two Fluid Model

3.1 Single phase conservation equations

Let V be a connected volume of a collection of fluid particles called material volume. Such material volume has constant mass. This property is important in deriving the conservation equation of mass. Let θ be some intensive property of the fluid. A property is called intensive if it can be defined at a point within the material volume V. Pressure, temperature and density are examples of intensive properties of the fluid. In general, θ is a function of the temporal and spatial dimensions:

$$\theta = \theta(\mathbf{X}, t),$$

and the surface S of the material volume V changes with time.
One interesting question is how the intensive property θ changes in time. To answer this question we must take the time derivative of the total amount of the intensive property θ. This time derivative must include the changes related to θ and the surface S wrapping θ. This is formally stated by the Reynolds transport theorem:

$$\frac{d}{dt}\int_V \theta \mathrm{d}V = \int_V \left(\frac{\partial \theta}{\partial t} + \nabla \cdot (\theta v)\right) \mathrm{d}V \tag{3.1}$$

The temporal change of θ is represented by $\frac{\partial \theta}{\partial t}$ while $\nabla \cdot (\theta v)$ represents the flux of θ through the surface bounding the material volume. Setting $\theta = \rho$ in (3.1) and using the assumption that the mass of the material volume $\int_V \theta \mathrm{d}V$ is constant in time, the left hand side of (3.1) vanishes and we get:

$$\int_V \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v)\right) \mathrm{d}V = 0,$$

and for arbitrary V we obtain:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = 0, \tag{3.2}$$

which is the mass conservation equation. Here v is the velocity of the fluid. We can use the same procedure to derive a generic transport equation for mass, momentum and for the energy equation. This time let Ψ be an extensive property. A property of the fluid is called extensive if it is proportional to the extension of the fluid. Examples of extensive properties are mass, volume and kinetic energy.
Taking the time derivative of the intensive property and applying Reynolds conservation equation we obtain:

$$\frac{d}{dt}\int_V \rho\Psi \mathrm{d}V = \int_V \left(\frac{\partial \rho\Psi}{\partial t} + \nabla\cdot(\rho\Psi v)\right)\mathrm{d}V. \tag{3.3}$$

The time rate of change of the total amount of the extensive property Ψ is also equal to the sum of all forces acting of the volume V:

$$\frac{d}{dt}\int_V \rho\Psi \mathrm{d}V = \int_S \mathbf{J}\cdot\mathbf{n}\mathrm{d}S + \int_V \rho\phi \mathrm{d}V. \tag{3.4}$$

Using (3.3) and (3.4) and rearranging the terms we obtain:

$$\int_V \frac{\partial \rho\Psi}{\partial t}\mathrm{d}V + \int_V \nabla\cdot(\rho\Psi v)\mathrm{d}V - \int_S \mathbf{J}\cdot\mathbf{n}\mathrm{d}S - \int_V \rho\phi\mathrm{d}V. \tag{3.5}$$

The surface integral in (3.5) can be transformed to a volume integral by using the divergence theorem:

$$\int_S \mathbf{J}\cdot\mathbf{n}\mathrm{d}S = \int_V \nabla\cdot\mathbf{J}\mathrm{d}V$$

to obtain:

$$\int_V \left(\frac{\partial \rho\Psi}{\partial t} + \nabla\cdot(\rho\Psi v) - \nabla\cdot\mathbf{J} - \rho\phi\right)\mathrm{d}V = 0,$$

and for arbitrary V we get the genetic conservation equation:

$$\frac{\partial \rho\Psi}{\partial t} + \nabla\cdot(\rho\Psi v) - \nabla\cdot\mathbf{J} - \rho\phi = 0. \tag{3.6}$$

Conserved quantity	Ψ	$\mathbf{J}$	ϕ
Mass	1	0	0
Momentum	$\mathbf{v}$	$\mathbf{T}$	$-g$
Energy	e	$\mathbf{T}\cdot\mathbf{v}-\mathbf{q}$	$g\cdot\mathbf{v}+r$

Table 3.1: Values of parameters in generic conservation equation (3.6)

Table 3.1 gives the different values of Ψ, $\mathbf{J}$, and ϕ for the conservation of mass momentum and energy. $\mathbf{T}$ is the stress tensor, g the gravity force, $\mathbf{v}$ the velocity of the fluid, $\mathbf{q}$ the heat flux and r the distributed heat source per unit mass.

3.2 Conservation equation for two fluid equation

In most applications of two-phase turbulence flows, the average form of the conservation equations is solved. These average conservation equations describe the mean properties of the flow, such as pressure, velocity etc. They are derived by multiplying the generic conservation equation for single phase flow given by (3.6) by a phase function defined by:

$$X_\varphi(\mathbf{x},t) = \begin{cases} 1 & for \quad \mathbf{x} \quad in \quad phase \quad \varphi \\ 0 & otherwise, \end{cases} \tag{3.7}$$

before applying standard averaging techniques [4]. For incompressible fluids, a conditional averaging process is applied to derive the momentum and continuity equation for each phase φ [4]:

$$\frac{\partial \alpha_\varphi \overline{\mathbf{U}}_\varphi}{\partial t} + \nabla \cdot (\alpha_\varphi \overline{\mathbf{U}}_\varphi \overline{\mathbf{U}}_\varphi) + \nabla \cdot (\alpha \overline{\mathbf{R}}_\varphi^{eff}) = \frac{\alpha_\varphi}{\rho_\varphi} \nabla \overline{\mathbf{p}} + \alpha_\varphi \overline{\mathbf{g}} + \frac{\overline{\mathbf{M}}_\varphi}{\rho_\varphi} \tag{3.8}$$

$$\frac{\partial \alpha_\varphi}{\partial t} + \nabla \cdot (\overline{\mathbf{U}}_\varphi \alpha_\varphi) = 0. \tag{3.9}$$

α_φ	phase fraction of phase φ
φ	liquid or gas
$\overline{\mathbf{U}}_\varphi$	average velocity of phase φ
$\overline{\mathbf{R}}_\varphi^{eff}$	Reynolds stress plus viscous stress for phase φ
$\overline{\mathbf{p}}$	average pressure of the flow
$\overline{\mathbf{M}}_\varphi$	averaged interphase momentum transfer term for phase φ
ρ_φ	density of phase φ
$\mathbf{g}$	gravity force

Table 3.2: Description of parameters in equation (3.8) and (3.9)

Table 3.2 describes the different terms in the conservation equation of mass and momentum, i.e. equation (3.8) and (3.9).
The averaged interphase momentum transfer term $\overline{\mathbf{M}}_\varphi$ represents mass transfer between the phases. In the case of dispersed flow, $\overline{\mathbf{M}}_\varphi$ is derived by summing up all the forces acting on a dispersed phase element [4]:

$$\frac{\overline{\mathbf{M}}_a V}{\alpha_a} = \mathbf{F}_d + \mathbf{F}_l + \mathbf{F}_{vm},$$

where V is the volume of the dispersed phase element (DPE), $\mathbf{F}_d$ is the drag force, $\mathbf{F}_l$ is the lift, $\mathbf{F}_{vm}$ is the virtual mass. The subscript a designates the dispersed phase, while b stands for the continuous phase. For most practical computations the different forces are given by [4]:

$$\mathbf{F}_d = \frac{1}{2}\rho_b A C_d |\overline{\mathbf{U}}_b - \overline{\mathbf{U}}_a|(\overline{\mathbf{U}}_b - \overline{\mathbf{U}}_a) \tag{3.10}$$

$$\mathbf{F}_d = \rho_b C_l V(\overline{\mathbf{U}}_b - \overline{\mathbf{U}}_a) \times (\nabla \times \overline{\mathbf{U}}_b) \tag{3.11}$$

$$\mathbf{F}_d = \rho_b C_{vm} V \left(\frac{\partial \overline{\mathbf{U}}_b}{\partial t} + \overline{\mathbf{U}}_b \cdot \nabla \overline{\mathbf{U}}_b - \frac{\partial \overline{\mathbf{U}}_a}{\partial t} - \overline{\mathbf{U}}_a \cdot \nabla \overline{\mathbf{U}}_a \right) \tag{3.12}$$

where C_d, C_l and C_{vm} are the dead, lift and virtual mass coefficient respectively.

Alternatively, the momentum exchange term $\overline{\mathbf{M}}_\varphi$ can be decomposed in terms of the drag, lift and virtual momentum exchange contributions:

$$\overline{\mathbf{M}}_\varphi = \overline{\mathbf{M}}_{\varphi,drag} + \overline{\mathbf{M}}_{\varphi,lift} + \overline{\mathbf{M}}_{\varphi,vm} \tag{3.13}$$

where these are defined in terms of the dispersed phase a as

$$\overline{\mathbf{M}}_{\varphi,drag} = \frac{3}{4}\alpha_b \left(\alpha_b \frac{C_{D,a}\rho_b}{d_a} + \alpha_a \frac{C_{D,b}\rho_a}{d_b} \right) |\overline{\mathbf{U}}_b - \overline{\mathbf{U}}_a| \overline{\mathbf{U}}_b \tag{3.14}$$

$$\overline{\mathbf{M}}_{\varphi,lift} = \alpha_a \alpha_b (\alpha_b C_{a,lift} \rho_b + \alpha_a C_{a,lift} \rho_a)(\overline{\mathbf{U}}_b - \overline{\mathbf{U}}_a) \times (\nabla \times \overline{\mathbf{U}}_b) \tag{3.15}$$

$$\overline{\mathbf{M}}_{\varphi,vm} = \alpha_a \alpha_b (\alpha_b C_{a,vm} \rho_b + \alpha_a C_{a,vm} \rho_a) \left(\frac{\partial \overline{\mathbf{U}}_b}{\partial t} + \overline{\mathbf{U}}_b \cdot \nabla \overline{\mathbf{U}}_b - \frac{\partial \overline{\mathbf{U}}_a}{\partial t} - \overline{\mathbf{U}}_a \cdot \nabla \overline{\mathbf{U}}_a \right) \tag{3.16}$$

The term $\overline{\mathbf{R}}_\varphi^{eff}$ representing the Reynolds stress is given by [4]:

$$\overline{\mathbf{R}}_\varphi^{eff} = -\nu_\varphi^{eff} (\nabla \overline{\mathbf{U}}_\varphi + \nabla \overline{\mathbf{U}}_\varphi^T - \frac{2}{3}\mathbf{I}\nabla \cdot \overline{\mathbf{U}}_\varphi) + \frac{2}{3}\mathbf{I}k_\varphi \tag{3.17}$$

In discretising the momentum and the continuity equation in the next section, the Reynolds stress will be explained in detail.

3.3 Drag force models

A particle moving through air experiences a resisting or drag force which is proportional to the square of its velocity or its velocity, depending on the value of the Reynolds number. This phenomenon was studied experimentally by Newton and Coulomb. Coulomb established experimentally that the drag force of a particle in a moving fluid depends partly on the velocity and on the square of the velocity [7]. By measuring the terminal velocity of a falling particle with respect to different particle radius in free-fall, Newton observed that the drag force was proportional to the square of the terminal velocity [6]. The theoretical study of a particle moving in a fluid started with the analytical work of Stokes [4]. Stokes studied the flow around a spherical particle where inertial forces are small compared to viscous forces. This flow is called Stokes flow or creeping flow. This is typical for flows with small velocity. By neglecting the convection part of the Navier-Stokes equations, Stokes derived the drag force around an "ascending or descending" spherical particle in a fluid as [7]

$$-F = 6\pi\mu\rho a V$$

Where μ, ρ, a are the fluid viscosity, density and radius respectively. V is the terminal velocity defined by Stokes as

$$V = \frac{2g}{9\mu}\left(\frac{\sigma}{\rho} - 1\right)a^2$$

where g is the acceleration of gravity constant and σ is the surface tension. We recall that the terminal velocity is the maximum velocity attained by an object as it moves through a fluid. From the work of Newton, Coulomb and Stokes, we see that the drag force varies with the characteristic length (radius) of the particle immersed in the moving fluid and the properties of the fluid.

For a two-phase flow, the drag force also depends on other factors, such as the fluid properties characterised by Eötvös number ε_0 and Mortons number M_0, and the density and viscosity ratios between the continuous phase and the dispersed phase (the bubble) [4]:

$$\varepsilon_0 = \frac{g_{eff}|\Delta\rho|d^2}{\sigma}$$

$$M_0 = \frac{g_{eff}\mu_c^4|\Delta\rho|}{\rho_c^2\sigma^3}$$

$$\varrho = \frac{\rho_d}{\rho_c}$$

$$\kappa = \frac{\mu_d}{\mu_c}$$

where

$$\Delta\rho = \rho_d - \rho_c$$

$$g_{eff} = g - \frac{\partial U_c}{\partial t} + U_c \cdot \nabla.$$

The subscripts c and d stand for continuous and dispersed phase. When a fluid particle (bubble) is set in motion in a fluid (water), the bubble is subject to different kinds of motion: a primary upward motion due to buoyancy forces acting on the bubble, circulation inside the fluid due to momentum transfer from the continuous phase to the bubble and a deformation of the bubble shape depending on the Eötvös number ε_0, see figure 3.1.

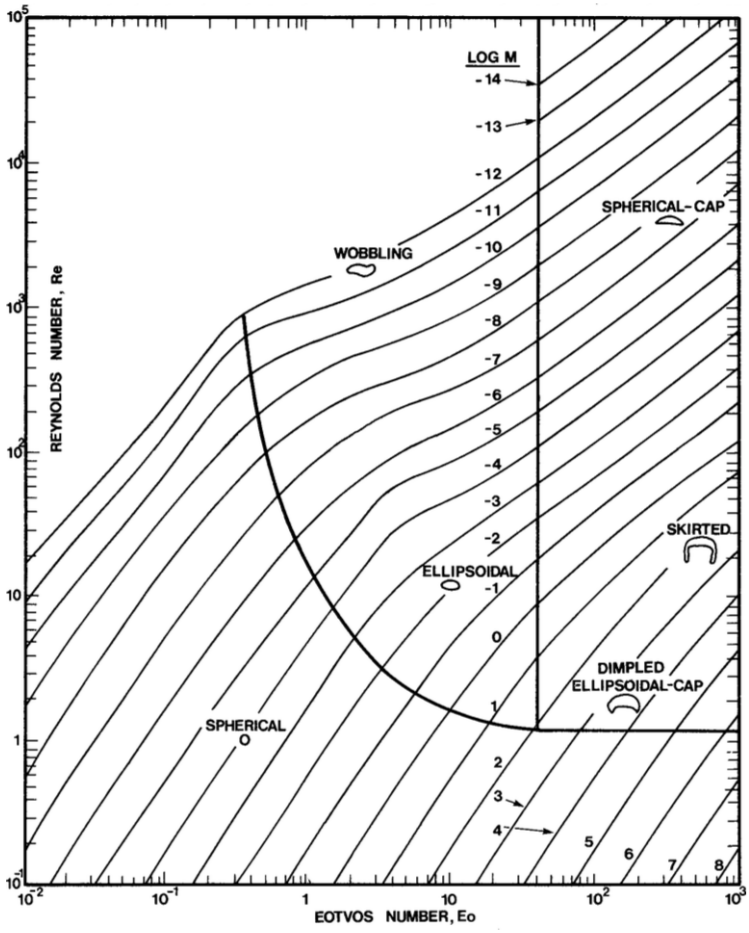

Figure 3.1: Affect of Eötvös number and Reynolds number on bubble shape

For a very low Morton and Eötvös number (around 3), a bubble in a fluid with low viscosity encounters an unstable or "wobbling" motion and the bubble shape is significantly deformed [4], see figure 3.1. This can be explained as follows: a rising bubble in a fluid must do work on the continuous phase. The work done is equal to the rising velocity times the buoyancy force. From the conservation of energy or work-energy principle, the net work is equal to the change in kinetic energy. When viscous forces are dominant the energy is completely damped or dissipated through laminar viscous dissipation [4]. This means that the kinetic energy is converted into internal energy (meaning that the fluid is heated). When the viscosity of the fluid is low, inertia forces dominate and some of the energy is dissipated through vortex shedding. The fluid passing the bubble creates an oscillatory flow inducing a low pressure zone around the downstream side of the bubble, which then moves toward the low pressure zone, creating the "wobbling" secondary motion.

3.4 Lift

A particle moving in a fluid can be subject to a force perpendicular to the direction of the flow. This force is called lift. It was observed by Poiseuille while studying the movement of blood in capillaries [15]. His observation was validated by Segre' and Silberberg in an experiment using a flow apparatus consisting of a vertical tube, an optical scanning device and an electronics counting device [12, 13]. For spherical bubbles the lift coefficient C_L is positive, and the lift force acts in direction of decreasing liquid velocity [9]. It was established experimentally and numerically that the lift force changes direction if the bubble is deformed considerably [9, 10, 11]. By using a single bubble rising in a shear flow of glycerol water the lift coefficient was derived [9, 10] as

$$C_L = \begin{cases} \min(0.288\tanh(0.121R_e), f(E0_\perp)) & \text{if } E0_\perp < 4 \\ f(E0_\perp) & \text{if } 4 < E0_\perp < 10 \\ -0.27 & \text{if } E0_\perp > 10 \end{cases}$$

and is valid for

$$-5.5 \leq \log_{10}(M0) \leq -2.8 \quad , 1.39 \leq E0 \leq 5.74 \quad 0 \leq R_e \leq 10$$

Where $M0$ is the Morton number, R_e is the Reynolds number of the bubble and

$$f(E0_\perp) = 0.00105E0_\perp^3 - 0.0159E0_\perp + 0.474.$$

The modified Eötvös number is given by

$$E0_\perp = \frac{g(\rho_L - \rho_G)d_\perp^2}{\sigma}$$

with $d_\perp$ being the maximum horizontal extension of the bubble given by [Wellek-1996]

$$d_\perp = d_B \left(1 + 0.163E0^{0.757}\right)^{\frac{3}{2}}$$

where d_B is the diameter of the bubble. With the present parameters the lift changes direction when the bubble size is 5.8 mm [9].

The causes of lift force on particles are diverse. In a shear flow, that is a flow where adjacent layers of fluid move at different speed, inertia and walls effect are the cause of lift force [14]. Saffman derived theoretically the lift force in this case as

$$F = \frac{81.2\mu V a^2 k^{\frac{1}{2}}}{\nu^{\frac{1}{2}}},$$

assuming that the particle is spherical and moves through a very viscous fluid with velocity V relative to a uniform simple shear. μ and ν are the viscosity and the kinematic viscosity, respectively. For a very weak shear flow it was shown that a cylindrical object can experience a lift force about 16 time larger than the lift force experienced in a large shear flow [5]. This result proves that lift forces are not only induced by shear forces.

3.5 Turbulence model

The turbulence model is given by the $k-\varepsilon$ turbulence model [1, 4] for the continuous phase b. Turbulence on the dispersed phase is accounted for by scaling the dispersed phase turbulence viscosity:

$$\frac{\partial k_b}{\partial t} + (\overline{\mathbf{U}}_b \cdot \nabla)k_b - \nabla \cdot \left(\frac{\nu_b^{eff}}{\sigma_k}\nabla k_b\right) = P_b - \varepsilon_b + S_k \tag{3.18}$$

$$\frac{\partial \varepsilon_b}{\partial t} + (\overline{\mathbf{U}}_b \cdot \nabla)\varepsilon_b - \nabla \cdot \left(\frac{\nu_b^{eff}}{\sigma_\varepsilon}\nabla \varepsilon_b\right) = \frac{\varepsilon_b}{k_b}(C_1 P_b - C_2 \varepsilon_b) + S_\varepsilon \tag{3.19}$$

k is the turbulence kinetic energy per unit mass and ε is the amount of energy dissipated per unit time. Here b denotes the continuous phase. The production of kinetic energy P_b is given by

$$P_b = 2\nu_b^{eff}(\nabla \overline{\mathbf{U}}_b \cdot dev(\nabla \overline{\mathbf{U}}_b + (\nabla \overline{\mathbf{U}}_b)^T)).$$

The effective viscosity of the continuous phase ν_b^{eff} is given by

$$\nu_b^{eff} = \nu_b + \nu^t = \nu_b + C_\mu \frac{k_b^2}{\varepsilon_b}$$

The source term S_k and S_ε are given respectively by [4]

$$S_k = \frac{2k_b\alpha_a A_d(C_i - 1)}{\rho_b} + \frac{A_d\nu^t\nabla\alpha_a}{\rho_b\sigma_a\alpha_b} \cdot \overline{\mathbf{U}}_r$$

$$S_\varepsilon = \frac{2C_3\varepsilon_b\alpha_a A_d(C_i - 1)}{\rho_b}$$

The coefficient $C_i = C_t$ where C_t is the turbulence response coefficient given by

$$C_t = \frac{\mathbf{U}'_a}{\mathbf{U}'_b}$$

where $\mathbf{U}'_b$ and $\mathbf{U}'_b$ are the r.m.s of fluctuation in velocity of the continuous and dispersed phase, respectively. The rest of the coefficients are given in table 3.3 below.

C_μ	C_1	C_2	C_3	σ_k	σ_ε	σ_α
0.09	1.44	1.92	1.0	1.0	1.3	1.0

Table 3.3: Coefficient in the $k - \varepsilon$ model

Turbulence is an important aspect of most fluid flow. However, there is no exact definition of turbulence. At first glance, turbulence flow can be characterised by the Reynolds number: the Reynolds number measures the magnitude of inertia forces and viscous forces. When viscous forces are predominant such that the Reynolds number is below a critical value Re_c, the flow is smooth. Adjacent layers of the fluid slide pass each other in an ordering fashion with no disruption between the layers. The fluid particles travel along well ordered stream lines. If the boundary condition does not change with time, the flow is steady [1]. This flow regime is called laminar flow. However, when the Reynolds Number is above the critical value Re_c, inertia forces are predominant and the flow properties are characterised by random and chaotic behaviour. For this reason, the velocity is decomposed into fluctuating (u') and mean part (U) according to the Reynolds decomposition:

$$u(t) = U + u(t)'$$

This has led to define turbulence flow in terms of characteristics such as *randomness and chaotic*, hight *diffusivity*, large Reynolds number, highly *dissipative*, three-dimensional and unsteady, etc.

Despite being chaotic and random, turbulence flows are governed by Navier-Stokes equations and comprise a spectrum of different whirls called turbulence eddies. The eddies define the macro structure of turbulence flow. Corresponding to each eddy is a length, time and velocity scale. Kinetic

energy is transferred from one scale to the other through a process called energy cascade, illustrated in figure 3.2.

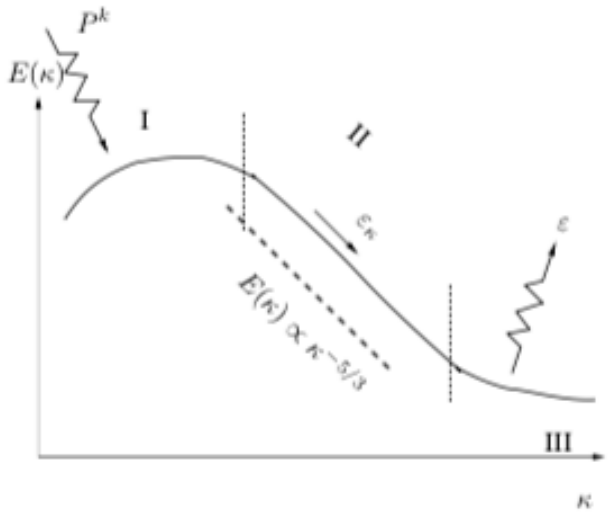

Figure 3.2: Energy cascade process

In figure 3.2 part 1 represents the larger scales. These scales are of the order of the flow geometry and extract kinetic energy from the mean flow. At these scales the Reynolds number is large. This implies that viscous effects are negligible and due to hight inertia forces, rapid mixing or turbulence *diffusion* occurs at these scales. After a time scale comparable to the time scale of the mean flow, the larger eddies breaks up to smaller eddies and transfer most of their kinetic energy to the intermediate scales called inertial scales represented by 2 in figure 3.2. The same process will continue and the inertial scales will transfer their kinetic energy to the smaller scales or micro scales 3. At the smallest scales called the Kolmogorov scales, viscous forces are large and the kinetic energy is dissipated into thermal energy at the rate ε. This will end the cascade process. Let $\mathcal{L}$, $\mathcal{T}$ and $\mathcal{V}$ be the length, time and velocity scale of the Kolmogorov scale. These scales are dominated by viscous force and dissipation taking place. Therefore these scales can be expressed in terms of viscosity ν and rate of dissipation ε. Using dimensionless analysis we can write:

$$\mathcal{V} = \nu^a \varepsilon^b$$

$$\left[\frac{m}{s}\right] = \left[\frac{m^2}{s}\right]\left[\frac{m^2}{s^3}\right]$$

from which we get two equations: one for m and one for s:

$$1 = 2a + 2b$$

$$-1 = -a - 3b$$

when solved gives $a = b = \frac{1}{4}$. Applying the same principle we get

$$\mathcal{V} = (\nu\varepsilon)^{\frac{1}{4}}$$

$$\mathcal{L} = \left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}}$$

$$\mathcal{T} = \left(\frac{\nu}{\varepsilon}\right)^{\frac{1}{2}}$$

The Reynolds number of the Kolmogorov scales is then

$$\begin{aligned} R_e &= \frac{\mathcal{L}\mathcal{V}}{\nu} \\ &= \frac{\left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}} (\nu\varepsilon)^{\frac{1}{4}}}{\nu} \\ &= 1. \end{aligned}$$

This implies that at the micro scales, viscous forces completely dominate.

Chapter 4

Volume Of Fluid Method

In the perspective of fluid mechanics, an interfacial flow or free surface boundary flow can be defined as a flow in which two or more fluids share a surface forming a common boundary between the adjacent fluid regions. Interfacial flow is commonly encountered in multiphase flow, such as droplets of oil in liquid water or droplets of water in steam. In industrial processes, during a casting operation, the interface between the covering slag and the liquid steel in the casting mould can strongly affect the quality of the final product [16]. In biology interfacial flow processing is used to produce a collagen substrate on which culture cells can align [38]. In interfacial flow, surface tension plays an import role. For example, when an air bubble rises in liquid water, the surface tension force strongly affects the shape of the bubble and can cause breaking of the bubble. The larger the surface tension force the less the bubble deforms.

The numerical treatment of interfacial flow must take into account the changing interface between the fluid regions. It must be able to compute and track the interface but also cope with the topological changes such as breaking and merging. Among many numerical methods for treating interfacial flow, the most prominent are marker and cell, the level set method and the volume of fluid method [16]. This chapter is organised as follows: in section 4.1 an overview of the main numerical methods for interfacial flow is discussed. In section 4.2 the volume of fluid method is covered.

4.1 Interface capturing method for two-phase flow

In the numerical computation of interfacial flow, the numerical method must be able to sharply define and track the interface to allow accurate boundary conditions. There are two main numerical frameworks for interfacial flow: direct method or surface method and indirect method or volume method. In surface methods, the focus is placed on the interface, while volume methods focus on the entire flow domain. An example of surface method is the level

set method. This class of methods was developed by Osher and Sethian [42].

Level set method
The level set method defines the interface Γ between fluid 1 and fluid 2 by

$$\Gamma(X,t) = \{X; \Phi(X,t) = 0\} \tag{4.1}$$

where Φ is a smooth function (at least Lipschitz continuous), which can be viewed as an indicator function. Γ is said to be the zero level set of Φ. The function Φ indicates the position of the interface because

$$\begin{cases} \Phi > 0 & \text{in fluid 1} \\ \Phi < 0 & \text{in fluid 2} \\ \Phi = 1 & \text{at the interface} \end{cases}$$

The indicator function Φ measures the distance away from the interface and represents the solution of the convection equation

$$\frac{\partial \Phi}{\partial t} = v|\nabla \Phi| \tag{4.2}$$

where v is the normal velocity component at the interface defined by

$$v = \mathbf{v} \cdot \frac{\nabla \Phi}{|\nabla \Phi|} \tag{4.3}$$

where $\mathbf{v} = (v, u)$, u being the normal velocity on the interface and u being the tangential velocity component. The interface is captured for all later time values t by locating the set $\Gamma(t)$ for which $\Phi(X,t) = 0$ [39]. This allows topological changes such as breaking and merging to be well-defined [39]. It can be noted that equation (4.2) is non-conservative, and therefore Φ is a non-conservative quantity [40]. This implies that the level set will have a tendency to induce numerical diffusion.

Marker and cell method
The method was initially developed for time dependent motion of a viscous, incompressible fluid in an Eulerian staggered grid system, and it used the finite element method as discretisation tool [43, 40]. After discretising the computational domain into cells, the cells are marked with Lagrangian virtual particles [40]. The initial coordinates of each particle are stored and their subsequent coordinates are computed according to the latest velocity field by linear interpolation between the cells [43, 40]. A cell with no marker particle contains no fluid. A cell with marker, adjacent to an empty cell is considered to be on the interface, and all other cells are considered to be filled with fluid [43].

4.2 Volume of fluid

The volume of fluid method can be seen as a variant of the marker and cell method. Rather than marking the computational domain, a volume fraction γ defined by

$$\begin{cases} \gamma = 1 & \text{in fluid 1} \\ \gamma = 0 & \text{in fluid 2} \\ 0 < \gamma < 1 & \text{at the interface} \end{cases}$$

is used to track the interface. If γ_1 is the volume fraction of fluid 1 and u, v are the x and y velocity components in a cartesian coordinate system then the phase fraction of fluid 2 is given by

$$\gamma_1 = 1 - \gamma_2$$

and γ_1 can be computed by solving the partial differential equation [18]

$$\frac{\partial \gamma_1}{\partial t} + u\frac{\partial \gamma_1}{\partial x} + v\frac{\partial \gamma_1}{\partial y} = 0. \tag{4.4}$$

As opposed to the marker and cell method, the VOF method is less computationally expensive since there is no need to store information about each cell in the computational domain. The information about the interface between the fluids is retrieved by solving equation (4.4). In a Lagrangian coordinate system or mesh, the solution of equation (4.4) remains constant in each mesh cell. Thus γ serves as a flag to identify cells that are filled with fluid or empty [18]. In a Lagrangian-Eulerian mesh the solution γ_1 of (4.4) is a step function: 1 in fluid 1, 0 in fluid 2 and between 0 and 1 at the interface. In numerical computations it is common to identify the interface with $\gamma_1 = 0.5$ [20]. A correct computation of γ_1 will ensure the sharpness of the interface while an incorrect computation will lead to a smearing of the interface.

Standard finite element approximation of (4.4) will lead to a smearing of the quantity γ_1 [18]. This is due to the fact that standard finite element approximations are not conservative and equation (4.4) is written in a non-conservative form. However, since γ_1 is a step function, a flux approximation will preserve its discontinuous nature [18]. To achieve a flux approximation of γ_1 equation (4.4) must be reformulated to a conservative form by using the continuity equation. For an incompressible fluid the continuity equation can be expressed as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{4.5}$$

Multiplying equation (4.5) by γ_1 gives

$$\gamma_1 \frac{\partial u}{\partial x} + \gamma_1 \frac{\partial v}{\partial y} = 0. \tag{4.6}$$

Now we can rewrite equation (4.4) as

$$\frac{\partial \gamma_1}{\partial t} + u\frac{\partial \gamma_1}{\partial x} + \gamma_1 \frac{\partial u}{\partial x} - \gamma_1 \frac{\partial u}{\partial x} + v\frac{\partial \gamma_1}{\partial y} + \gamma_1 \frac{\partial v}{\partial y} - \gamma_1 \frac{\partial v}{\partial y} = 0$$

and using the product rule for derivatives we get:

$$\frac{\partial \gamma_1}{\partial t} + \frac{\partial (u\gamma_1)}{\partial x} - \gamma_1 \frac{\partial u}{\partial x} + \frac{\partial (v\gamma_1)}{\partial y} - \gamma_1 \frac{\partial v}{\partial y} = 0. \tag{4.7}$$

Adding equation (4.6) to (4.7) we get

$$\frac{\partial \gamma_1}{\partial t} + \frac{\partial (u\gamma_1)}{\partial x} + \frac{\partial (v\gamma_1)}{\partial y} = 0. \tag{4.8}$$

Equation (4.8) is conservative. In fact it is a conservation law. To see this define F_x and F_y as the flux of γ_1 in the x and y direction by

$$F_x = u\gamma_1 \quad F_y = v\gamma_1,$$

then (4.8) can be rewritten as

$$\frac{\partial \gamma_1}{\partial t} + \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} = 0. \tag{4.9}$$

For simplicity let us assume a one dimensional case:

$$\frac{\partial \gamma_1}{\partial t} + \frac{\partial F_x}{\partial x} = 0. \tag{4.10}$$

By integrating equation (4.10) between the boundary points a and b we get

$$\begin{aligned}\int_a^b \frac{\partial \gamma_1}{\partial t}\mathrm{d}x + \int_a^b \frac{\partial F_x}{\partial x}\mathrm{d}x &= \frac{\partial}{\partial t}\left(\int_a^b \gamma_1 \mathrm{d}x\right) + \int_a^b \frac{\partial F_x}{\partial x}\mathrm{d}x \\ &= \frac{\partial}{\partial t}\left(\int_a^b \gamma_1 \mathrm{d}x\right) + F_x(\gamma_{1(b,t)}) - F_x(\gamma_{1(a,t)})\end{aligned}$$

or simply

$$\frac{\partial}{\partial t}\left(\int_a^b \gamma_1 \mathrm{d}x\right) = F_x(\gamma_{1(a,t)}) - F_x(\gamma_{1(b,t)}). \tag{4.11}$$

Thus we can state that the total amount of the quantity γ_1 in the interval $[a, b]$ can only change due to the flux across the boundary points a and b.

In the VOF method formulated by Hirt et al. [18] equation (4.8) is used to compute the volume fraction γ_1. To avoid smearing at the interface and thus

preserve a sharp interface, the integration of (4.8) over a computational cell reduces the flux across the interface for a change in γ_1 [18]. This is referred to as compression of the interface.

According to [16, 4] the upper boundedness $\gamma_1 \leq 1$ is not guaranteed. To circumvent this problem a surface compression VOF method was implemented in OpenFoam. In the surface compression VOF method an extra term is added to equation (4.8) to further compress the interface in order to reduce smearing at the interface.
The surface compression VOF formulation for γ_1 is

$$\frac{\partial \gamma_1}{\partial t} + \nabla \cdot (\mathbf{U}\gamma_1) + \nabla \cdot [\mathbf{U}_r \gamma_1 (1 - \gamma_1)] = 0 \tag{4.12}$$

where $\mathbf{U}_r$ is an explicitly fixed relative velocity and $\mathbf{U}$ is set to be the mixture velocity of the flow. The semi-discrete formulation of (4.12) is given by [16]

$$(\gamma_1)_i^{n+1} = (\gamma_1)_i^n - \Delta t \left(\nabla \cdot [(\gamma_1 U) + U_r \gamma_1 (1 - \gamma_1)]\right). \tag{4.13}$$

Integrating (4.13) over the control volume v gives

$$\int_V (\gamma_1)_i^{n+1} \, \mathrm{d}V = \int_V (\gamma_1)_i^n \, \mathrm{d}V - \Delta t \int_V \left(\nabla \cdot [(\gamma_1 U) + U_r \gamma_1 (1 - \gamma_1)]\right) \mathrm{d}V. \tag{4.14}$$

The term involving ∇ can be computed by using Gauss's theorem and integrating over a control volume V:

$$\int_V [(\gamma_1 U) + U_r \gamma_1 (1 - \gamma_1)] = \sum_f (\gamma_1 \phi)_f + (\gamma_1 \phi_r (1 - \gamma_1))_f \tag{4.15}$$

where f denotes the face of the control volume and ϕ, ϕ_r are the fluxes of the mixture velocity and the relative velocity respectively.

Chapter 5

Simulation and Parameter Optimisation Strategy

Computational Fluid Dynamics (CFD) uses various numerical methods and algorithms to solve and analyse fluid flow problems such as multiphase flow. The solution procedure in any CFD problem involves discretising the flow domain, discretising the continuous differential equation(s) to produce discrete equation(s) and applying various solution methods to solve the discrete equation(s).
The open-source CFD software OpenFoam is used here to simulate the rising of a single air bubble in liquid water, and then a fully dispersed two-phase flow. The discrete set of algebraic equations is solved either by the PISO algorithm or by the SIMPLE algorithm.
This chapter is divided as followed: Chapter One presents the Richardson extrapolation method and Grid Convergence Index (GCI) method. The GCI method is used to compute the discretisation error. Chapter Two gives an overview of neural networks. In chapter Three presents the numerical parameters affecting the numerical solution(s). Chapter Four and Five present the simulation of a single rising bubble and a fully dispersed two phase flow respectively. Chapter Six outlines the parameter optimisation strategy/algorithm follow by results obtained by applying the optimisation strategy/algorithm.

5.1 Richardson extrapolation and Grid Convergence Index

The solution of a discretised differential equation is prone to different types of errors. Since differential equations are typically solved on a computer, the solution is prone to round-off error. On a computer, each real number is represented by a fixed number of digits. So in practice each real number is rounded off to some near approximation, resulting in a round-off error. The

discretisation process of the continuous differential equation to its discrete form involves approximating the continuous derivative to a discrete derivative though Taylor's theorem. The truncation error is then defined as the difference between the true derivative and the finite difference approximation [26]. A more important type of error is the discretisation error, which is the main source of numerical error [25]. The discretisation error measures the error induced by replacing the continuous differential equation by the discrete equation and by applying various methods to obtain the approximate solution of the continuous problem. It is more concerned with the entire numerical scheme, including the algorithms and the solution methods built to solve the differential equation.
Stability, convergence, satisfaction of the CFL conditions etc. are taken into account, and the error will be a measure of the validity of the entire scheme.

For the estimation of the discretisation error in CFD, the most reliable method is Richardson's extrapolation method, in particular its variant Grid Convergence Index (GCI) method [21, 22, 25]. The Richardson extrapolation method was first introduced by Richardson in 1910 and was improved later in 1927 [21]. In the method the discretised solution u_h of a differential equation is assumed to have a series representation depending on the mesh size h [23, 24]

$$u(\mathbf{x}, h) = u(\mathbf{x}) + hg_1 + h^2 g_2(\mathbf{x}) + h^4 g_4(\mathbf{x}) + \cdots \tag{5.1}$$

where u is the exact solution and g_i is some function which does not depend on any discretisation [21]. From (5.1) the discretisation error δ_j for grid j can be defined as

$$\delta_j = u_j(\mathbf{x}, h) - u(\mathbf{x}) = hg_1 + h^2 g_2(\mathbf{x}) + h^4 g_4(\mathbf{x}) + \cdots. \tag{5.2}$$

For a first and second order discretisation scheme equation (5.2) can be approximated respectively by [25]

$$\delta_1 = u_j(\mathbf{x}, h) - u(\mathbf{x}) \approx gh^p \tag{5.3}$$

$$\delta_2 = u_j(\mathbf{x}, h) - u(\mathbf{x}) \approx h_j g_1 + h_j^2 g_2 \tag{5.4}$$

where p is the apparent order of convergence, h_j is the characteristic grid spacing. In practice the exact solution u is the zero grid solution, meaning the solution at zero grid size:

$$u = \lim_{h_j \to 0} u_h.$$

In equation (5.3) there are three unknowns: u, g, p, while the unknowns in equation (5.4) are u, g_1, g_2. To find these unknowns, the approximate solution of the differential equation must be computed on three grids , $j =$

$1, 2, 3$ where 1 is the finer grid, 2 the medium grid and 3 the coarser grid. The unknowns can then be computed by using a least square root approach to solve the minimisation problem [22]:

$$\min S(u, g, p) = \sqrt{\left(\sum_{j=1}^{3} (u_j^h - (u + g h_j^p))^2 \right)} \tag{5.5}$$

$$\min S(u, g_1, g_2) = \sqrt{\left(\sum_{j=1}^{3} (u_j^h - (u + \alpha_1 h_j^2 + \alpha_2 h_j^3))^2 \right)} \tag{5.6}$$

A more generalised Richardson extrapolation method was proposed by Roache [21], called Grid Convergence Index method (GCI). Assuming three locally computed solutions on mesh 3,2 and 1, the ratio change of the solution R is computed:

$$R = \frac{u_2 - u_1}{u_3 - u_2} \tag{5.7}$$

and the computational nodes are classified as [22]

$$\begin{aligned} 0 < R < 1 \quad & \text{monotonic convergence} \\ R < -1 \quad & \text{oscillatory divergence} \\ R > 1 \quad & \text{monotonic divergence.} \end{aligned} \tag{5.8}$$

For monotonic divergent computational nodes, the apparent convergence rate can be computed by [21, 22]

$$p = \frac{\log(1/R)}{\log(r_{21})} \tag{5.9}$$

where $r = r_{21} = h_2/h_1 = h_3/h_2$, and h_i is the size of grid i.

The GCI defined in [21], also called error band U_1 in [25] can now be computed on the finest grid. The error band can be interpreted as the uncertainty on the computed solution or the discretisation error. It is the uncertainty of how far the computed solution deviates from the zero grid solution [21]. Even though the zero grid solution can be explicitly computed from the GCI method, the latter is not conservative [21]. It is therefore recommended to use the GCI method to evaluate the discretisation error or error band U_1. Using (5.6) and based on the value of p computed from equation (5.5) or (5.9), and assuming that (3.1) is satisfied, U_1 is computed as follows [25]:

$$U_1 = 1.25|\delta_1|, \quad \text{for } 0.5 < p \leq 2$$

$$U_1 = 1.25 \max(|\delta_1|, |\delta_2|), \quad \text{for } 2 < p \leq 3$$

$$U_1 = 3 \max(|u_3 - u_2|, |u_2 - u_1|), \quad \text{for } p \leq 0.5 \text{ or } p > 3$$

The admissible values of p for the computation of the discretisation error are in the range $0 < p < 8$ [25].

In Roaches description of the grid convergence index method [21], the error band for the finer grid GCI_1 and the coarser grid GCI_2 are computed respectively from

$$GCI_1 = 100 \frac{FsU_1}{r^p - 1} \tag{5.10}$$

$$GCI_2 = 100 \frac{FsU_2}{r^p - 1} \tag{5.11}$$

where $Fs = 1.25$, $r = r_{21} = h_2/h_1 = h_3/h_2$ and

$$U_1 = \frac{u_1 - u_2}{u_1} \tag{5.12}$$

$$U_2 = \frac{u_2 - u_3}{u_2} \tag{5.13}$$

An important notion in the GCI method is the asymptotic convergence range. Given three grids (coarse, medium and fine), the finer grid is in the asymptotic range of convergence if the asymptotic convergence rate (ACR) is approximately equal to one. The latter is given by

$$ACR = \frac{GCI_1}{GCI_2} \frac{1}{r^p}. \tag{5.14}$$

Equation (5.14) is based on the general observation that the discretisation error is proportional to H^p where H is some parameter involving the mesh size and p is the convergence rate:

$$error = CH^p. \tag{5.15}$$

If ACR is set to equal one we get

$$GCI_1 = GCI_2 r^p. \tag{5.16}$$

Comparing equation (5.15) and (5.16) we can see the equivalences $C = GCI_2$ and $r = H$, as GCI_1 is the error. As we will demonstrate later, the GCI method provides an efficient way to evaluate the discretisation error, also called error band. The latter can be expressed as the percentage deviation from the zero grid solution. The importance in computing the discretisation error lies in the fact that residuals from solvers (interFoam,

twoPhaseEulerFoam) do not necessary take into account the overall accuracy of the solution.

5.2 Neural networks

A neural network or more precisely an artificial neural network is a circuit composed of interconnected simple circuit elements called neurons. Each neuron represents a map, typically with multiple input and single or multiple output [28]. The output of a neuron is a function of the sum of the inputs.

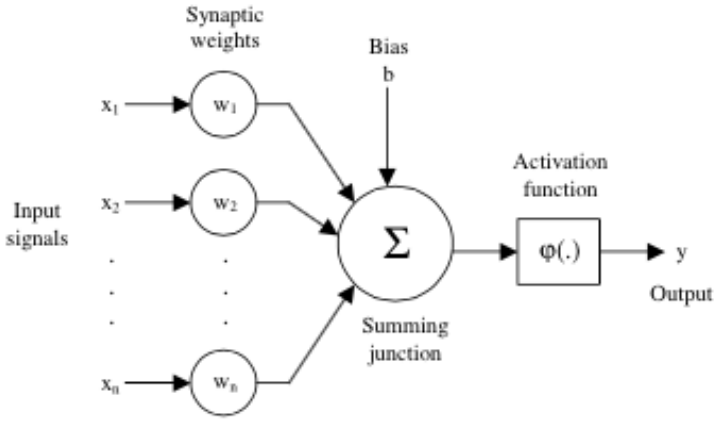

Figure 5.1: A single neuron with multiple inputs and a single output. Taken from [29]

The output of the neuron is called activation function. Figure 5.1 shows a single neuron with multiple inputs. The output of the neuron is given by

$$y = \varphi\left(\sum_{j=1}^{n} w_j x_j + b\right) \tag{5.17}$$

where w_j are the weights of the neuron network, b is called the bias of the network and φ is the activation function. Figure 5.1 represented by equation (5.17) can be viewed as a map from $\mathbb{R}^n$ to $\mathbb{R}$. In practical problems it is custom to use a feedforward neural network where the neurons are interconnected in layers. In such a network, the first layer is the input layer and the last layer is the output layer. The layers in between the input and the output layer are called hidden layers.

A feedforward neural network is shown in figure 5.2. This network represents a generalisation of a neural network as a map from $\mathbb{R}^n$ to $\mathbb{R}^m$ and

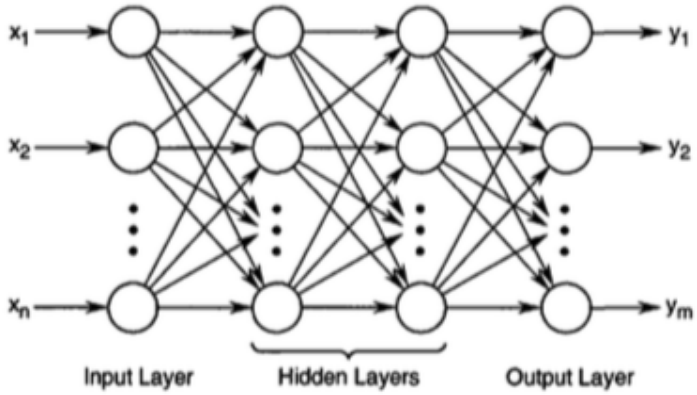

Figure 5.2: A feedforward neural network with n inputs, m outputs and several hidden layers. Taken form [28].

can be defined by

$$y_s = f_s^o \left(\sum_{j=1}^{l} w_{sj}^o f_j^h \left(\sum_{i=1}^{n} w_{ji}^h x_i \right) \right) \tag{5.18}$$

where f_s^o, f_j^h are the activation functions of the output layer and the hidden layers respectively. w_{sj}^o and w_{ji}^h are the weights of the output layer and the hidden layers. In this work we are only considering a feedforward network with one hidden layer, as shown in figure 5.3.

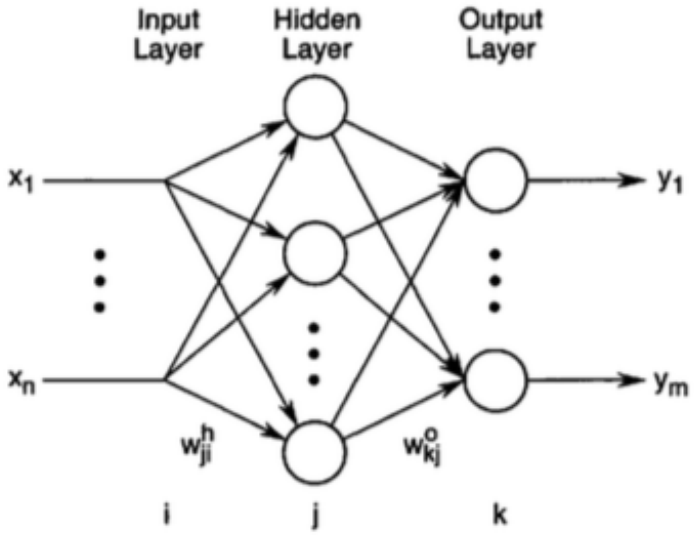

Figure 5.3: A feedforward network with one hidden layer. taken form [28]

The activation function of a neural network can be any bounded function from $\mathbb{R}$ to $\mathbb{R}$. For the present work we use the Sigmoid function as activation function for the hidden layer and the output layer, given by equation (5.19) and figure 5.4.

$$f_s^o(\theta) = f_j^h(\theta) = \frac{1}{1+e^{-\theta}} \tag{5.19}$$

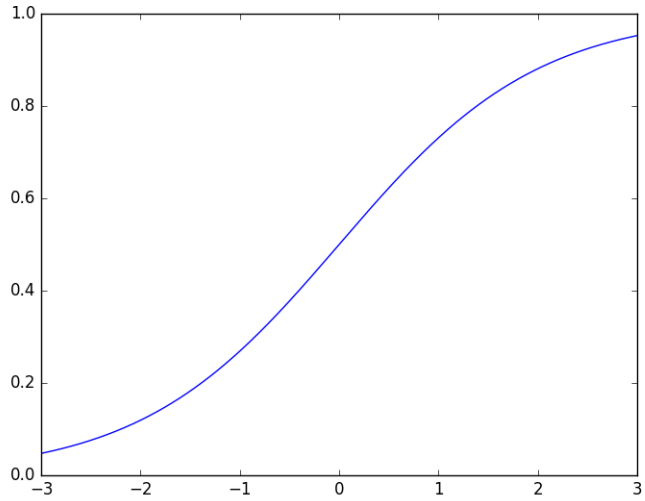

Figure 5.4: The sigmoid function

Given the neural network from figure 5.2 and equation (5.18), the objective is to approximate a given process defined by input $\{x_i\}_{i=1}^n$ and output $\{y_s\}_{s=1}^m$. To do this the network needs to be trained. The training process involves adjusting the weights of the network such that the output generated by the network for the given input $\{x_i\}_{i=1}^n$ is close to the output $\{y_s\}_{s=1}^m$ as much as possible [28]. The training process can be formulated as the following optimisation problem [28]

$$minimise \quad E(w) = \frac{1}{2}\sum_{s=1}^{m}(yin_s - y_s) \tag{5.20}$$

where yin_s is the output of the process while y_s given by equation (5.18) is the output of the network in response to the process input $x = \{x_i\}_{i=1}^n$. The objective of the minimisation problem (5.20) is to find the weights

$$w = \{w_{sj}^o, w_{ji}^h : i = 1, \cdots, n; j = 1, \cdots, l; s = 1, \cdots, m\},$$

where l, m are the number of neurons in the hidden and output layer respectively. Problem 5.20 is solved by using the backpropagation algorithm

[28] involving the following steps:

1. derive the derivative of the objective function E with respect to w^o_{sj} by using the chain rule
2. derive the derivative of the objective function E with respect to w^h_{ji} by using the chain rule
3. formulate the update equation for w^o_{sj} and w^h_{ji}

By setting

$$z_q = f^h_q \left(\sum_{i=1}^{n} w^h_{qi} \right)$$

we can simplify y_s and apply the chain rule to obtain

$$\frac{\partial E}{\partial w^0_{sj}}(w) = -(yins - y_s) f^{o\prime}_s \left(\sum_{q=1}^{l} w^o_{sq} z_q \right) z_j$$

and by setting

$$\delta_s = -(yins - y_s) f^{o\prime}_s \left(\sum_{q=1}^{l} w^o_{sq} z_q \right)$$

we get the partial derivative of E with respect to w^0_{sj}

$$\frac{\partial E}{\partial w^0_{sj}}(w) = -\delta_s z_j. \tag{5.21}$$

Similarly,

$$\frac{\partial E}{\partial w^h_{ji}}(w) = -\sum_{p=1}^{m} (yins - y_s) f^{o\prime}_p \left(\sum_{q=1}^{l} w^o_{pq} z_q \right) w^o_{pj} f^{h\prime}_j \left(\sum_{r=1}^{n} w^h_{jr} x_r \right)$$

and by setting

$$v_j = \sum_{r=1}^{n} w^h_{jr} x_r$$

we get the partial derivative of E with respect to w^h_{ji}

$$\frac{\partial E}{\partial w^h_{ji}}(w) = -\left(\sum_{p=1}^{m} \delta_p w^o_{pj} \right) f^{h\prime}_j(v_j) x_i \tag{5.22}$$

Now the update equation for w_{sj}^0 and w_{ji}^h can be formulated for a fixed step size η:

$$\left(w_{sj}^o\right)^{k+1} = \left(w_{sj}^0\right)^k + \eta(\delta)^k(z_j)^k \tag{5.23}$$

$$\left(w_{ji}^h\right)^{k+1} = \left(w_{ji}^h\right)^k + \eta\left(\sum_{p=1}^{m}(\delta_p)^k(w_{pj}^o)^k\right) f_j^{h\prime}(v_j)^k x_i \tag{5.24}$$

with

$$(v_j)^k = \sum_{i=1}^{n} w_{ji}^{h(k)} x_i \tag{5.25}$$

$$(z_j)^k = f_j^h\left((v_j)^k\right) \tag{5.26}$$

$$(y_s)^k = f_s^o\left(\sum_{q=1}^{l} w_{sq}^{o(k)}(z_q)^k\right) \tag{5.27}$$

$$\delta_s = -(yins - (y_s)^k) f_s^{o\prime}\left(\sum_{q=1}^{l} w_{sq}^{ok}(z_q)^k\right) \tag{5.28}$$

5.3 Numerical parameters

5.3.1 General formulation of the SIMPLE algorithm

SIMPLE stands for Semi-Implicit Methods for Pressure-Linked Equations. The algorithm is a guess-and-correct procedure for the computation of solutions of a differential equation [1]. The algorithm has one predictor (guess) and one corrector step. It was developed by Patankar and Spalding in 1972 [1, 3].
For simplicity let f and g be the discretized momentum and continuity equations for a steady-state convection-dominated problem given respectively by

$$f(U, p) = 0 \tag{5.29}$$

$$g(U) = 0 \tag{5.30}$$

where U and p are the correct velocity and pressure fields. To initialize the SIMPLE algorithm, a guessed value p^* of the pressure field is fed into the discretised momentum equation (5.29), i.e.

$$f(U^*, p^*) = 0. \tag{5.31}$$

The latter is solved for U^*. Since p^* was guessed, the computed velocity field U^* may or may not satisfy the discrete continuity equation (5.30). Defining the correction of the velocity and the pressure by U' and p' respectively, the correct velocity and pressure can be expressed as

$$U = U^* + U' \tag{5.32}$$

$$p = p^* + p'. \tag{5.33}$$

Subtracting equation (5.29) from (5.31) yields

$$f(U, p) - f(U^*, p^*) = F(U - U^*, p - p^*) = 0, \tag{5.34}$$

and inserting equation (5.32) and (5.33) into (5.34) gives the correction equation

$$F(U', p') = 0. \tag{5.35}$$

The solution of (5.35) can be expressed as

$$U' = Cp' \tag{5.36}$$

for some constant C. Inserting the last expression into the correction equation (5.32) gives

$$U = U^* + Cp'. \tag{5.37}$$

Since continuity must be satisfied, the latter equation is inserted into the discretised continuity equation (5.30) yielding

$$g(U^* + Cp') = 0. \tag{5.38}$$

Equation (5.38) represents the discretised continuity equation as an equation for pressure correction p' [1, 3]. Solving for the pressure correction p' the correct pressure p is obtained from equation (5.33) and the correct velocity field U can be computed from (5.36) and (5.32).
The pressure correction equation (5.38) is susceptible to divergence unless some under-relaxation is used during the iterative process [1]. Thus the pressure p and velocity U can be under-relaxed as

$$p^{new} = p^* + \gamma_p p'$$
$$U^{new} = \gamma_U + (1 - \gamma_U)U^{n-1}$$

where γ_p, γ_U are the under-relaxation factors for pressure and velocity. Note that in general $U = (u, v, w)$ and to each velocity component corresponds an under-relaxation factor. U^{n-1} is the computed velocity at the previous iteration. The under-relaxation factor is characterised by

$$0 \leq \gamma \leq 1.$$

Too large values of γ can induce oscillatory solutions or divergent iteration solutions, and too small values can cause extremely slow convergence [1]. The optimum value of γ is flow dependent and must be sought on a case-by-case basis [1]. The computational flow of the SIMPLE algorithm can be formulated as [2]:

1. Guess the pressure p^*
2. Solve the discretised momentum equation (5.29) to obtain $U^* = (u^*, v^*, w^*)$
3. Solve the correction equation (5.38) for p'
4. Compute p from (5.33)
5. Compute $U = (u, v, w)$ from (5.37)
6. Compute other flow field(s) such as turbulence quantities, heat transfer terms etc ...
7. Set $p^* = p$ and return to step 2 until convergence is achieved

It is recommended to incorporate step 6 into the algorithm only if the computed quantities affect the flow significantly. If this is not the case, this step should be removed from the algorithm and performed after convergence of

the flow field is achieved [2].

Despite the success of the SIMPLE algorithm in producing solutions for a wide range of problems in computational fluid dynamics, it suffers from a few shortcomings. The original algorithm was developed for steady-state problems and uses a first order finite difference scheme for problems involving convection in Cartesian or polar coordinates [3]. Thus for complex geometry problems the algorithm is not suitable. Instead, for unsteady flow computations the PIMPLE algorithm must be applied at each time step, until convergence is achieved. More details about this algorithm will follow.

5.3.2 General formulation of the PISO Algorithm

The PISO algorithm, which stands for Pressure Implicit with Splitting of Operator, is a pressure-velocity computation procedure. It was originally developed for non-iterative computation of unsteady compressible flows, but has also been used successfully for iterative solutions of steady-state problems [1]. It may be seen as an extension of the SIMPLE algorithm with one extra corrector step [1]. In the SIMPLE algorithm the correct velocity and pressure where given by equation (5.32) and (5.33). To illustrate the PISO algorithm we set

$$U^{**} = U = U^* + U'$$

$$p^{**} = p = p^* + p'.$$

where U and p are the correct velocity and pressure obtained from the first SIMPLE correction while U^* and p^* are the guessed pressure and velocity fields. In the PISO algorithm a second correction is performed by setting

$$U^{***} = U^{**} + U'' \tag{5.39}$$

$$p^{***} = p^{**} + p''. \tag{5.40}$$

where U'' and p'' are the correctors for the velocity and pressure respectively. The first procedure employed in the SIMPLE algorithm is used in the PISO algorithm to find a second corrector equation

$$g(U^{**} + Cp'') = 0, \tag{5.41}$$

similarly to (5.38). As in SIMPLE, the relationship between the correctors for pressure and velocity similar to (5.36) is given by

$$U'' = kp'' \tag{5.42}$$

for some constant k. Solving (5.41) for p'' and using (5.42), the correctors for pressure and velocity are computed. Using (5.39) and (5.40) we compute

the correct pressure and velocity, obtained from the second corrector step. The computational flow of the PISO algorithm can be formulated as follows [1]:

1. Perform step 1-3 of SIMPLE algorithm
   - Solve discretised momentum equation
   - Solve pressure correction equation
   - Correct pressure and velocities
2. Solve second pressure equation
3. Correct pressure and velocity
4. Solve all other discretised transport equations. If convergence stop, else go to step 1.

5.4 Simulation of a single rising bubble

In the previous sections we saw that the SIMPLE algorithm required one pressure correction while the PISO algorithm required two. In addition, the pressure and velocity fields need to be under-relaxed to avoid divergence. In practice, to achieve convergence several correction numbers are needed.
In OpenFoam the SIMPLE and the PISO algorithm are combined to form the PIMPLE algorithm. We therefore expect at least three numerical parameters from the PIMPLE algorithm: the number of SIMPLE corrections, the number of PISO corrections and the under-relaxation factor. By using the Volume of Fluid (VOF) method described in chapter 4 and implemented in OpenFoam, our objective is to:

1. Simulate the rising of a single air bubble in liquid water
2. Compare our result with reported benchmark data from the literature
3. Show how the above numerical parameters affect the solution in terms of accuracy and execution time

5.4.1 Interface-capturing model for single bubble in a liquid

In this section we derive the mathematical equation describing the motion of a single rising bubble in water. A single bubble rising in water is governed by the Navier-Stokes equations and the Continuity equation [16]

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot (\rho \mathbf{U}\ \mathbf{U}) - \nabla \cdot \tau = -\nabla P + \rho \mathbf{g} + \sigma \kappa \nabla \gamma_i \tag{5.43}$$

$$\nabla \cdot \mathbf{U} = 0 \tag{5.44}$$

where the stress tensor τ is given by

$$\tau = \mu(\nabla \mathbf{U} + \nabla \mathbf{U}^T).$$

The volume fraction γ_2 of air (2) is computed from the volume fraction γ_1 of water (1) by

$$\gamma_2 = 1 - \gamma_1.$$

The mixture density ρ and the mixture viscosity μ are given by

$$\rho = \gamma_1 \rho_1 + (1 - \gamma_1)\rho_2$$
$$\mu = \gamma_1 \mu_1 + (1 - \gamma_1)\mu_2.$$

The curvature κ of the interface is given by

$$\kappa = -\nabla \cdot \mathbf{n}, \tag{5.45}$$

where the interface unit normal vector n is given by [4]

$$n = \frac{\nabla \gamma}{|\nabla \gamma| + \delta}. \tag{5.46}$$

Since equation (5.45) is only valid at the interface, outside the interface $|\nabla \gamma| \to 0$; therefore δ which is a very small number is used to avoid division by zero.
This model assumes that the bubble is subject to gravitational force g and surface tension forces σ. The latter force is the attractive force exerted upon the surface molecules of the fluid. It is directed inside the fluid. In the fluid, each molecule exerts an equally strong force in all directions upon each other, resulting in a zero net force. However, the molecules at the interface are pulled inside the fluid since they have no neighbouring molecules at the interface side. The influence of surface tension and gravity forces is given by Eötvös number [17]

$$Eo = \frac{\mathrm{g}\Delta\rho d^2}{\sigma} = \frac{\text{gravitational force}}{\text{surface tension}} \tag{5.47}$$

where $\Delta\rho$ is the variation in density and d is the diameter of the bubble. Note that the diameter of the bubble will vary during its ascending motion. The surface tension σ will affect significantly the shape of the bubble.
For small surface tension, gravitational forces dominate and the Eötvös number is larger. From figure 5.5 we see that the bubble is greatly deformed for a Reynolds number greater than approximately 2. We can interpret the surface tension as a cohesive force acting on the bubble.

Equation (5.43) and (5.44) are the same for the continuous fluid (water) and the bubble filled with air. The volume fraction of each fluid is used as an

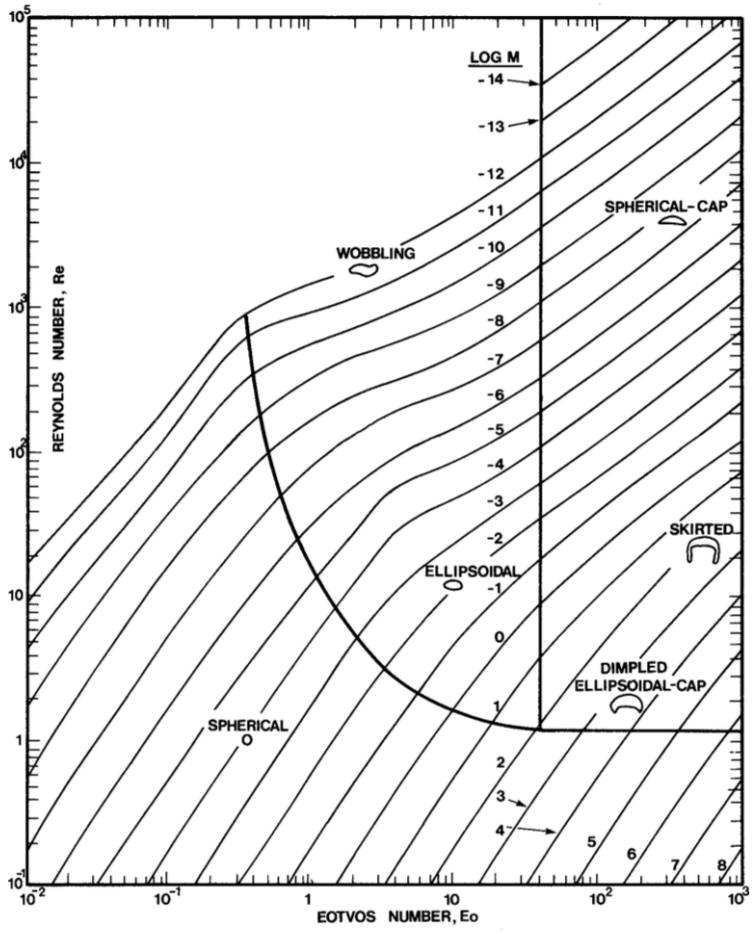

Figure 5.5: Effect of surface tension on bubble size

indicator function to define the difference between the two fluids separated by the interface where a transitional region exists [4]. The indicator function is given by

$$\gamma_i = \begin{cases} 1 & \text{in fluid } i \\ 0 < \gamma_i < 1 & \text{at the interface} \\ 0 & \text{in fluid } i \pm 1 \end{cases}$$

and is governed by the transport equation

$$\frac{\partial \gamma_i}{\partial t} + \nabla \cdot (\mathbf{U}\gamma_i) = 0 \quad i = 1, 2 \tag{5.48}$$

where $\mathbf{U}$ is the velocity of the flow [4, 18].
Note that equation (5.48) is a Conservation Law: it indicates that the quantity modelled by γ is neither created nor destroyed. To see this, define the flux function f by

$$f(\gamma(y,t)) = \mathbf{U}\gamma(y,t)$$

and assume that the bubble is rising upwards in the y direction. By integrating (5.48) between points a and b we get

$$\begin{aligned} \int_a^b \frac{\partial \gamma}{\partial t} = \frac{d}{dt}\int_a^b \gamma(y,t)\mathrm{d}y = -\int_a^b \frac{df(\gamma(y,t))}{dx}\mathrm{d}y \\ = f(\gamma(a,t)) - f(\gamma(b,t)). \end{aligned} \tag{5.49}$$

This means that the rate of change of γ can only change due to the flow across the boundary points a and b. It was noted ([4]) that the form of equation (5.48), which was in the original formulation of the VOF method in [18], could not damp the smearing of the interface enough. To circumvent this an artificial compression term only valid at the interface was added to (5.48) to give

$$\frac{\partial \gamma_i}{\partial t} + \nabla \cdot (\mathbf{U}\gamma_i) + \nabla \cdot [\mathbf{U}_r \gamma_i (1 - \gamma_i)] = 0 \quad i = 1, 2 \tag{5.50}$$

where U_r is the velocity needed to compress the interface and given as the relative velocity between the two fluids. To solve equation (5.50) it suffices to only consider one fluid, $i = 1$.

5.4.2 Setting up the simulation case

The simulation case is taken from [19] where the complete benchmark data and definition are given. Using nondimensionalization, the simulation domain is a rectangle with length scale $x : y = 1 : 2$. A bubble with original

diameter $r_b0 = 0.25$ is initially located at position $(x, y) = (0.5, 0.5)$. A no-slip boundary condition is imposed at the top and bottom of the rectangle, while slip wall boundary conditions are imposed at the left and right of the domain, see figure 5.6.

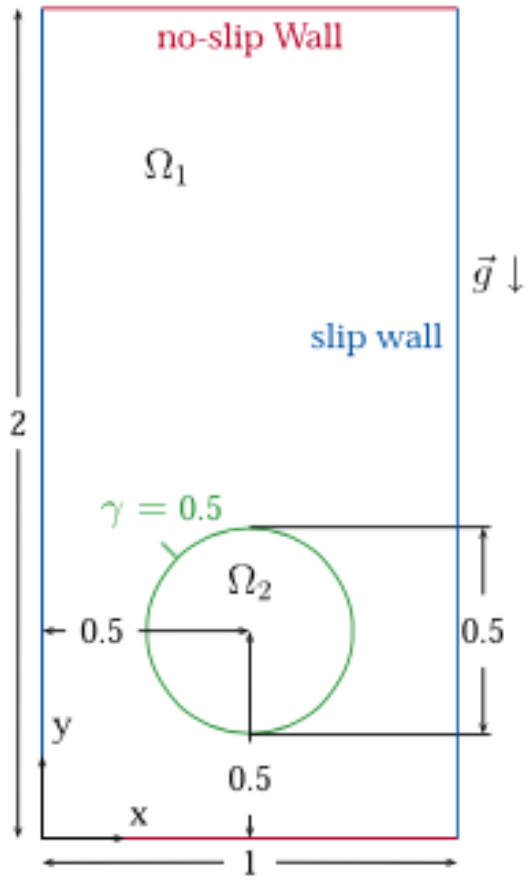

Figure 5.6: Computational domain of a rising bubble. Figure taken from [16] and simulation configuration taken from [19].

The water occupies a domain Ω_1 while the bubble occupies a domain Ω_2. The interface is then given by $\partial\Omega = \Omega_1 \cap \Omega_2$. Using dimensionless analysis, the characteristic length scale, time scale, and velocity scales are given respectively by

$$L = 2r_{b0}, \quad t = \frac{L}{U_g} \quad , U_g = \sqrt{g2r_{b0}}.$$

The Reynolds number R_e, Eötvös number E_o and the Capillary number Ca are given respectively by

$$R_e = \frac{\rho_1 U_g L}{\mu_1}, \quad E_o = \frac{\rho_1 U_g^2 L}{\sigma}, \quad Ca = \frac{\mu_1 U_g}{\sigma} = \frac{E_o}{R_e}.$$

For comparison with the benchmark data given in [19] and the study done in [16], the maximum rising velocity u_{max} and the maximum rising position h_{max} of the bubble must be computed. To do this we must compute the time evolution of the centroid and the velocity of the bubble given by [19, 16]:

$$Y_c = \frac{\int_{\partial\Omega} \gamma_2 y_c \mathrm{d}A}{\int_{\partial\Omega} \gamma_2 \mathrm{d}A}, \quad u = \frac{\int_{\partial\Omega} \gamma_2 v_c \mathrm{d}A}{\int_{\partial\Omega} \gamma_2 \mathrm{d}A}. \tag{5.51}$$

In equation (5.51) Y_c is the centroid of the bubble, γ_2 is the volume fraction of air inside the bubble, y_c is the vertical component of the bubble centroid, u is the rising vertical velocity of the bubble computed from the velocity of the centroid.

5.4.3 OpenFoam simulation

OpenFoam is a C++ library used to create executables known as applications [20]. There are two types of applications in OpenFoam: Solvers and Utilities. The Solvers are designed to solve a range of problems in continuum mechanics, while Utilities are used to perform post-processing tasks [20]. Through the C++ language users can in principle build their own utilities and solvers, building either on existing ones or from scratch.

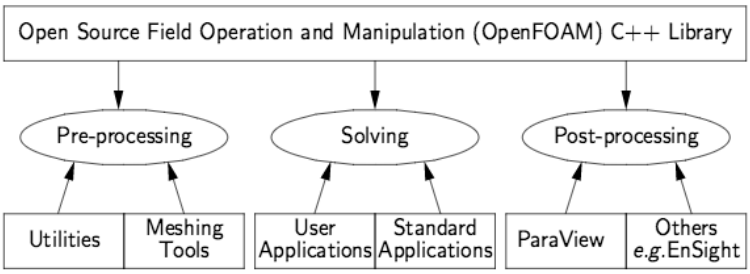

Figure 5.7: The structure of OpenFoam [20]

Figure 5.7 shows the structure of OpenFoam. To run an OpenFoam solver and solve a specific problem, users must select a suitable solver and prepare a case directory. An OpenFoam case directory contains three subdirectories: the 0 directory, the *constant* directory and the *system* directory. The 0 directory contains files whose contents hold the solution field of the problem to be solved. For example, if one wants to solve the Navier-Stokes equation for pressure p and velocity u, then the 0 directory must contain files u and p. The *constant* directory contains the physical properties of the problem and the definition of the mesh. In the *system* directory the user can

specify time step, the duration of the simulation, the finite volume scheme, the algorithm, and common numerical parameters such as under-relaxation parameters and the numerical tolerance for each solution. Figure 5.8 shows a typical directory structure of a simulation case.

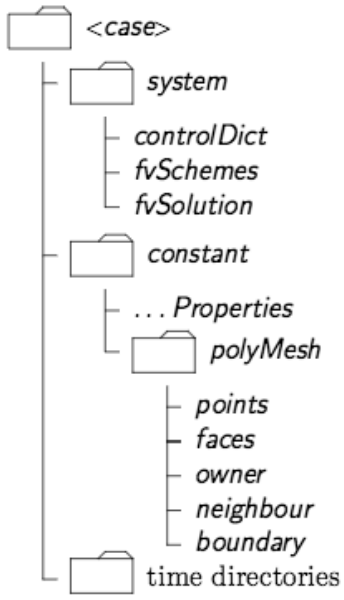

Figure 5.8: A case directory structure in OpenFOam

It is custom that a user desires to change the existing solver by adding additional models. In such a case the user must make a copy of an existing solver, modify the solver and recompile it. In our case we used the interFoam solver, which is a solver for 2 incompressible fluids, and which tracks the fluid interface by using the volume of fluid method (VOF).

Mesh generation for the rising bubble case

```
vertices
(
    (0 0 0)
    (1 0 0)
    (1 2 0)
    (0 2 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 2 0.1)
    (0 2 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (80 80 1) simpleGrading (1 1 1)
);

boundary
(
    leftWall {type wall;faces((0 4 7 3) );}
    rightWall {type wall;faces( (2 6 5 1));}
    bottom { type wall; faces( (0 4 5 1) );}
    atmosphere {type wall; faces( (3 7 6 2)); }
    frontAndBack {type empty; faces ((0 3 2 1)(4 5 6 7)); });
```

Transport Properties

```
phases (water air);
water
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1e-02; //1e-02
    rho             rho [ 1 -3 0 0 0 0 0 ] 1000;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }

    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
    }
}
air
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1e-02; //1e-01
    rho             rho [ 1 -3 0 0 0 0 0 ] 100;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }

    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
    }
}

sigma           sigma [ 1 0 -2 0 0 0 0 ] 24.5;
```

Parameters for PIMPLE algorithm

```
PIMPLE
{
  momentumPredictor no;
  nOuterCorrectors 1;
  nCorrectors 11;
  nNonOrthogonalCorrectors 0;
  nAlhaCorr 1;
  nAlphaSubCycles 2;
  cAlpha 1;
  pRefCell 0;
  pRefValue 0;
  residualControl
  {
    p_rgh
    {
      tolerance 1e-06;
      relTol 0;
      absTol 0;
    }
    U
    {
      tolerance 1e-06;
      relTol 0;
      absTol 0;
    }
  }
}

relaxationFactors
{
  fields
  {
    U 1.08;
    p_rgh 1.08;
    alpha.water 1.08;
  }
  equations
  {
    ".*" 1.053;
  }
}
```

Finite volume schemes

```
ddtSchemes
{
 default Euler;
}

gradSchemes
{
    default          Gauss linear;// second order
}

divSchemes
{
    div(rhoPhi,U)  Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss linear;
    div((muEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default          Gauss linear corrected;
}

interpolationSchemes
{
    default          linear;
}

snGradSchemes
{
    default          corrected;
}

fluxRequired
{
    default          no;
    p_rgh;
    pcorr;
    alpha.water;
}
```

Figure 5.9 shows the initial configuration of the bubble for two different mesh sizes. It can be seen that the interface is more sharp for finer mesh resolutions. Table 5.1 and 5.2 show physical properties used in the simulation and the simulation results for various relaxation factors. The results are compared with the results obtained in [19] and [16].
In [19] there were three different groups, each solving the rising bubble problem with different methods. Group one used the Level Set method and as discretization method the Finite Element method with non-conforming basis functions for the phase fraction, pressure and velocity field. A conforming bilinear basis function was used for the level set function [19]. Group two used the Level Set method to track the interface, while discretization in space was achieved with the Finite Element method. Piecewise linear elements were adopted as basis functions. $P_1 - isoP_2$ elements were used for the pressure and P_1 elements for the velocity field [19]. Group three used a pure Finite Element software package which adopted the inf-sup stable isoparametric Finite Element method to discretize the Navier-Stokes equation, and stabilized the Finite Element method to a discretized convection diffusion equation [19]. In [16] the software package OpenFoam was used. The VOF method was used to track the interface.

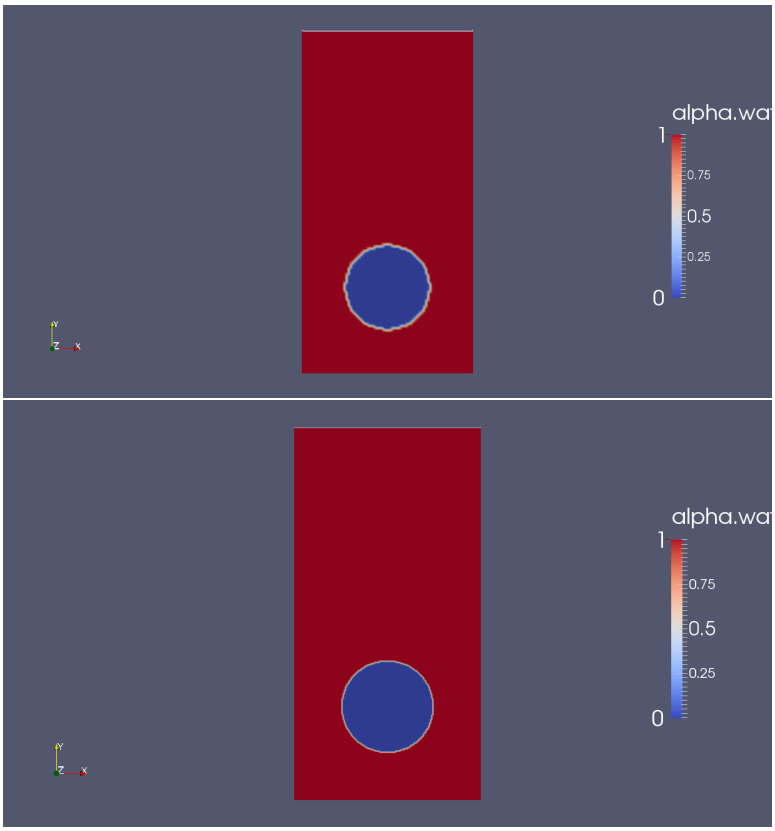

Figure 5.9: Bubble at t=0 for mesh size $x : y = 80, 160$ *(top) and* $x : y = 160, 320$ *(bottom)*

parameters	μ_1	ρ_1	μ_2	ρ_2	R_e	Ca	Eo	σ
Values	10^{-2}	1000	10^{-2}	100	35017	0.00028	10	24.5

Table 5.1: Physical properties used in the simulation

	α_p, α_U	nCorrectors	nOuterCorrectors	dt	U_{max}	y_{max}
present work	1	10	1	0.0125	0.21246	1.00684
present work	1.052	10	1	0.0125	0.2416	1.0882
present work	1.053	10	1	0.0125	0.24216	1.08946
Hysing [19] 1	–	–	–	–	0.2419	1.0812
Hysing [19] 2	–	–	–	–	0.2421	1.0799
Hysing [19] 3	–	–	–	–	0.2417	1.0817
Klostermann [16]	–	–	–	–	0.2365	1.0668

Table 5.2: Simulation parameters and results for changing relaxation factor α. Mesh size for group 1,2,3 from [19] is $x : y = 160, 160$. Mesh size from [16] is $x : y = 160, 160$. Mesh size from this work is $x : y = 80, 160$. The residual obtained in theis present work is of the order of 10^{-6}

From table 5.2 it can be seen that the residuals are similar for relaxation factors 1, 1.052 and 1.53. However, there is a significant difference in the three results when compared with the benchmark results. From figure 5.12, 5.10 and 5.11 we can clearly see the effect of the relaxation factor on the solution. When the solution is not relaxed (relaxation factor = 1), the computed maximum velocity and computed maximum centroid are far away from the benchmark solution. When the solution is over-relaxed (relaxation factor = 1.052, 1.053) the computed solution and the benchmark solution are closer. From numerical experiments, when the solution is under-relaxed (relaxation factor < 1), with value less than one, the computed maximum velocity and maximum centroid are very far from the benchmark solutions. The effect of the relaxation factor cannot be clearly seen on the two dimensional simulation in figure 5.13-5.18, where the three cases for relaxation factor $= 1, 1.052, 1.053$ are shown.

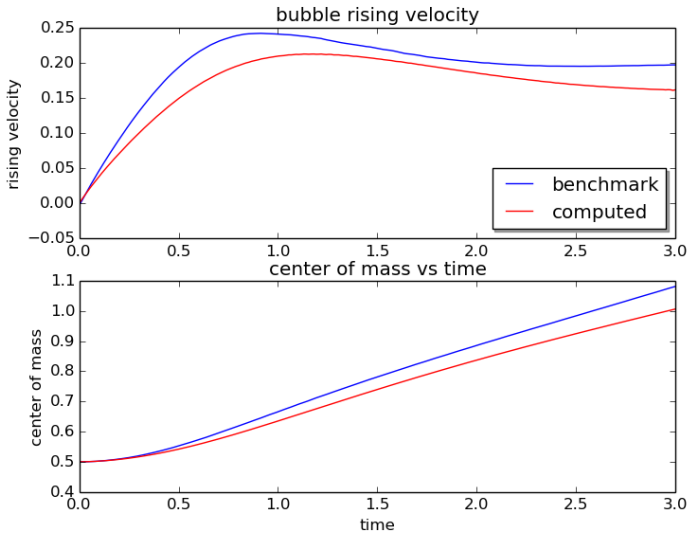

Figure 5.10: Comparison with benchmark [19]. Bubble rising velocity profile (top), and centre of mass (bottom). In our simulation all fields are relaxed with relaxation factor 1. Mesh size $x : y = 80, 160$

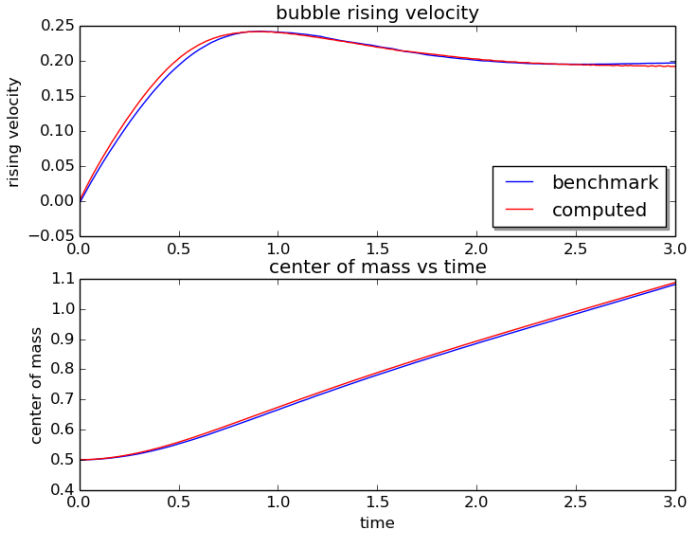

Figure 5.11: Comparison with benchmark [19]. Bubble rising velocity profile (top), and centre of mass (bottom). In our simulation all fields are relaxed with relaxation factor 1.052. Mesh size $x : y = 80, 160$

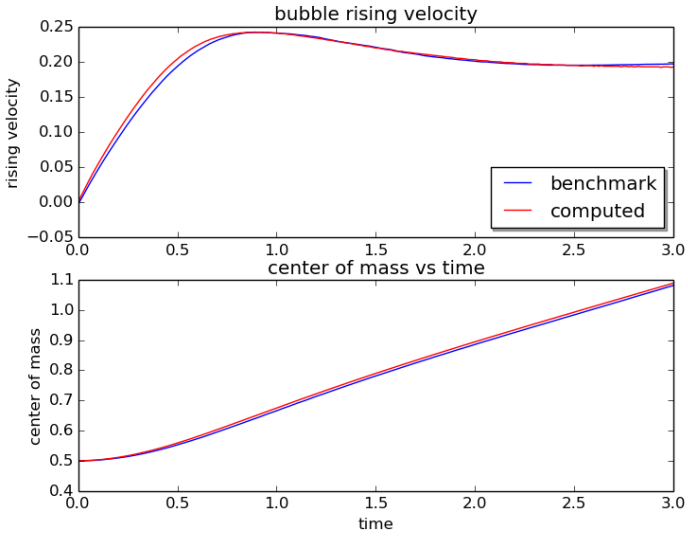

Figure 5.12: Comparison with benchmark [19]. Bubble rising velocity profile (top), and centre of mass (bottom). In our simulation all fields are relaxed with relaxation factor 1.053. Mesh size $x : y = 80, 160$

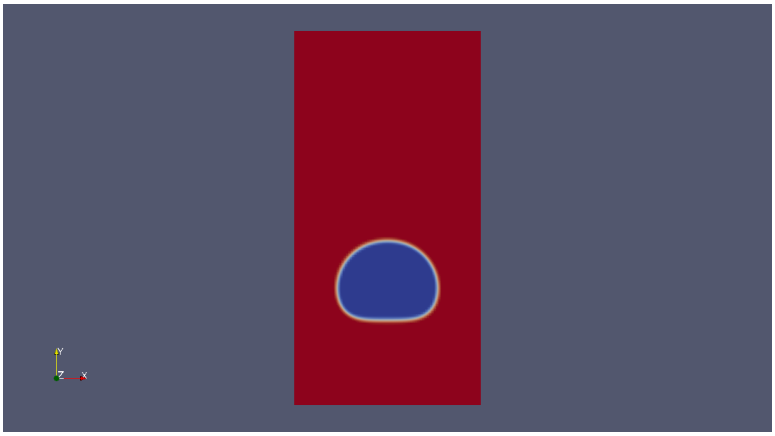

Figure 5.13: Bubble at $t = 1s$ with relaxation factor 1.053: Mesh size $x : y = 80, 160$

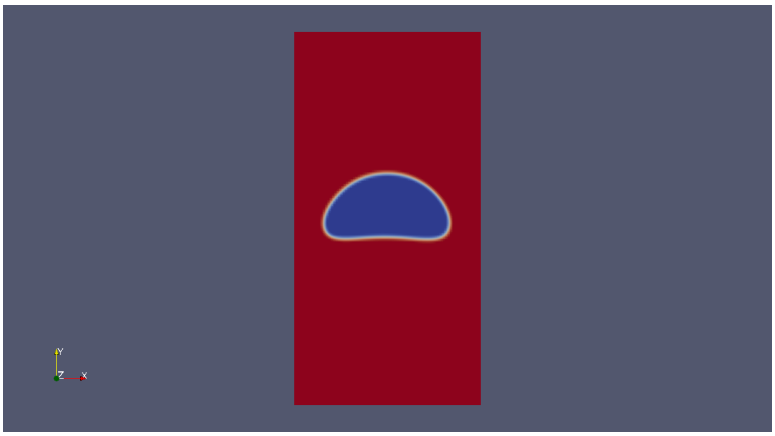

Figure 5.14: bubble at $t = 3s$ with relaxation factor 1.053. Mesh size $x : y = 80, 160$

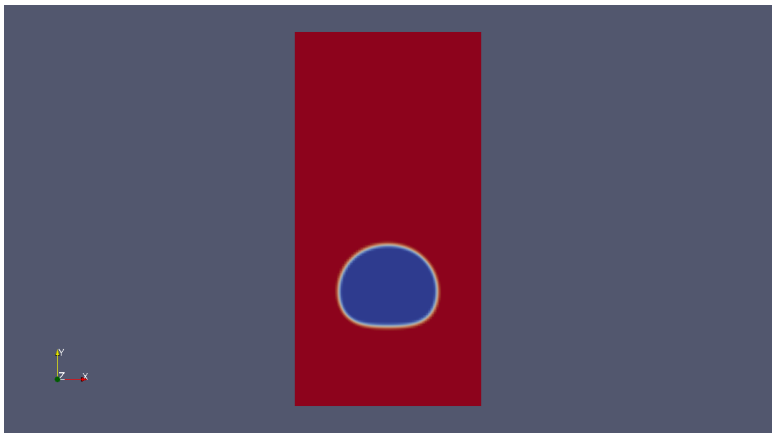

Figure 5.15: Bubble at $t = 1s$ with relaxation factor 1: Mesh size $x : y = 80, 160$

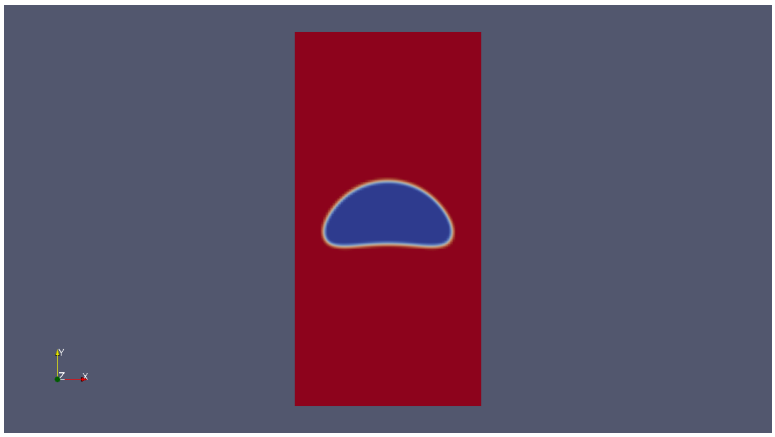

Figure 5.16: bubble at $t = 3s$ with relaxation factor 1. Mesh size $x : y = 80, 160$

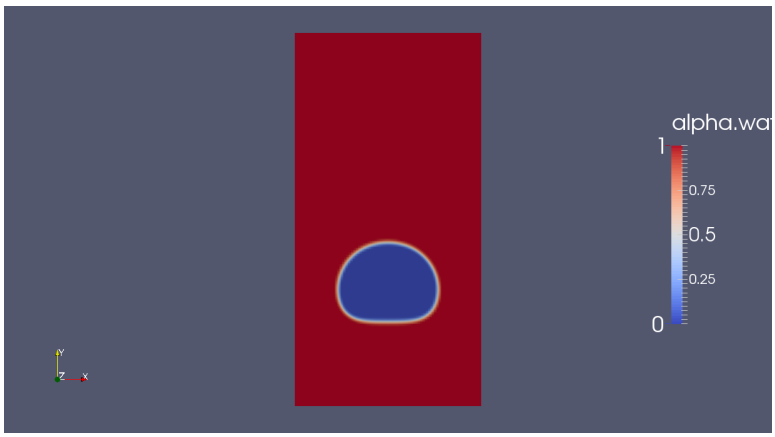

Figure 5.17: Bubble at $t = 1s$ with relaxation factor 1.050: Mesh size $x : y = 80, 160$

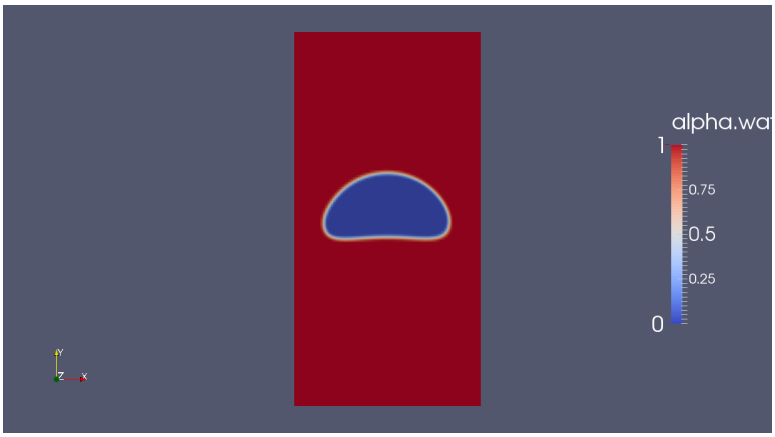

Figure 5.18: bubble at $t = 3s$ with relaxation factor 1.050. Mesh size $x : y = 80, 160$

We have seen the effect of the relaxation factor on the rising velocity and on the center of mass of the bubble. There are two other parameters that can affect the solution: the number of correction in the PIMPLE and SIMPLE algorithms on OpenFoam. Recall that the PIMPLE algorithm is the combined SIMPLE and PISO algorithm. It was designed to take advantage of the fast convergence of the SIMPLE algorithm together with the accuracy of the PISO algorithm. The algorithm will run only in the PISO mode if nCorrectors is greater than 1, while it will run in SIMPLE mode when nOuterCorrectors is greater than one. From numerical experiments whose results will be shown later, it was best to only run the solver in the PISO mode. The reason for that is that PISO was designed for unsteady state while SIMPLE, even though it can be used for unsteady-state problems, was originally designed for steady-state problems.

	α_p, α_U	nCorrectors	nOuterCorrectors	U_{max}	y_{max}	Residual	time
present work	1.052	100	1	0.25	1.1	$9.8.10^{-8}$	484
present work	1.052	50	1	0.25	1.1	$9.8.10^{-8}$	284
present work	1.052	30	1	0.25	1.1	$9.9.10^{-8}$	206
present work	1.052	10	1	0.2416	1.08882	$2.5.10^{-6}$	106
present work	1.052	5	1	0.2191	1.03943	$4.7.10^{-5}$	62
Hysing [19] 1	–	–	–	–	0.2419	1.0812	
Hysing [19] 2	–	–	–	–	0.2421	1.0799	
Hysing [19] 3	–	–	–	–	0.2417	1.0817	
Klostermann [16]	–	–	–	–	0.2365	1.0668	

Table 5.3: Simulation parameters and results for changing relaxation factor α. Mesh size for group 1,2,3 from [19] is $x : y = 160, 160$. Mesh size from [16] is $x : y = 160, 160$. Mesh size from this work is $x : y = 80, 160$. The above-mentioned time is the execution time(CPU time) and is measured in seconds (s)

	α_p, α_U	nCorrectors	nOuterCorrectors	U_{max}	y_{max}	Residual	time
present work	1.052	50	200	0.232	1.06634	7.10^{-8}	512
present work	1.052	1	200	0.2384	1.07775	$4.8.10^{-7}$	370
present work	1.052	1	100	0.2384	1.07775	$4.8.10^{-7}$	368
Hysing [19] 1	–	–	–	–	0.2419	1.0812	
Hysing [19] 2	–	–	–	–	0.2421	1.0799	
Hysing [19] 3	–	–	–	–	0.2417	1.0817	
Klostermann [16]	–	–	–	–	0.2365	1.0668	

Table 5.4: Simulation parameters and results for changing relaxation factor α. Mesh size for group 1,2,3 from [19] is $x : y = 160, 160$. Mesh size from [16] is $x : y = 160, 160$. Mesh size from this work is $x : y = 80, 160$. The above-mentioned time is the execution time(CPU time) and is measured in seconds (s)

Tables 5.3 and 5.4 show the influence of the number of corrections in the PISO (nCorrectors) and SIMPLE (nOuterCorrectors) algorithm respectively. For the PISO algorithm the solution does not change after about thirty iterations and the residual is of order 10^{-08}. The maximum velocity of the bubble converges to 0.25 while the maximum centroid converges to 1.1. In the SIMPLE algorithm, the maximum velocity converges to 0.2384 while the maximum centroid converges to 1.07775. Convergence is reached after about approximately 100 iterations and the residual is of the order of 10^{-07}. The CPU time for the SIMPLE is less than the one for the PISO algorithm. Figures 5.19 and 5.20 show the rising velocity and centroid profile with 50 PISO corrections and 50 SIMPLE corrections respectively.

In the study of the rising bubble problem, several parameters affected the solution. The main parameter is the relaxation factor. Typically, the relaxation factor is between 0 and 1 for the under-relaxation factor, and can

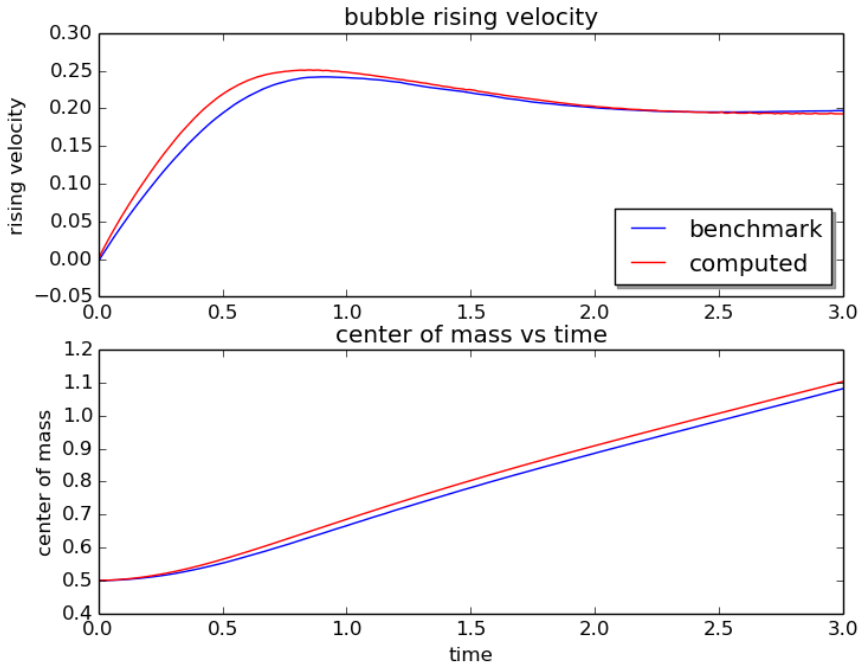

Figure 5.19: PISO algorithm with 50 PISO corrections

be greater than 1 as seen in this study. Other important parameters are the number of PISO correctors nCorrectors, the number of SIMPLE correctors nOuterCorrectors, and the mesh size. The time step can typically be mesh-dependent to maintain a fixed value for the Courant number Co.

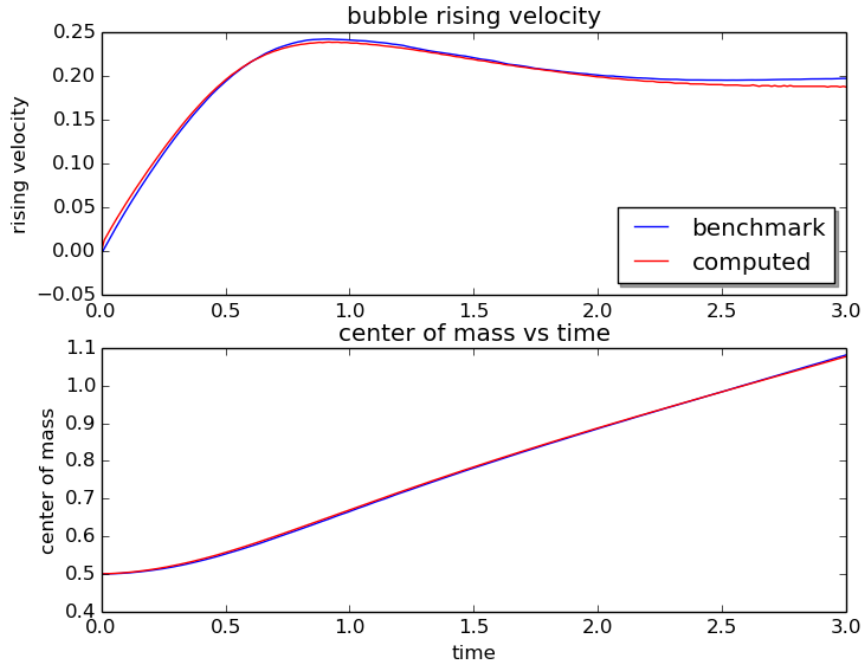

Figure 5.20: SIMPLE algorithm with 50 corrections

5.5 Simulation of fully dispersed two-phase flow

We now turn our attention to a fully dispersed two-phase turbulence flow. The objective is to show the effects of the relaxation parameter on the flow. In the last section we saw how the numerical parameters such as the number of pressure correctors and the relaxation factors affected the rising velocity of the rising single air bubble in liquid water. The fully dispersed two-phase flow is modelled by the two-fluid equation. The turbulence model is the $k-\epsilon$ model. The solutions are obtained using the twoPhaseEulerFoam solver in OpenFoam. The latter is a solver for two incompressible fluids. In this case study we consider air as the dispersed phase and water as the continuous phase. Our attention will center on the value of the phase fraction of air in the internal field.

Geometry of the domain and mesh generation

The simulation domain is a rectangular domain with height $h = 1m$ and width $w = 0.15m$. The mesh is generated by the OpenFoam utility BlockMesh. The domain is made of an hexagon with eight corners whose coordinates are given by the entry vertices. The flow direction is predominantly in the y direction. The mesh is divided by 25 points in the x direction and 75 points in the y directions, specified in the blocks entry. The boundaries of the domain are specified by the entry patches: inlet (bottom), outlet (top) and walls (left and right).

```
vertices
(
    (0 0 0)
    (0.15 0 0)
    (0.15 1 0)
    (0 1 0)
    (0 0 0.1)
    (0.15 0 0.1)
    (0.15 1 0.1)
    (0 1 0.1)
);
blocks
(
    hex (0 1 2 3 4 5 6 7) (25 75 1) simpleGrading (1 1 1)
);
patches
(
    patch inlet((1 5 4 0))
    patch outlet((3 7 6 2))
    wall walls((0 4 7 3)(2 6 5 1))
);
```

Initial and boundary condition

The value of the phase fraction of air alpha.air is fixed at the inlet to 0.5 initially. At the outlet (top) there is only air: alpha.air is equal to 1. Slip-conditions are applied at the wall. The pressure, velocity, turbulence kinetic energy k and rate of dissipation ϵ are fixed at the boundaries (inlet, outlet and wall).

```
alpha.air
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0.5;
    }
    outlet
    {
        type            inletOutlet;
        phi             phi.air;
        inletValue      uniform 1;
        value           uniform 1;
    }
    walls
    {
        type            zeroGradient;
    }
}
```

```
pressure p
dimensions          [ 1 -1 -2 0 0 0 0 ];
internalField       uniform 1e5;
boundaryField
{
    inlet
    {
        type                fixedFluxPressure;
        value               $internalField;
    }
    outlet
    {
        type                fixedValue;
        value               $internalField;
    }
    walls
    {
        type                fixedFluxPressure;
        value               $internalField;
    }
}
```

```
velocity of air U.air
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0.1 0);
boundaryField
{
    inlet
    {
        type                fixedValue;
        value               $internalField;
    }
    outlet
    {
        type                pressureInletOutletVelocity;
        phi                 phi.air;
        value               $internalField;
    }
    walls
    {
        type                fixedValue;
        value               uniform (0 0 0);
    }
}
```

```
kinetic energy air k.air
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 3.75e-5;
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }
    outlet
    {
        type            inletOutlet;
        phi             phi.air;
        inletValue      $internalField;
        value           $internalField;
    }
    walls
    {
        type            kqRWallFunction;
        value           $internalField;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

```
rate of dissipation epsilon.air
dimensions      [0 2 -3 0 0 0 0];
internalField   uniform 1.5e-4;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }

    outlet
    {
        type            inletOutlet;
        phi             phi.air;
        inletValue      $internalField;
        value           $internalField;
    }
    walls
    {
        type            epsilonWallFunction;
        value           $internalField;
    }

    defaultFaces
    {
        type            empty;
    }
}
```

Physical properties

```
phases (air water);
air
{
    diameterModel   isothermal;
    isothermalCoeffs
    {
        d0              3e-3;
        p0              1e5;
    }
}
water
{
    diameterModel   constant;
    constantCoeffs
    {
        d               1e-4;
    }
}
```

```
Turbukence property
simulationType  LES;
LES
{
    LESModel continuousGasKEqn; //Smagorinsky;

    turbulence      on;
    printCoeffs     on;

    delta           cubeRootVol;

    cubeRootVolCoeffs
    {
    }
}
```

```
thermophysicalProperties.air
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        nMoles      1;
        molWeight   28.9;
    }
    thermodynamics
    {
        Cp          1007;
        Hf          0;
    }
    transport
    {
        mu          1.84e-05;
        Pr          0.7;
    }
}
```

```
drag
(
    (air in water)
    {
        type            SchillerNaumann;
        residualAlpha   1e-6;
        residualRe      1e-3;
        swarmCorrection
        {
            type        none;
        }
    }

    (water in air)
    {
        type            SchillerNaumann;
        residualAlpha   1e-6;
        residualRe      1e-3;
        swarmCorrection
        {
            type        none;
        }
    }

    (air and water)
    {
        type            segregated;
        residualAlpha   1e-6;
        m               0.5;
        n               8;
        swarmCorrection
        {
            type        none;
        }
    }
);
```

```
virtualMass
(
    (air in water)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }

    (water in air)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }
);
```

```
sigma: surface tension
(
    (air and water)     0.07
);

aspectRatio
(
    (air in water)
    {
        type            constant;
        E0              1.0;
    }

    (water in air)
    {
        type            constant;
        E0              1.0;
    }
);
```

Finite volume scheme

The finite volumes used are first order in time and second order in space.

```
ddtSchemes
{
    default          Euler;
}

gradSchemes
{
    default          Gauss linear;
}

divSchemes
{
    default                          none;

    div(phi,alpha.air)               Gauss vanLeer;
    div(phir,alpha.air)              Gauss vanLeer;

    "div\(alphaRhoPhi.*,U.*\)"       Gauss limitedLinearV 1;
    "div\(phi.*,U.*\)"               Gauss limitedLinearV 1;

    "div\(alphaRhoPhi.*,(h|e).*\)"   Gauss limitedLinear 1;
    "div\(alphaRhoPhi.*,K.*\)"       Gauss limitedLinear 1;
    "div\(alphaPhi.*,p\)"            Gauss limitedLinear 1;
    "div\(alphaRhoPhi.*,k.*\)"       Gauss limitedLinear 1;

"div\(alpha.**thermo:rho.*\)*nuEff.*\)*dev2\(T\(grad\(U.*\)"
    Gauss linear;
}

laplacianSchemes
{
    default          Gauss linear uncorrected;
}

interpolationSchemes
{
    default          linear;
}
```

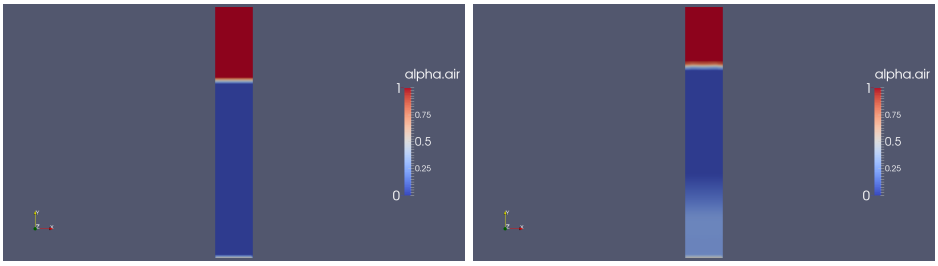

Figure 5.21: Phase fraction alpha of air at $t = 0,1s$

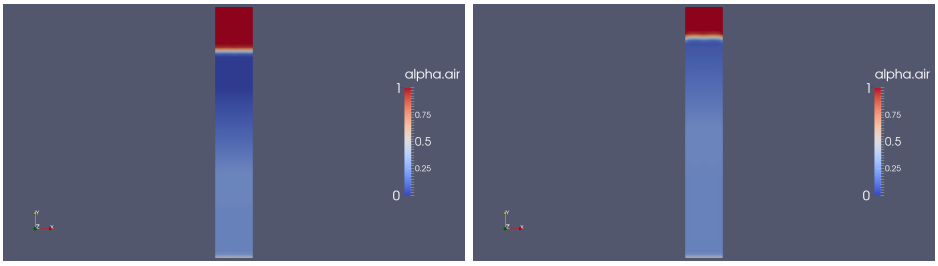

Figure 5.22: Phase fraction alpha of air at $t = 2,3s$

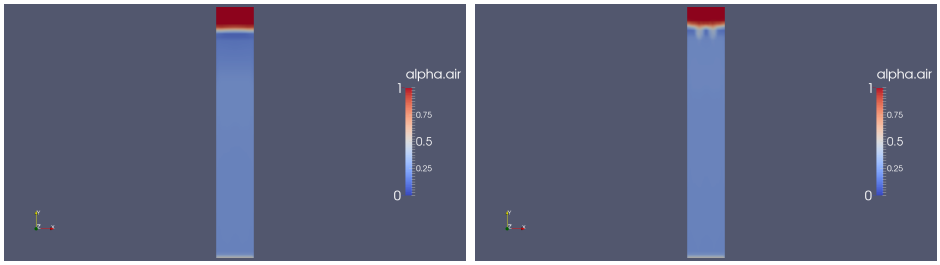

Figure 5.23: Phase fraction alpha of air at $t = 4,5s$

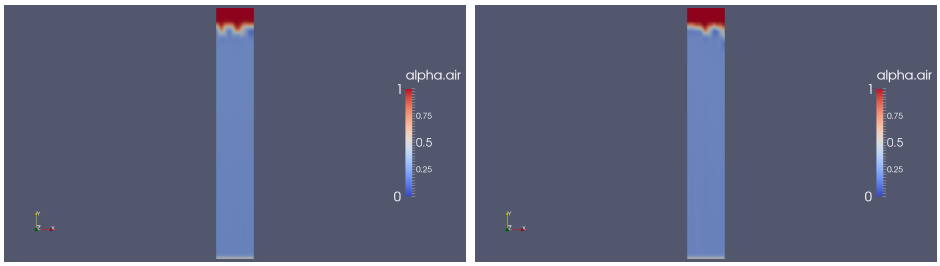

Figure 5.24: Phase fraction alpha of air at $t = 7,10s$

$alpha$	0.20	0.23	0.24	0.25	0.26	0.27	0.28	0.5	0.95
$\|p\|$	0.76	0.13	0.16	0.20	0.20	0.010	0.57	5.6	1.4
Ratio R	1.7	0.91	0.89	0.87	0.87	0.93	1.5	-49.9	-0.37
Asymtotic range	0.83	0.90	0.90	0.90	0.90	0.90	0.86	-0.95	-1
CGI_1	62	-133	-110	-84	-197	63	0.86	14	45
CGI_2	30	-132	-112	-88	-189	37	-50	-0.26	-50

Table 5.5: Effect of relaxation factor on phase fraction, $nCorrector = 1$, $nOuterCorrectors = 10.\Delta t = 0.5/meshsize$.

Table 5.5 shows the effect of the relaxation factor on phase fraction of air. We recall here that for a monotonic convergent solution the ratio R must be positive according to the Grid Convergence Index (GCI) method [22, 21]. From the numerical experiment performed and given in table 5.5, the solution is monotonic convergent when the relaxation factor is between 0.2 and 0.3.

5.6 Parameters optimisation strategy

The solution process of the two cases studied involves an algorithmic process, where certain parameters controlled the accuracy of the solutions. Among other parameters, were the number of pressure corrections needed to achieve a certain level of accuracy. In OpenFoam, in particular in the solvers interFoam and twoPhaseEulerFoam used in this thesis, we have two such parameters: nCorrectors, which is the number of pressure corrections in the PISO algorithm, and nOuterCorrectors, which is the number of pressure corrections in the SIMPLE Algorithm. We have shown that for a more accurate solution of an unsteady state differential equation the PISO mode of the PIMPLE algorithm was preferable. Recall that the PIMPLE algorithm implemented in the InterFoam solver can run either in the PISO mode, the SIMPLE mode or in both modes. Another parameter controlling the solutions was the relaxation factor. This factor is typically between zero and one, but can also be greater than one. It is very difficult do derive a general strategy to find an optimum value for the relaxation factor. In fact, optimum values for under-relaxation are flow/application specific [59]. In the two-phase dispersed flow case study, we saw that the mesh size affected the accuracy of the solution. There exists an explicit relation between the mesh size dx and the time step dt through the Courant number Co:

$$Co = u\frac{dt}{dx}$$

where u is the velocity. To maintain stability the Courant number must be less then a maximum value, typically 1. Table 5.6 summarises the numerical parameters affecting the flow field in our case study. b is any integer greater than 1, while δ is a small number between 0 and 1.

parameters	mesh size	nCorrectors	nOuterCorrectors	relaxation factor	dt
Values	user specified	$[1, b]$	$[1, b]$	$[0, 1+\delta]$	$-$

Table 5.6: Numerical parameters affecting solution and possible range

Based on the analysis of the SIMPLE/PISO algorithms and the two applications considered, a parameter optimisation strategy or algorithm can be formulated as follows:

1. Take initial values from user: mesh size, nCorrectors, nOuterCorrectors, time step, relaxation factor, case directory name, tolerance.

2. if tolerance is small (10^{-2}) use SIMPLE:

$$nCorrectors = 1$$

$$nOuterCorrectors = \max(nCorrectors, nOuterCorrectors)$$

else use PISO mode:

$$nCorrectors = 1$$

$$nOuterCorrectors = \max(nCorrectors, nOuterCorrectors)$$

set all field (pressure, velocity, phase fraction) relaxation factors to 1

3. Choose step according to the maximum value of nOuterCorrectors or nCorrectors and decrease the corresponding value (nOuterCorrectors or nCorrectors) by the step sequentially. For each step make sure that the computed residual is less then the tolerance given by the user. Do this until the the tolerance limit is reached. At the end of this step the minimum values of nCorrectors and nOuterCorrectors are returned.

4. Find the optimum value of the relaxation factor. Use the optimum values of nCorrectors and nOuterCorrectors obtained from step 1, 2 and 3. Chose a range of value from 0.1 to 1.5 for the relaxation factor. Sequentially compute the discretisation error for each value in the range of relaxation factors. To select the best possible values the following conditions must be satisfied:

   - the solution must be monotonic convergent: the value of R computed from equation (5.8) must be positive
   - the mesh must be in the asymptotic range of convergence: $ASR \approx 1$. Equation (5.14)
   - the computed convergence rate must be close to its theoretical value. Equation (5.9)
   - the discretisation error for the finer and coarse mesh must be within a tolerance value specified. Equations (5.10), (5.10)

5. (Optional). Chose few mesh points in x and y direction ex: 10,20, 30... max. Use the neural network to make a model of CPU time as a function of mesh points. Use the model to predict the simulation time of the mesh point you want to use.

By applying the above algorithm we found the results given in Tables (5.7), (5.8), (5.9), (5.10), (5.11) and Figures (5.25), (5.26), (5.27) and (5.26) for the interFoam solver applied to the single rising bubble.

	mesh x	mesh y	nCorrector	nOuterCorrector	α^*	CPU time (s)	Residual
Initial	80	80	100	100	1	126.8	$9.8761.10^{-8}$
step 1	80	80	80	1	1	105.75	$9.8761.10^{-8}$
step 2	80	80	60	1	1	84.42	$9.8761.10^{-8}$
step 3	80	80	40	1	1	63.52	$9.8721.10^{-8}$
step 4	80	80	20	1	1	42.41	$1.982.10^{-7}$

Table 5.7: Parameter optimisation strategy result. α^* means relaxation factor for all fields: pressure, velocity and water phase fraction. The tolerance is set to 10^{-5}. Solver: InterFoam, algorithm mode: PISO

	mesh x	mesh y	nCorrector	nOuterCorrector	α^*	CPU time	Residual
Initial	80	80	75	10	1	99.44	$9.8761.10^{-8}$
step 1	80	80	59	1	1	85.39	$9.8763.10^{-8}$
step 2	80	80	43	1	1	67.45	9.875510^{-8}
step 3	80	80	27	1	1	54.27	$9.15.10^{-8}$
step 4	80	80	11	1	1	30.15	$4.1.10^{-6}$

Table 5.8: Parameter optimisation strategy result. α^* means relaxation factor for all fields: pressure, velocity and water phase fraction. The tolerance is set to 10^{-5}. Solver: InterFoam, algorithm mode:PISO

	mesh x	mesh y	nCorrector	nOuterCorrector	α^*	CPU time	Residual
Initial	80	80	10	75	1	95.12	$7.78909.10^{-7}$
step 1	80	80	1	60	1	95.97	$7.70909.10^{-7}$
step 2	80	80	1	45	1	97.55	$6.58861.10^{-6}$
step 3	80	80	1	30	1	94.57	$9.85948.10^{-7}$
step 3	80	80	1	15	1	68.77	$4.1.10^{-6}$

Table 5.9: Parameter optimisation strategy result. α^* means relaxation factor for all fields: pressure, velocity and water phase fraction. The tolerance is set to 10^{-2}. Solver: InterFoam, algorithm mode: SIMPLE

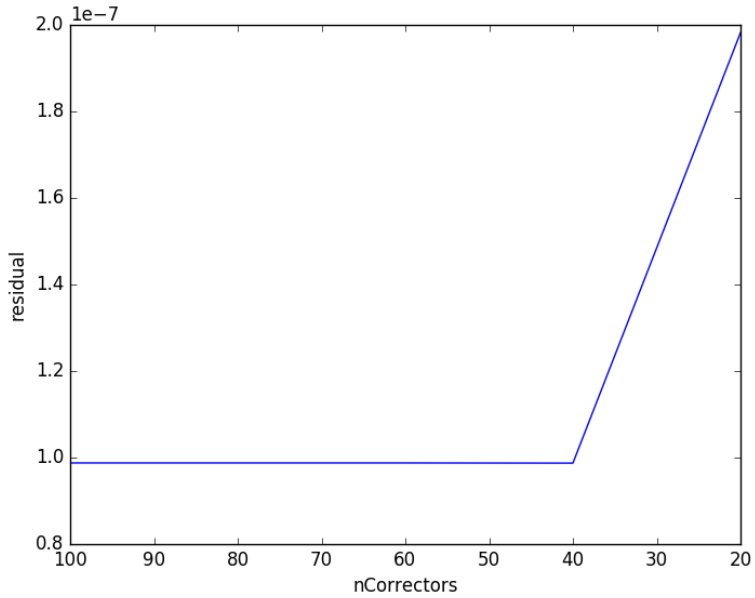

Figure 5.25: Residual as a function of nCorrectors. Graphic representation of table 5.7. The solver InterFoam was running on PISO mode

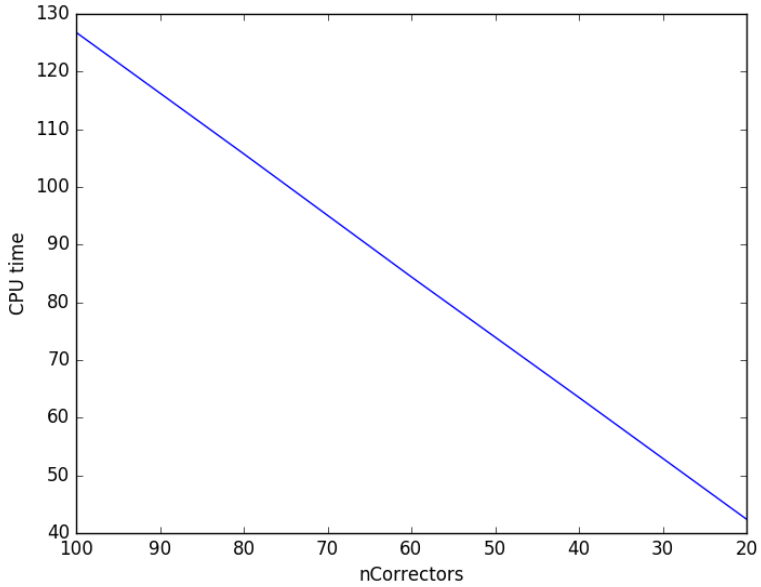

Figure 5.26: CPU time as a function of nCorrectors. Graphic representation of table 5.7. The solver InterFoam was running on PISO mode

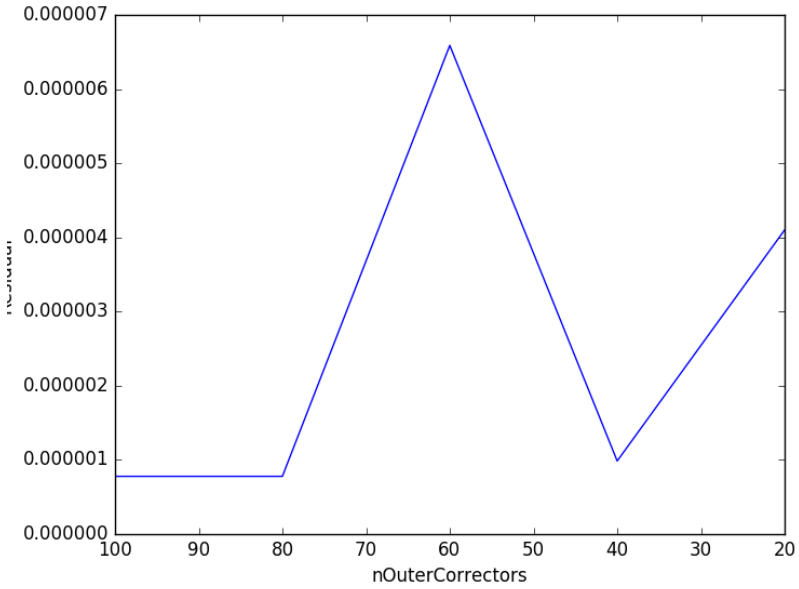

Figure 5.27: Residual as a function of nOuterCorrectors. Graphic representation of table 5.9. The solver InterFoam was running on SIMPLE mode

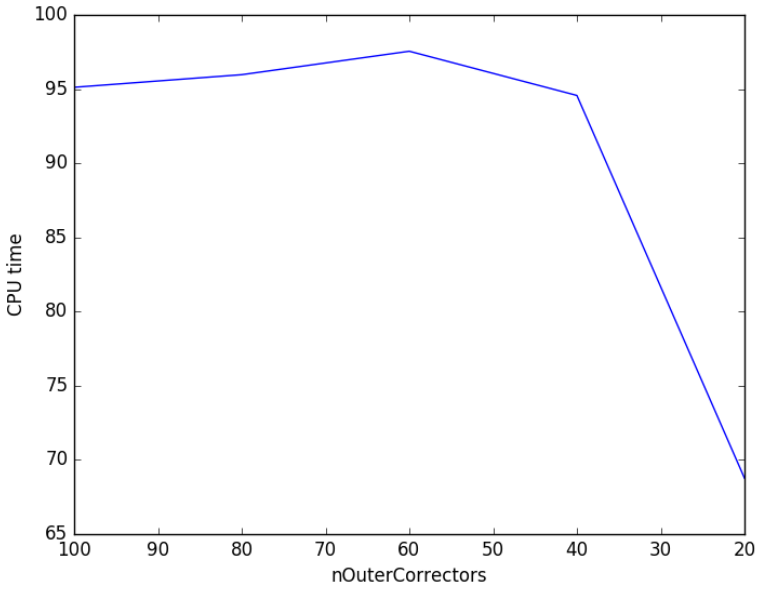

Figure 5.28: CPU time as a function of nOuterCorrectors. Graphic representation of table 5.9. The solver InterFoam was running on SIMPLE

α^*	GCI_2	GCI_1	Asymptotic range	spatial convergence rate	ratio change (5.7)
0.9	25.2	-22.1	-0.97	0.23	-0.85
1	0.359	-0.019	-0.99	4.2	-0.053
1.053	2.3	0.5	1.014	2.136	0.23
1.06	2.6	0.62	1.016	2.031	0.24
1.07	3.14	0.85	1.018	1.85	0.28
1.08	4.47	1.55	1.02	1.5	0.35

Table 5.10: Application of grid convergence index method. α^* means that the relaxation factor was applied to all equations. We used second order schemes in space and first order schemes in time for stability. $nCorrector = 11, nOuterCorrector = 1$.Based on three grids of size: 20,40,80 with $r_{21} = r_{23} = 2$. InterFoam solver

relaxation factor	0.6	0.7	0.8	0.9	1	1.01
Residual	$4.2.10^{-6}$	$4.2.10^{-6}$	$4.2.10^{-6}$	$4.2.10^{-6}$	$4.2.10^{-6}$	$4.2.10^{-6}$

Table 5.11: Relaxation factor with corresponding residual values. $nCorrectors = 11$, $nOuterCorrectors = 1$,mesh size $x = 80$, mesh size $y = 80$. InterFoam solver

mesh size	CPU time (s)	Approximated CPU time (s)	error
35	4	2.8	1.2
65	15	9.8	5.2
151	175	42	133

Table 5.12: CPU time and approximated CPU time. Result obtained by applying the neural network to predict mesh size as a function of CPU time.

5.7 Discussion

Table (5.7) and (5.8) show the optimisation process of nCorrectors and nOuterCorrectors from applying step 1-3 of the optimisation strategy. Since the tolerance is set to 10^{-5} in both cases, the optimisation algorithm forces the interFoam solver to run in the PISO mode. On the other hand, in Table 5.8 the tolerance is set to 10^{-2}, which forces the interFoam solver to run in the SIMPLE mode. By comparing the residuals from the solvers we see that the PISO mode produces lower residuals. From Figures (5.25), (5.26), (5.27) and (5.26) we can see that the residual and the CPU time as a function of nCorrectors follows a monotonic trend for the PISO algorithm. On the other hand, a zigzag trend is observed for the SIMPLE algorithm for residual versus nOuterCorrectors.
Table (5.10) and (5.11) demonstrate that it is not good practice to only rely on the residual provided by the solver. We see that for all relaxation factors ranging from 0.2 to 1 the residual was constant. However, from the Grid Convergence Index method we see that the solution does not converge for relaxation factors between 0.2 and 1. A convergent solution is obtained for relaxation factors in the range 1.053-1.08.
The neural network mentioned on the last step of the algorithm is optional. In fact the results obtained are not satisfactory. The objective was to create a model for predicting future mesh sizes as a function of CPU time. Table (5.5) shows the application of Grid Convergence Index to the simulation of the turbulent fully dispersed two-phase flow. The discretisation error computation is based on the minimum phase fraction of air in the internal field flow. The optimum value for the relaxation factor obtained was around 0.2. The GCI method showed that the coarser grid was better than the finer grid.

The parameter optimisation strategy proposed is solely based on the OpenFoam solvers interFoam and twoPhaseEulerFoam. Both solvers use the PIMPLE algorithm to solve the Navier-Stokes equations and the two fluid model equation respectively. The PIMPLE algorithm can run in SIMPLE mode, PISO mode or both. To run in SIMPLE mode the user must set nOuterCorrectors to a value greater then one and set nCorrectors to one. To run in PISO mode the opposite is true.
The parameters affecting both CPU time and accuracy are then nCorrectors and nOuterCorrectors. Lower values of these parameters result in higher residual. This is because these parameters represent the number of pressure corrections, and more correction means more accurate solution. The relaxation factor however seems to affect much more the accuracy of the solution than the CPU time: the latter is flow/application dependent. The turbulent two-phase flow simulation is less accurate then the laminar two-phase flow. The mesh size is affecting the accuracy and the CPU time. This can be seen from the Grid Convergence Index method, where the finer grid shows

more accuracy than the coarser grid. However, from the simulation of the fully dispersed turbulence flow the opposite was true. This is perhaps due to the complexity of turbulence flow. The limitation on the fully dispersed two-phase flow was due to the fact that the physical parameters such as phase diameter were kept at a constant value. It is well-known that drag forces are affected by the size of the phases, such as air particle diameter. This could explain the less satisfactory result in terms of discretisation error for the fully dispersed turbulence flow.

Chapter 6

Conclusion

This thesis was set out to explore the effect of numerical parameters on two-phase flow. Particular attention was dedicated to the effect of these parameters on the accuracy and CPU time. An open-source CFD software was used to simulate the rising of a single air bubble in liquid water and a fully dispersed turbulence two-phase flow of air and water. From the simulation results the effect of numerical parameters was shown.
The thesis was also set out to explore the accuracy of solutions in OpenFoam in terms of the residual provided by the solvers, and answer the question of whether the residual was a good representative for the accuracy of the solution. Two solvers were used in this study: interFoam and twoPhaseEulerFoam. The former solves the Navier-Stokes equations and applies the Volume Of Fluid method to track the interface between the phases, while the latter solves the Navier-Stokes equations for each phase and is coupled with the $k - \varepsilon$ model to deal with turbulence.

From the simulation of a single rising air bubble in liquid water, the parameters affecting accuracy and CPU time were the number of correctors in the OpenFoam solver algorithm PIMPLE: nCorrector and nOuterCorrector. The PIMPLE algorithm includes two parameters that allow the solver to run either in PISO mode or SIMPLE mode. The PISO algorithm was originally designed for unsteady state problems, while the SIMPLE algorithm was originally designed for steady-state problems but can solve unsteady state problems as well. To run the PIMPLE algorithm in PISO mode, the parameter nCorrectors must be greater than one and the parameter nOuterCorrector must be set to one. The converse is true for the SIMPLE algorithm. From the simulation results it was shown that the PISO mode of the PIMPLE algorithm performs better than the SIMPLE mode in terms of residual computed from the solver. In both cases, the larger the number of correctors the higher the CPU time, and the lower the residual. This is due to the fact that the correctors nCorrector and nOuterCorrector correct the pressure to improve the solutions.

Another important parameter affecting the accuracy was the relaxation factor. The effect of the relaxation factor on the solution could not be seen from the residual computed from the solvers (interFoam and twoPhaseEulerFoam). In fact, for a range of relaxation factors from 0.2 to 1.05 the residual from the solver was constant for all relaxation factor values. To reveal the importance of the relaxation factor on the solution, the Grid Convergence Index (GCI) method was used. The method is a multi-grid method. It uses three local solutions computed on a coarse, medium and finer grid to compute the Grid Convergence Index. The latter is the percentage of how far the computed solution is away from the zero grid solution.
The GCI method also provides a way to compute the zero grid solution. However this solution is not conservative. It is therefore recommended to use the GCI method to compute the discretisation error. Using the GCI method we saw that a value of 1.05-1.08 provided a better solution for the rising velocity of the bubble. The rising velocity computed was further compared to benchmark data provided from the literature and the results were satisfactory. The residual computed from the solver is therefore not representative of the absolute accuracy of the solution. To test the solution extra methods must be used to evaluate the discretisation error.

The solution computed for the turbulent dispersed two-phase flow obtained from the twoPhaseEulerFoam solver was not as good as the solution computed from the rising bubble problem. In the dispersed flow problem, the discretisation error computed on the phase fraction of air in the two-phase flow was quite large: 62 percent on the finer grid and 30 percent on the coarser grid. This means that on the finer grid the computed solution is 60 percent far away from the zero grid solution, while 30 percent away on the coarse grid.

In order to effectively optimise the numerical parameters involved in this study an optimisation strategy/algorithm based on the performance of the PIMPLE algorithm was derived: (i) run the PIMPLE algorithm on the PISO mode to achieve better convergence, (ii) use the GCI method to sequentially find the optimum relaxation factor.
In the course of this study a neural network was used to test the possibility to predict CPU time for a chosen mesh size. This would give the user the flexibility to choose appropriate mesh size depending on the computational power available. The prediction power of the neural network was limited for values far away from the one used to train the network.

One of the limitations of this study was the fact that the turbulence model only considers the effect of numerical parameters on the solutions. Considerations should have also been given to the physical parameters such as

the size of the phase diameters. This would have allowed a more general setting for the parameter optimisation. The parameter optimisation study performed in this thesis was limited to few numerical parameters.
A natural extension of this thesis is to study both numerical and physical parameters and provide an optimisation strategy by considering all numerical and physical parameters. Ultimately, an interesting extension would be to incorporate the optimisation at the solver level: this means redesigning the solvers to incorporate automatic discretisation error computation and numerical and physical parameter optimisation capabilities.

Bibliography

[1] H.k. Versteeg, W. Malalasekera *An Introduction to Computational Fluid Dynamics. The Finite Volume Method*. Second edition, Pearson, London, 2007

[2] Suhas V. Patankar Numerical Heat Transfer and Fluid Flow *Series in computational methods in mechanics and thermal sciences, HEMISPHERE PUBLISHING CORPORATION, Washington, New York London*

[3] Patankard S.V, Spalding, D.B A calculation procedure for heat, mass and momentum transfer in thee-dimensional parabolic flows *Int.J of Heat and Mass Transfer, Volume 15,issue 10, October 1972, Page 1787-1806*

[4] H.Rusche *Computational Fluid Mechanics of Dispersed Two Phase Flows at Higher Phase Fraction*. PHD thesis, University of London, 2002.

[5] S.K Jordan and J.E Fromm *Fondamentals concerning wave loading on offshore structures*. Phys. Fluids, 15(6):972-976, 1972

[6] Newton I *Philosophiae Naturalis: Principia mathematica*. Colloniae Allobrocum, Sumptibus CI. and Ant.Philibert Bibliop., Roma, 1760

[7] Stokes G.C *ON the effect of internal friction of fluids on the motion of pendulums*. Trans. Camb. Phil. Soc., 9;8-27,1851

[8] Smolianski A. Haario H. Luukka P *Vortex Shedding Behind a Rising Bubble and Two-Bubble Coalescence: A numerical Approach*. Zurich, Lappeenranta.

[9] T. Ma., T.Ziegenhein., D.Lucas.,E.Krepper.,J.Frohlich *Euler-Euler large eddy simulation for dispersed turbulent bubbly flows*. International Journal of Heat and Fluid Flows 56 (2015) 51-59

[10] A. Tomiyama., H.Tamai., I.Zun., S. Hosokawa *Transverse migration of a single bubble in a simple shear flow* . Chem. Eng. Sci. 57, 1849

[11] M. Schmidtke *Investigation of the Dynamics of Fluid Particles Using Volume of Fluid Method*. PHD-Thesis, Universitat Paderborn

[12] G. Segre And A. Silberberg *Radical particle displacement in Poiseuille flow of suspensions.* Nature, 189:209-210, 1961

[13] G. Segre And A. Silberberg *Behaviour macroscopic rigid spheres in Poiseuille flow.* J. Fluid Mech., 14:115-157, 1962

[14] P. G. Saffmanm *The lift on a small sphere in a slow shear flow.* J. Fluid Mech. (1965), vol. 22, part 2,pp.385-400

[15] J.L.M Poiseuille *Recherches sur le mouvement du sang dans les vein capillaire.* Mem. de l ' academide des Sciences, 7:105-175,1841

[16] Klostermann J,. Schaake K,. Schwarze R Numerical simulation of a single rising bubble by VOF with surface compression *International Journal for Numerical Methods in Fluid*, 2012.

[17] Clift R., Grace J.R Weber M.E Bubble, Drops and Particles *Brown University. Report GEOFLO/5, DOE/ET/27225-8,Rhode Island, USA*, 1981.

[18] Hirt. C.W,. Nichols B.D Volume of Fluid (VOF) Method for the Dynamics of the Free Boundaries *Journal of Computational Physics 39;201-225(1981)*

[19] Hysing S. Turek S. Kuzmin D. Parolini N. Burman E. Ganesan S. Quantitative benchmark Computation of two-dimensional bubble dynamics *International Journal for Numerical Methods in Fluid 2009; 60:1259-1288*

[20] OpenFOAM. Thee Open Source CFD Toolbox *User Guide. Version 3.0.1 13th December 2015*

[21] Roache P.J Perspective: Method for Uniform Reporting of Grid Refinement Studies *Journal of Fluids Engineering 406 Vol. 116 September 1994*

[22] Bodvarson, Eca L,. Hoekstra M. A verification Exercise for Two 2-D Steady incompressible Turbulence Flows *European Congress on Computational Methods in Applied Science and Engineering. CCOMAS 2004*

[23] Richardson L.F ,The Approximation Arithmetical Solution by Finite Differences of Physical Problems involving Differential Equations, with an Application to the Stress in a Masonry Dam *Trans. Roy. Soc. Lond. Ser. A 210, 307-357*

[24] Richardson L.F ,The deferred approach to the limit *Trans. Roy. Soc. Lond Ser A 226, 229-361*

[25] Kranke J. Frank W. , Application of generalised Richardson extrapolation to the computation of the flow across an asymmetric street intersection *Journal of Wind Engineering and Industrial 96(2008)1616-1628 ELSEVIER*

[26] Celia M. A, Gray W. G Numerical Methods for Differential Equations. Fundamental Concepts for Scientific and Engineering Applications. *Prentice Hall, New Jersey. 1992*

[27] Brenner C. Susanne, Scott L. Ridgway The mathematical theory of finite element methods. *Springer, Third edition, New York*, 2008.

[28] Edwin K.P Chong. Stanislaw H Zak An introduction to Optimisation *Wiley-Interscience Series in Discrete Mathematics and Optimisation, Second edition, 2001.*

[29] Balaz Csanad Csaji Approximation with artificial neural networks *Master thesis. Faculty of science Eotvos Lorand University of Hungary. 20011.*

[30] Bringedal. C Linear and nonlinear convection in porous media between coaxial cylinder. Master of science thesis in applied computational mathematics, University of Bergen, 2011.

[31] Schulkes R. Introduction to multiphase in pipe. *Lecture Note.*

[32] Hill D. The computer simulation of dispersed two phase flows. *Doctorate thesis, Imperial College of science and technology, London.*

[33] Wikipedia the free encyclopaedia
https://en.wikipedia.org

[34] Chetveryk, H. Analytical and numerical modeling of cold water injection in to Horizontal reservoir,. *Geothermal training program, The United Nation University, Iceland, 2000, Report number 4*

[35] Daryl L. Logan A first course in finite element method. *THOMSON, Canada, Fourth edition*, 2007.

[36] Edwards A.L Trump: A computer program for transient and steady state temperature distribution in multi dimensional systems *Report UCRL-14754. Rev. 3. Lawrence Livermore National Laboratory. Livermore CA. USA*, 1972.

[37] Evan, S. L Partial Differential equation,. *AMS, RHodes Island., VOLume 19* 2000

[38] Goffin A.J.J; Rajadas J;Filler G.G Interfacial flow processing of collagen *American Chemistry Society*, 2009.

[39] Osher Level set methods: An overview and some recent results, *2000*

[40] Mckee, S. Tome M.F; Ferreira V.G Ciminato J.A The MAC method *Computers And Fluid 37 (2008) 907-930*

[41] Weston Ben A marker and cell solution of the incompressible solution of Navier-Stokes equation for the free surface. *Numerical Analysis Report 6 2000*

[42] Osher, S., and Sethian J.A Front propagating with curvature dependent speed: Algorithms based on the Hamilton-Jacobi formulation *Comput. Phys. 79,12-49 (1998)*

[43] Harlow F.H and Welch J.E Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with free Surface. *Physics of Fluids 8, 2182 (1965).*

[44] Gudmudsson, J.S and G. Olsen,S. Thorhalsson, 1985 Svartsengi Field Production and Depletion Analysis *, Proceeding, Tenth Workshop on Geothermal Reservoir Engineering, Stanford University, Stanford, California , January 22-24, 1985*

[45] Gudmudsson, J.S and G. Olsen, 1987 Water-influx modelling of the Svartsengi Geothermal field Iceland. *,SPE Reservoir Engineering (Feb.1987),77-84*

[46] Hagdu T, Zimmerman R. W, Bodvarsson G Coupled Reservoir-Wellbore Simulation of Geothermal Reservoir Behavior. *Geothermics, Vol, 24, No. 2.pp. 145-166*, 1995.

[47] Hjartarson A. Bjornsson G. Saemundsson K Stimulation and measurement in well MN 8 in Munadarnes. *A report prepared for Orkuveita Reykjavik by Iceland geosurvey*, 2003.

[48] Holden, H. Risebro, H.H Front tracking for Hyperbolic Conservation Laws,. *Springer, Berlin. New York-USA., 2th edition, 2007*

[49] Kjaran, S:P., G:K. Halldorsson, S. Thorhallson and J. Eliasson, 1979, Reservoir engineering aspects of Svartsengi geothermal area *Geoth. Ressources Concil Trans.,3,337-339.*

[50] Kocabas, I. Geothermal characterization via Thermal injection back flow and inter wells tracer testing,. *Geothermic, 34, 27-46 (2005)*

[51] Logg Anders Automated solutions of differential equations by the finite element methods. *Germany* , 20011.

[52] Narasimhan T.N, Witherspoon P.A An integral finite difference method for analysing fluid flow in porous media. *Water Resour. Res. 12.57-64*, 1976.

[53] Narasimhan Norton D.L Theory of Hydrothermal systems. *Ann. Rev. Earth Planet. Sci 1984. 12:155-77*

[54] Miller C.W Wellbore Users Manual. *Berkley University of California. Report , No LBL-10910, USA* 1980.

[55] Ockendon, J. Howinson, S Applied Partial Differential equations,. *Oxford, New York.,Revised edition*2003

[56] Olver, p.J Equivalence, Invarients, and symmetry, *Cambridge University press* 1995

[57] Osullivan, M.J, Karsten Pruess, Marcelo J Lippman State of the art of geothermal reservoir simulation. *Geothermics 30(2001). 395-429* .

[58] Pritchett J. W STAR: A geothermal simulation system. *Proceedings of the World Geothermal Congress*, 1995.

[59] Barron R.M, and Neyshabouri A. A .S Effect of under relaxation factors on Turbulent flow simulation *int J. Numer. Meth Fluids 2003; 42:923-928*

[60] Pruess Kasten Mathematical modeling of fluid flow and heat transfer in geothermal systems-An introduction in five lectures. *United state University, Iceland, Reykjavik* , 2002.

[61] Sayantan G. Kumar M Geothermal Reservoir- A brief Review *Journal of Geological Society of India. Vol.79, Jun 2012, pp.582-602*

[62] Stark M.A, W.T Box Jr, J.J Beall, K.P Goyal and A.S Pingol, 2005 The Santa Rosa-Geyser Recharge Project . *Geyser Geothermal Field California, Proceedings of the World Geothermal Congress 2005, Antalya, Turkey, April, 9 pp*

[63] Stefansson V., 1997 Geothermal Reinjection experience. *Geothermics, 26, 99-130*

[64] Salas, D.M The curious events leading to the theory of shock waves,. *Invited lecture at the 17th shock International symposium, Rome-Italy*(2006)

[65] Springer reference. Springer. *www.springerreference.com*

[66] Vinsome P.K.W., Shook, G.M Multi-Purpose Simulation. *Petrol Science and Engineering 9(1)29-38.*

[67] Wajnarowski, P. Stapo, S Analytical model of cold water front movement in a geothermal reservoir,. *Geothermics, 35, 59-64* (2006)

[68] Young, H.D. Fredman, R.A University physics,. *Addison Wesley, Boston, New York.,11th edition*2004

[69] Zoback, M.D Reservoir Geomechanics ,. *Cambridge University press, United Kingdom,* 2008

[70] A. Logg., *Injection at the Beowa geothermal reservoir*, Elsvier, USA, 2011.

[71] P. S. Pacheco., *Tracing of injection in the Geysers*, Elsvier, USA, 2011.

[72] Wang K., 2005 Studies of the Reinjection Tests in Basement Geothermal Reservoir, Tianjin, China. *Proceedings of the World Geothermal Congress 2005, Antalya, Turkey, April, 12 pp.*

[73] Woods. John,. P. K nobloch., *vaporisation of a liquid moving front moving through a hot rock porous medium*, Advance in computation mathematics, USA, 2007

[74] Whiting, R.L,Ramey, H.J., *Application of Material and Energy Balances to Geothermal Steam production*, J.Pet.Tech. (July,1989) 893-900

[75] Iceland Geosurvey Iceland Geosurvey webpage *www.isor.is*

[76] US Department of energy Energy Efficiency and Renewable Energy *www1.eere.energy.gov*