University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Risk Management/Offshore Safety	Spring semester, 2013 Open
Writer: Martin Hagenes Solheim	…… (Writer's signature)
Faculty supervisor: Bjørn Helge Hjertager External supervisor(s):	
Title of thesis: CFD study of gas-liquid flow from a subsea gas release	
Credits (ECTS): 30	
Key words: Bubble plume Bubble column Two phase flow CFD OpenFOAM bubbleFoam twoPhaseEulerFoam	Pages: 61 + Enclosure: 25 + CD Stavanger, 14.06.2013

Abstract

The worlds demand for oil and gas is constantly increasing, and this has lead the exploration into deeper water. Introducing exploration in deep-water, poses new challenges when it comes to consequence modeling of blowouts or releases of gas from subsea pipelines. It is important to understand how the gas will disperse as it moves up through the water in what is recognized as a bubble plume. In todays practice, the fate of the released gas is often based on old experiments (Fanneløp and Sjøen, 1980) and a number of assumptions. These experiments suggest a good understanding of what goes on in shallow water, but there are a number of factors that can complicate the situation in deep-water. These assumptions are usually recognized as legit for water depths up to 300m, although this limit is questionable. As the exploration moves into water depths significantly deeper than 300 meters, these assumptions cannot be considered legit any more. In the industry, methods based on these assumptions are being used for large transient releases and/or deep-water. When consultancy firms are simulating scenarios of subsea releases of gas, they will either use formulas based on Fanneløp and Sjøen (1980), or integral models for the subsea part of the release. When the gas reach the surface they will normally use numerical simulation for the spreading of the released gas. As the computer hardware is developing rapidly, the potential of numerical simulation using computational fluid dynamics (CFD) tools is increasing. The CFD codes have also been undergoing a lot of research and development in the later years. When considering this, it could be of interest to evaluate whether numerical simulation can be used for the gas-liquid flow resulting from underwater blowouts or pipeline breakage.

The objective of this thesis was to use the open source CFD code OpenFOAM to set up an ideal bubble plume for comparison against experiment. I wanted to set up four cases in both 2D and 3D similar to the experiment of Fanneløp and Sjøen (1980) in a 10 meter pool. The parameters of particular interest were vertical velocity, plume radius and concentration of gas. The goal was to see if OpenFOAM could provide realistic results for 10 meter water depth. These results should provide a basis for further discussion on the potential of numerical simulation of bubble plumes. The results of the simulations had some similarities to the experiments. The slip velocity shows promising agreement with the theory. On the other hand, the most important parameters are clearly under-predicted. The spreading of the plume and the entrainment of water is on average about half of what is reported in the experiments. This leads to a higher void fraction, on average a factor of 2. This is believed to be affected by the turbulence model used, the boundary conditions and inherent limitations in the `bubbleFoam` solver. Other limitations were the settings set by the user. Also the velocity profile along the centerline is clearly over-predicted in the simulations. This is believed to be affected by the same things mentioned above, but also the constant bubble diameter of 3mm used.

Some alternative approaches are at the end considered to further test the potential of OpenFOAM. The cases are run laminar and with a changed inlet condition. They show a good tendency when comparing against the experiments.

Based on the information gained through this thesis, the conclusion is that we must rely on integral models today and for the foreseeable future, especially for practical use in the industry. The immediate focus should be on lecturing the industry on the clear limitation inherent in the basic relation based on Fanneløp and Sjøen (1980). That being said, research into numerical simulation of bubble plumes should also be prioritized. The challenge is to include non-ideal gas behavior, effects of cross currents, dissolution of gas and so forth. Also models that can include bubble breakage and coalescence should be considered.

Acknowledgements

This thesis is written as part of a Master of Science in Risk Management – Offshore Safety at the Department of Industrial Economics, Risk Management and Planning, University of Stavanger. I would especially like to thank my supervisor at the University of Stavanger, Bjørn Helge Hjertager. With my limited background in fluid dynamics and CFD, it has been extremely helpful with your good insight and advices along the way. In addition I would like to thank Safetec Nordic AS for providing an office place at their office in Stavanger.

Table of Contents

Abstract III
Acknowledgements V
List of Figures IX
List of Tables X
Nomenclature XI
1. Introduction 1
1.1. Background 1
1.2. Previous Work 2
1.3. Objective 2
2. Theoretical Study 3
2.1. Basic Hydrodynamics of Underwater Blowouts 3
2.1.1. Empirical Coefficients 7
2.2. Practical Application 9
2.2.1. Integral Models 10
2.3. Computational Fluid Dynamics - CFD 10
2.3.1. Mathematical Modeling of a Bubble Plume 12
2.4. Experiment (Fanneløp and Sjøen, 1980) 16
3. Case Setup 17
3.1. Introduction to OpenFOAM 17
3.2. bubbleFoam 18
3.2.1. Capabilities 18
3.2.2. Limitations 19
3.2.3. Governing Equations 19
3.2.4. Solution Procedure 23
3.3. Case Description 23
3.3.1. Mesh Generation 24
3.3.2. Boundary Conditions 25
3.3.3. Transport Properties 27
3.3.4. Turbulence Modelling 28
3.3.5. Environmental Properties 28
3.3.6. Discretization Schemes 28
3.3.7. Solution Controls 29
3.3.8. Time Step and Data Output 29

3.3.9. Sampling of Data 30
3.3.10. Running the Code 31
3.3.11. Post-Processing 31
3.4. Parameter Study 31
3.4.1. Bubble Size 32
3.4.2. Lift Coefficient 33
3.4.3. Turbulent Response Coefficient 33
3.4.4. Virtual Mass 34
4. Results and Discussions 35
4.1. General Observations 35
4.2. Velocity 40
4.3. Void Fraction 44
4.4. Plume Radius 47
4.5. Summary and Discussion 48
5. Alternative Approaches 51
5.1. Results 52
5.1.1. Velocity 52
5.1.2. Void Fraction 53
5.1.3. Plume Radius 55
5.2. Summary and Discussion 55
6. Conclusion 57
7. Future Work 59
8. References 60
Appendix A i
A.1 Boundary Conditions i
Appendix B viii
B.1 Transport Properties viii
B.2 Turbulence Properties ix
B.3 Gravitational Acceleration x
B.4 BlocMeshDict-3DMesh xi
B.5 BlockMeshDict-2DMesh xiv
Appendix C xvii
C.1 controlDict xvii
C.2 fvSchemes xx
C.3 fvSolution xxii
C.4 setFieldsDict xxiv

Appendix F xxv
F.1 Content of Encolosed CD xxv

List of Figures

Figure 1 Definition sketch of a time-averaged bubble plume (Friedl and Fanneløp 2000) 3
Figure 2 Sketch of a fountain (detail from Figure 1) (Friedl and Fanneløp 2000) 4
Figure 3 OpenFOAM case structure (Hjertager, 2009) 17
Figure 4 Case structure of the bubbleFoam solver(openfoanwiki.net, 2012) 18
Figure 5 Simulations using bubble size from 1-5mm 32
Figure 6 Simulations using lift coefficients 0, 0.1, 0.5 and 1 33
Figure 7 Simulations using Ct values of 0.5, 0.9, 1, 1.1 and 1.5 34
Figure 8 Simulations using Cvm value of 0.1, 0.4, 0.5, 0.6 and 0.9 34
Figure 9 Initial bubble plume after 4 seconds 36
Figure 10 Bubble plume after reaching surface at respectively 8 and 10 seconds 36
Figure 11 Bubble plume after 14 seconds 37
Figure 12 Bubble plume after 30 seconds 37
Figure 13 gas velocity after 50 seconds 38
Figure 14 gas velocity averaged over time 38
Figure 15 Epsilon values a few seconds after the plume hits the surface 39
Figure 16 k values a few seconds after the plume hits the surface 39
Figure 17 Epsilon values after 50 seconds for Case 2 in 3D 40
Figure 18 Gas velocity at 9m for the 2D cases 40
Figure 19 Liquid velocity at 9m for the 2D cases 40
Figure 20 Liquid velocity at 9m for the 3D cases 41
Figure 21 Gas velocity at 9m for the 2D cases 41
Figure 22 Vertical gas velocity averaged over time for Case 1 42
Figure 23 Vertical gas velocity averaged over time for Case 2 42
Figure 24 Vertical gas velocity averaged over time for Case 3 43
Figure 25 Vertical gas velocity averaged over time for Case 4 43
Figure 26 Void fraction for 2d cases at 9m 44
Figure 27 Void fraction for 3D cases at 9m 44
Figure 28 Void fraction averaged over time for Case 1 45
Figure 29 Void fraction averaged over time for Case 2 45
Figure 30 Void fraction averaged over time for Case 3 46
Figure 31 Void fraction averaged over time for Case 4 46
Figure 32 Plume radius for the Fanneløp experiment (Fanneløp and Sjøen, 1980) 47
Figure 33 Plume radius for cases run in 2D 48
Figure 34 Plume radius for cases run in 3D 48
Figure 35 Entrainment coefficient plotted against flow rate for experiment and 3D cases 49
Figure 37 Laminar flow using `bubbleFoam` and `twoPhaseEulerFoam` 51
Figure 38 Velocity profile of alternative cases at 9m 52
Figure 39 Vertical velocity profile of alternative cases compared against experiment 53
Figure 40 Void fraction of alternative cases at 9m 54
Figure 41 Void fraction along centerline for alternative cases compared against experiment 54
Figure 42 Plume radius of alternative cases compared against experiment 55

List of Tables

Table 1 Shape coefficients for the plume equations (Fanneløp and Bettilini 2007) 7
Table 2 Recommended empirical values (Fanneløp and Bettilini, 2007) 9
Table 3 Constants for the turbulence model 23
Table 4 Mesh data taken from `checkMesh` 25
Table 5 Fluid properties of air and water 27
Table 6 Overview of cases 35
Table 7 Overview of cases run as alternative approaches 52

Nomenclature

Latin

b	Radius	[m]
c	Void fraction	[-]
c_p, c_v	Specific heats	[J/kg K]
Fr	Froude number	[-]
g	Gravitational constant	[m2/s]
H	Pressure depth	[m]
H_o	Source depth w.r.t the surface	[m]
n	Polytrophic exponent	[-]
p	Pressure	[Pa]
Q	Mass flow rate	[kg/s]
r	Horizontal distance from the plume axis	[m]
T	Surface tension	[N/m]
V_o	Gas volume flow at the source, local pressure	[m3/s]
V_g	Gas volume flow at normal pressure	[Nm3/s]
w	Centerline velocity	[m/s]
w_s	Slip velocity	[m/s]
z	Vertical distance from the source	[m]
C_μ	Model constant in the k-epsilon model	[-]
$C_{\varepsilon 1}, C_{\varepsilon 2}$	Constant in turbulence models	[-]
C_D	Drag coefficient	[-]
C_L	Lift force coefficient	[-]
C_{TD}	Turbulent dispersion coefficient	[-]
C_{VM}	Virtual mass coefficient	[-]
d_B	Bubble diameter	[m]
Eo		[-]
G	Production of turbulent kinetic energy	[kg/ms3]
I	Unity tensor	[-]
k	Turbulent kinetic energy	[m2/s2]
M_D	Drag force	[N/m3]
M_I	Total interfacial force	[N/m3]
M_L	Lift force	[N/m3]
M_{TD}	Turbulent dispersion force	[N/m3]
M_{VM}	Virtual mass force	[N/m3]
Re_B	Bubble Reynolds number	[-]
u	Axial component of velocity	[m/s]
F	Force	[N]

Greek

α	Entrainment coefficient	[-]
α	Volume fraction	[-]
φ_i	Shape parameters	[-]
γ	Momentum amplification factor	[-]

Fr	Froude number	[-]
λ	Ratio between the widths of the momentum and buoyancy profiles	[-]
ρ	Density	[kg/m3]
σ_k	Model constant k-epsilon model	[-]
σ_ε	Model constant k-epsilon model	[-]
μ_L	Dynamic viscosity	[kg/ms]
ν	Molecular kinematic viscosity of liquid	[m2/s]
ν_t	Turbulent kinematic viscosity of liquid	[m2/s]
τ	Stress tensor	[N/m2]

Subscripts

a	Atmospheric	[-]
g	Gas phase	[-]
o	Source	[-]
w	Water	[-]
G	Gas phase	[-]
L	Liquid phase	[-]
φ	Either phase	[-]

1. Introduction

1.1. Background

The worlds demand for oil and gas is constantly increasing, and this has lead the exploration into deeper water. Introducing exploration in deep-water, poses new challenges when it comes to consequence modeling of blowouts or releases of gas from subsea pipelines. It is important to understand how the gas will disperse as it moves up through the water in what is recognized as a bubble plume. Critical aspects in this regard is the raise velocity of the gas, whether or not there will be a flammable concentration of gas at the surface, and how large the potential flammable area will be. These factors will influence the risk of fire and explosion hazards at the surface. In todays practice, the fate of the released gas is often based on old experiments (Fanneløp and Sjøen, 1980) and a number of assumptions. These experiments suggest a good understanding of what goes on in shallow water, but there are a number of factors that can complicate the situation in deep-water. The massflux is considered to be conserved, and the gas is considered to expand like an ideal gas. Factors like currents and stratifications in the water masses are considered negligible. These assumptions are usually recognized as legit for water depths up to 300m, although this limit is questionable. As the exploration moves into water depths significantly deeper than 300 meters, these assumptions cannot be considered legit any more. In the industry, methods based on these assumptions are being used for large transient releases and/or deep-water. In many cases there will be added a factor to account for the high initial flow rate, deep-water and the extended lateral flow of air at the surface caused by the high flow rate. Despite this, it is clear that there is a need for better methods to predict the fate of the released gas. Some work have been done in this regard. Sintef has developed an integral model taking account of possible hydrate formation, dissolved gas in seawater, and gas leaking out of the plume (Johansen, 2003, Johansen, 2000). Integral models have proven to predict decent results when comparing against experiments and numerical simulations (Fanneløp and Bettilini, 2007). When consultancy firms are simulating scenarios of subsea releases of gas, they will either use formulas based on Fanneløp and Sjøen (1980), or integral models for the subsea part of the release. When the gas reach the surface they will normally use numerical simulation for the spreading of the released gas. As the computer hardware is developing rapidly, the potential of numerical simulation using computational fluid dynamics (CFD) tools is increasing. The CFD codes have also been undergoing a lot of research and development in the later years. When considering this, it could be of interest to evaluate whether numerical simulation can be used for the gas-liquid flow resulting from underwater blowouts or pipeline breakage. Initially it is of interest to study an ideal case and compare against existing experiments to evaluate the further potential of numerical simulation when it comes to this phenomena. There are a number of CFD codes available today, both commercially and open source. For this thesis the open source CFD code OpenFOAM will be used. In the following subsections I will briefly present previous work that has formed the basis for this thesis. This includes theoretical studies as well as work done with the use of numerical simulation of bubble plumes. Also other works that can contribute to the discussion of the future potential of numerical simulations are presented. This includes work done into integral models. A central subject for further discussion is to look at integral models versus numerical simulation. Can numerical simulation give more accurate results, and is it worth the extra computational time/cost. I might not be able to answer this question in the thesis, but I can produce some relevant data for future discussion. In the end of this chapter I will present a detailed objective of the thesis.

1.2. Previous Work

The most relevant work into plume dynamics, that a lot of later research is based on, is the work done by Fanneløp and Sjøen (1980) and the measurement published by Milgram (1983). Fanneløp and Sjøen performed both a theoretical study as well as experiments in 5 and 10 meter pools with flow rates varying from 0.005Nm3/s to 0.022Nm3/s, while Milgram performed experiments in a 50 meter natural spring with flow rates up to 0.590Nm3/s.

Most of the work done in relation to bubble plumes and numerical simulation is for nuclear engineering, bioreactors in chemical engineering, and separators in oil and gas engineering. E.g. Deen performed in 2001 a computational study of bubble plumes in gas-liquid chemical reactors (Deen, 2001). Smith and Dhotre (2007) performed a CFD simulation of large-scale bubble plumes for comparison against experiments. Some other relevant papers in this regard is among others Zhang et al. (2006) and Tabib et al. (2008). What is common for all these works is that they consider rather small plumes in a small domain, e.g. a domain of 0.15mx0.15mx0.45m. Even though I will consider a considerably larger domain in this thesis, the applicability of these works are still large. They are considering the same phenomena using the same empirical coefficients and parameters. Especially the works of Deen (2001) and Smith and Dhotre (2007) will form the basis for this thesis.

Numerous integral models, or plume models, have been developed in later years. Examples of such models are DeepBlow – a Lagrangian Plume Model for Deep Water Blowouts developed by Sintef (Johansen, 2000), and StarPlume developed by Bettelini and Fanneløp (1993). These models are much simpler than numerical simulations, and therefore more attractive if they can produce good results.

1.3. Objective

The objective of this thesis is to use the open source CFD code OpenFOAM to set up an ideal bubble plume for comparison against experiment. I want to set up four cases in both 2D and 3D similar to the experiment of Fanneløp and Sjøen (1980) in a 10 meter pool. I will then study the important parameters and compare against the experiment. The parameters of particular interest are vertical velocity, plume radius and concentration of gas. The goal is to see if OpenFOAM can provide realistic results for 10 meter water depth. The information gathered should provide a basis for further discussions on the potential of numerical simulation of bubble plumes resulting from offshore blowouts or pipeline breakage, especially in relation to consequence modeling in risk analysis.

2. Theoretical Study

In this chapter I will present the theoretical basis needed to meet the objective of the thesis. This includes the basic hydrodynamics and the most important parameters and coefficients associated with a bubble plume. It is also of interest to look into the different approaches and models used today for prediction of subsea releases. Further I have studied the basics of computational fluid dynamics - CFD, and especially how one can mathematically model a bubble plume by the use of CFD. At the end of the chapter I have shortly presented the experiment that will be used for comparison against the results achieved by using the CFD-tool OpenFOAM. The results from this comparison are presented in chapter 4.

2.1. Basic Hydrodynamics of Underwater Blowouts

This sections is largely based on Friedl and Fanneløp (2000) and Fanneløp and Bettilini (2007), other sources are cited. When gas is released on the sea bottom, either following a blowout or a pipeline rupture, the gas will first form a jet and rise to the surface because of the different densities in gas and water. Gas bubbles will drag along seawater in a turbulent bubble plume that will create a conical shape while rising up through the water masses.

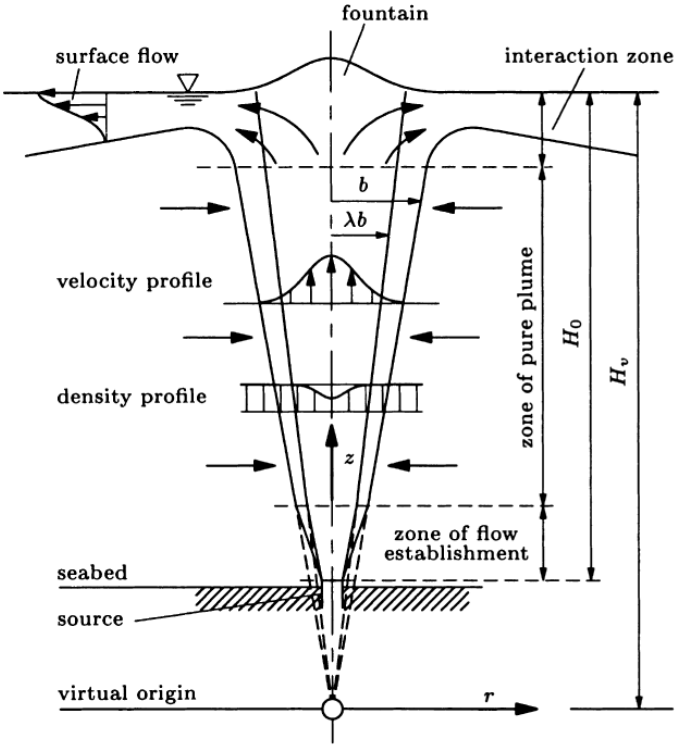

Figure 1 Definition sketch of a time-averaged bubble plume (Friedl and Fanneløp 2000)

In 1980 Fanneløp and colleges executed some of the first theoretical and practical studies of underwater gas plumes and their interaction with the surface, e.g. Fanneløp and Sjøen (1980). These studies are based on some assumptions. The massflux were considered to be conserved, and the gas was considered to expand like an ideal gas. Factors like currents and stratifications in the water masses were considered negligible. These assumptions are for practical application recognized as legit for water depths up to 300m(Tveit, 2006). When the water depth becomes deeper than 300-400m, Johansen (2003) points out some important characteristics that will affect the destiny of the plume. Some of the most important factors are stratification, and that the massflux of gas might not be conserved in the plume. Gas bubbles can diffuse into the water, and gas can create hydrates in contact with seawater under high pressures and low temperatures.

A typical flow pattern for a bubble plume is shown in Figure 1. When the bubble plume is approaching the surface it will generate a surface flow in horizontal direction. This is because of the surface tension between air and the water-gas flow. Generally we can divide the development of a

bubble plume into four categories, dependent on the momentum of the gas release and the buoyancy.

1. The establishment phase. Referred to in Figure 1 as the zone of flow establishment. Near the leak source the buoyancy will be negligible, and the flow is dominated by the following: the initial momentum of the gas, the gas expansion, and the breakup of the gas into bubbles.
2. The transition phase. The flow pattern is dependent on the momentum as well as the buoyancy.
3. Established bubble plume. Referred to in Figure 1 as zone of pure plume. In this zone the flow is primarily dependent on the buoyancy, and the plume velocity profile can be represented by a Gauss distribution. The plume consist of an inner core where most of the bubbles are found and a wider region with substantial upward velocity.
4. The near surface area. Referred to in Figure 1 and Figure 2 as the interaction zone. When the bubble plume approaches the surface the vertical flow will bend over and establish a radial flow away from the center of the plume. This is also referred to as surface flow, shown in Figure 2. We get this flow pattern because the massflux of gas below the surface is larger than the massflux of gas released above the surface.

For deep waters or small release rates the influence of the establishment phase and the transition phase is less important. The rising flow of gas will be dominated by buoyancy.

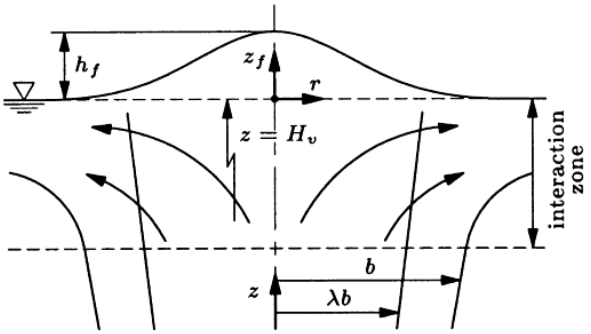

Figure 2 Sketch of a fountain (detail from Figure 1) (Friedl and Fanneløp 2000)

The governing equations for a steady-state bubble plume are expressed in terms of cylindrical coordinates and are integrated over the plume cross-section to give a one-dimensional model. This basic steady state bubble plume model is based on a number of assumptions. The mean flow is stationary and is expressed in terms of time-averaged quantities. Plume rise is considered in stagnant water and the time-averaged flow is axially symmetric and non-rotating. The water is incompressible and modelled as a continuous phase. Stratification due to variations in temperature or salinity is neglected. As the plume is slender, the radial pressure gradient can be neglected. The gas bubbles are modelled as a continuous distribution of void fraction. Since the bubbles are relatively small, their temperature is assumed to be in equilibrium with the environment. It follows that the gas will expand isothermally while rising in the bubble plume. Bubble drag is accounted for by introducing the velocity of the gas phase relative to the liquid phase, i.e. the slip velocity Vs. Vs is assumed to be constant and equal to the terminal speed of bubbles of the same size in stagnant water.

The gas expansion is represented by means of the polytrophic relation. Here z denotes the distance from the source. The index (0) denotes values at the source, and the indicia (g) denotes the gas phase:

$$\frac{\rho_g(z)}{\rho_g(0)} = \left[\frac{p(z)}{p(0)}\right]^{\frac{1}{n}} \quad \textbf{(1)}$$

Which includes the process of particular interest, i.e. the isothermal, isentropic and incompressible flow, obtained with n=1, cp/cv and • respectively. Using the definition:

$$H = H_0 + \frac{p_a}{\rho_w g} \tag{2}$$

Where the indicia (w) denotes the water phase and p_a denotes the atmospheric pressure. The gas density can be expressed explicitly as a function of the vertical coordinate z:

$$\frac{\rho_g(z)}{\rho_g(0)} = \left[\frac{H - z}{H}\right]^{\frac{1}{n}} \tag{3}$$

Only the similarity form of the governing equations, in which the dependence on r is eliminated through profile assumptions, is considered. Some coefficients in the plume equations are dependent on the profile assumptions used. Based on experimental observations, the spatial variations in r for the vertical liquid velocity and the buoyancy (void fraction) are assumed to have Gaussian shapes.

The Gaussian expression used are:

$$\bar{w}(r,z) = w(z)\exp\left(\frac{-r^2}{b^2}\right) \tag{4}$$

$$\rho_w - \bar{\rho}(r,z) = [\rho_w - \bar{\rho}(z)]exp\left[\frac{-r^2}{(\lambda b)^2}\right] \tag{5}$$

Where λ denotes the ratio between the widths of the buoyancy and momentum profiles.
The integral form of the governing equations for steady state bubble plumes are: (Fanneløp and Sjøen, 1980)

$$\frac{d}{dz}(\varphi_1 w b^2) = 2\alpha w b \tag{6}$$

$$\frac{d}{dz}(\varphi_2 w^2 b^2) = \varphi_3 g \frac{\rho_w - \rho}{\rho_w} b^2 \quad \textbf{(7)}$$

The coefficients φ_i are dependent on the assumed shape of the velocity and buoyancy profiles in the plume. The general expressions and the specific values for Gaussian profiles are given in Table 1.

The mass conservation of the gas phase can be expressed as:

$$\dot{m}_g = \varphi_3 \pi \rho b^2 \frac{\rho_w - \rho}{\rho_w} \left(\frac{w}{\varphi_4} + w_s \right) \quad \textbf{(8)}$$

Which leads, using Eq.3, to the following expression for the void fraction:

$$\frac{\rho_w - \rho}{\rho_w} = \frac{\dot{V}_0}{\varphi_3 \pi b^2 \left(\frac{w}{\varphi_4} + w_s \right)} \left[\frac{H}{H - z} \right]^{\frac{1}{n}} \quad \textbf{(9)}$$

This expression is inserted in Eq.7 to give final form of the momentum equation for the plume:

$$\frac{d}{dz}(\varphi_2 w^2 b^2) = \varphi_4 \frac{g \dot{V}_0}{\pi (w + \varphi_4 w_s)} \left[\frac{H}{H - z} \right]^{\frac{1}{n}} \quad \textbf{(10)}$$

In this equation w_s denotes the slip velocity, defined as the velocity of the bubbles relative to the surrounding water.

Coeff.	Definition	Gauss
φ_1	$\frac{2}{wb^2}\int_0^\infty \bar{w}\, r\, dr$	1
φ_2	$\frac{2}{w^2b^2}\int_0^\infty \bar{w}^2\, r\, dr$	1/2
φ_3	$\frac{2}{(\rho_w-\rho)b^2}\int_0^\infty (\rho_w-\bar{\rho}) r\, dr$	λ^2
φ_4	$\varphi_3 \frac{w(\rho_w-\rho)b^2}{2\int_0^\infty \bar{w}\,(\rho_w-\bar{\rho}) r\, dr}$	$1+\lambda^2$

Table 1 Shape coefficients for the plume equations (Fanneløp and Bettilini 2007)

2.1.1. Empirical Coefficients

I will in the following present the most important empirical coefficient related to a bubble plume, and also present some recommended values for the different parameters. If not mentioned otherwise the information presented is largely taken from Fanneløp and Bettilini (2007). The plume development is sensitive to variations in the entrainment coefficient α. Variations of the remaining parameters in their expected uncertainty ranges have only minor influence on the results. Only results based on the Gaussian assumptions will be presented, which is expected to represent the most realistic approach.

2.1.1.1. Gas Expansion

The gas expansion is assumed to be isothermal (n=1), as an adiabatic process would result in an unrealistically large drop in temperature for the rising gas.

2.1.1.2. Entrainment Coefficient α

The entrainment coefficient α has been found to increase with increasing gas flow rates. This can be accounted for by means of a semi-empirical correlation proposed by Milgram (1983):

$$\alpha = K\frac{Fr}{A+Fr} \qquad \textbf{(11)}$$

With:

$$K = 0.165 \; and \; A = 7.598$$

The bubble Froude number Fr:

$$Fr = c^{1/3} q^{2/5} g^{3/10} \left(\rho_w - \rho_g\right)^{1/2} T^{-1/2} \quad \textbf{(12)}$$

Where q is given as:

$$q = c\pi\lambda^2 b^2 \left(\frac{w}{1+\lambda^2} + w_s\right) \quad \textbf{(13)}$$

The lower range of the values recommended in Table 2 is found in laboratory experiments, using relatively small gas quantities. The larger values correspond to field experiments. This implies that the value can be different for large and small plumes.

2.1.1.3. Width Ratio λ

The range of variation in λ is smaller and the effect on the plume development is much less important. The lower values correspond also here to laboratory experiments, whereas for very large scales λ is expected to approach 1.

2.1.1.4. Bubble Slip Velocity

For w_s, values of 28-30 cm/s for bubbles of 0.2-1.5 cm diameter are quoted by Levich (1962). For larger diameters the values rises to 35-40 cm/s, but in the turbulent plume the bubbles tend to break up into smaller sizes. A value of 0.3 m/s is recommended. The influence of this parameter is known to be week, therefore this is considered a decent value without any further investigation.

2.1.1.5. Momentum Amplification Factor γ

The momentum amplification factor is defined as the ratio of the total momentum flux due to the momentum flux carried by the mean flow and is a measure for the momentum flux due to turbulent

fluctuations. As the bubbles become very small in comparison with the plume dimensions, bubble dynamics and interactions become less important and the flow behaves like a single-phase fluid. This indicates that γ approach unity. This factor is therefore disregarded in the governing plume equations presented above.

2.1.1.6. Recommended Values

These empirical values presented in Table 2 are considered recommended.

Parameter	Range	Recommended Value
$\boldsymbol{n}$	$1 - \frac{c_p}{c_v} - \cdot$	1
$\boldsymbol{\alpha}$	0.06-0.15	0.1
$\boldsymbol{\lambda}$	0.6-1.0	0.8
$\boldsymbol{w_s}$	0.1-0.4 m/s	0.3 m/s
$\boldsymbol{\gamma}$	1-2	1

Table 2 Recommended empirical values (Fanneløp and Bettilini, 2007)

2.2. Practical Application

To better understand how numerical solutions can be used to predict the flow of gas in water, it is of interest to look closer into todays practice. The Petroleum Safety Authority in Norway performed in 2006 a project aimed at release of gas subsea and the corresponding risks (Tveit, 2006). They were interested in the whole scenario, from leak in a pressurized pipeline at the sea bottom, the flow of gas through the water masses, and the spreading of the gas at the surface. As part of this study they challenged a number of consultancy firms with a case study of different releases between 50 and 300 meters, both steady state and transient. It was of interest to see the different approaches used for the scenario and if there were large discrepancies in the results. The project concluded that there were large discrepancies in the result and that there is a huge potential for improvement.

All the firms had different approaches to the problems. The gas-liquid flow subsea were for all firms based on the work done by Fanneløp and Sjøen (1980), with some interpretation and adjustment. The basis for the relation used is given by Sintef and Scandpower (2003) as:

$$D = 2 * tan\left(\frac{12.6}{2}\right) * z \quad \textbf{(14)}$$

Where D is the diameter of the plume and z is the water depth. It is observed from experiments that the angle tip of the plume is approximately 18 degrees, but most of the gas will be contained in the central part with tip angle 0.7x18 = 12.6 degrees. As mentioned in Chapter 1, this relation are based on a number of assumptions that might not be valid for larger depths and large transient releases. To minimize the effect of these limitations some adjustments are made to the standard bubble plume relation in the mentioned case study. It is expected that you get large radial distribution of gas at the surface with large flow rates. This is e.g. accounted for by adding a factor to the plume diameter dependent on the flow rate and water depth. Also other small adjustments are made. The large discrepancies in the results, and the inherent limitations in the assumption that this relation is based on, indicates a need for better models for the subsea phase of a gas release.

2.2.1. Integral Models

Numerous integral models, or plume models, have been developed in later years. In most plume models, the equation for continuity of mass, momentum and scalar properties (e.g. temperature and salinity) are defined for control volumes bounded by cross-sections normal to the trajectory (Yapa and Zeng, 1997). Examples of such models are DeppBlow – a Lagrangian Plume Model for Deep Water Blowouts developed by Sintef (Johansen, 2000), and StarPlume developed by Bettelini and Fanneløp (1993). These models are off course much simpler than numerical simulations, and therefore more attractive if they can produce good results. Verification and comparison against experiments have been done for both DeepBlow (Johansen, 2003) and StarPlume (Fanneløp and Bettilini, 2007) with promising results. The DeepBlow model aims at tackling the deep water challenges and have included effects of cross-currents, non-ideal gas behavior, dissolution of gas, and formation and subsequent disintegration of hydrate. These models can be used to develop databases for quick lookup of standard cases. Numerical models versus integral models is a central part of the discussion presented in the conclusion of this thesis.

2.3. Computational Fluid Dynamics - CFD

This section is drawn from Veersteg and Malalasekera (2007) and Hjertager (2001). CFD is explained as being the analysis of systems involving fluid flow, heat transfer and associated phenomena by means of computer-based simulation. Due to its complexity CFD requires high performance hardware, and has therefore had its resurgence in the last two decades. Numerical algorithms are used to solve the problem, and normally different codes contain three main elements:

1. Pre-processor – consists of the input and involves activities as defining the geometry (domain), grid generation (a grid of cells), selection of the phenomena to be modelled, defining the fluid properties and specification of boundary conditions.
2. Solver – numerical solution technique that includes integration of the governing equations, discretization (converting integral equations into system of algebraic equations) and solving of algebraic equations.
3. Post-processor – graphics, e.g. vector plots, contour plots, surface plots and particle tracking and animation for dynamic result display.

All fluid flows are governed by three fundamental principles:

- Conservation of mass
- Conservation of energy (1st law of thermodynamics)
- Newton's second law (force equal the rate of change of momentum)

By the use of these principles one can derive the Navier-Stokes equations which is a set of partial differential equations that describes the flow of fluid. Determine numerical simulations to the Navier-Stokes equation that can be propagated in time and space are the main issue of CFD. The following equations should be solved on a grid inside the region of interest. The dependent variables in a 3-dimensional numerical analysis are:

- p – pressure
- T – temperature
- μ – viscosity
- γ – heat conductivity
- ρ – density
- u – velocity in x-direction
- v – velocity in y-direction
- w – velocity in z-direction

In total there are 8 unknown variables, and therefore 8 equations are needed for solving CFD problems.

Eq. 1 – Conservation of mass (continuity equation)

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \vec{u}) = 0 \tag{15}$$

Eq. 2, 3 and 4 – Momentum balance in x, y and z direction (on vector form)

$$\rho \frac{D\vec{V}}{Dt} = -\vec{\nabla}P + \vec{\nabla}\vec{\tau} + \vec{F} \tag{16}$$

$\vec{V}$ is the velocity vector, $\vec{\tau}$ is the viscous stress vector and $\vec{F}$ is the body force.

Eq. 5 – Conservation of energy (on tensor form)

$$\frac{\partial(\rho h)}{\partial t} + \frac{\partial(\rho u_j h)}{\partial x_j} = -\frac{\partial q_j}{\partial x_j} + \frac{Dp}{Dt} + \tau_{i,j}\frac{\partial u_i}{\partial x_j} + \dot{S} \tag{17}$$

Where $\dot{S}$ is the source term and h is enthalpy.

Eq. 6 – Equation of state (general transport equation)

$$\frac{\partial(\rho\varphi)}{\partial t} + \frac{\partial(\rho u_i \varphi)}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\Gamma_\varphi \frac{\partial \varphi}{\partial x_i}\right] + S_\varphi \qquad \textbf{(18)}$$

Where φ is a general variable and Γ is the transport coefficient.

Eq. 7 and 8 – Empirical relations for μ and k

Viscosity is a measure of the internal resistance between neighboring particles. In a Newtonian fluid the viscous stresses and the rate of deformation is proportional. All variables are split into a mean and a fluctuating part, this is called Reynolds decomposition. The time average of the fluctuating part is introduced into the general transport equation.

2.3.1. Mathematical Modeling of a Bubble Plume

This section is largely drawn from Smith and Dhotre (2007). The equations of continuity and motion for phase φ in an Euler-Euler description of a two-phase flow can be represented in the following generalized form (Drew, 1971):

$$\frac{\partial}{\partial t}\left(\rho_\varphi \alpha_\varphi\right) + \nabla\left(\alpha_\varphi \rho_\varphi u_\varphi\right) = 0 \qquad \textbf{(19)}$$

$$\frac{\partial}{\partial t}\left(\alpha_\varphi \rho_\varphi u_\varphi\right) + \nabla\left(\alpha_\varphi \rho_\varphi u_\varphi u_\varphi\right) = -\nabla\left(\alpha_\varphi \tau_\varphi\right) - \alpha_\varphi \nabla \mathrm{p} + \alpha_\varphi \rho_\varphi g + M_{I,\varphi}. \qquad \textbf{(20)}$$

The terms on the right-hand side of Eq.20 represent, respectively, the viscous stress, the pressure gradient, gravity and the momentum exchange between the phases due to interface forces. The momentum exchange term $M_{I,L}$ describing the interface forces is given as follows:

$$M_{I,L} = -M_{I,G} = M_{D,L} + M_{L,L} + M_{VM,L} + M_{TD,L} \quad \textbf{(21)}$$

The terms on the right-hand side are, respectively, forces due to interphase drag, lift, virtual mass and turbulent dispersion. All other components of the interfacial momentum transfer term except the drag force are referred to as the non-drag forces.

2.3.1.1. Drag Force

The origin of the drag force is due to the resistance experienced by a body moving in a fluid continuum. For body flows the drag force density is usually expressed in the following form:

$$M_{D,L} = \frac{3}{4}\alpha_G \rho_L \frac{C_D}{d_b}|u_G - u_L|(u_G - u_L) \quad \textbf{(22)}$$

Where C_D is the drag coefficient and d_b is the mean bubble diameter. For a bubble with a given shape in steady motion in a Newtonian incompressible fluid, C_D depends only on the bubble Reynolds number, and is given by an experimental drag curve. In the Newton regime ($Re_B \geq 750$), form drag dominates, and $C_D$is practically independent of Re_B and take the value of $C_D = 0.44$ for solid spherical particle. This drag coefficient is used for bubbles in computational studies of two-phase flows and plume, e.g. Pfleger and Becker (2001). A number of models for drag force are developed. For bubbles in the viscous scheme, Ishii and Zuber (1979) propose the following correlation:

$$C_D = \frac{24}{Re_B}\left(1 + 0.1{Re_B}^{0.75}\right) \quad \textbf{(23)}$$

In the Newton regime, bubbles become distorted, and for the distorted particle regime Ishii and Zuber (1979) propose the following correlation:

$$C_D = \frac{2}{3}Eo^{\frac{1}{2}} \quad \textbf{(24)}$$

Where $Eo = g(\rho_L - \rho_G)\,{d_B}^2/\sigma$. Visual inspection of bubbly flows indicates that the bubble shape is not spherical but somewhat distorted (Simiano, 2005). The bubbles seems to be dragged out in

horizontal direction and moves with a wobbling motion. Ishii and Mishima (1984) propose the following correlation for wobbling bubbles:

$$C_D = \frac{\sqrt{2}}{3} N_\mu Re_B \quad \textbf{(25)}$$

2.3.1.2. Lift Force

A bubble traveling through a fluid in shearing motion will experience a lift force transverse to the direction of motion. This force may be expressed as (Drew and Lahey, 1987):

$$M_{L,L} = \alpha_G \rho_L C_L (u_G - u_L) \times (\nabla \times u_L) \quad \textbf{(26)}$$

It is observed from experiments that the rise velocities of the bubbles in the plume have a Gaussian radial distribution. The action of the lift force on the bubbles will therefore be away from the axis of the plume, causing the plume to spread laterally. A lift coefficient of $C_L = 0.1$ is suggested by e.g. Sheng and Irons (1995). Deen (2001) suggest a lift coefficient of $C_L = 0.5$.

2.3.1.3. Virtual Mass Force

The virtual mass force accounts for relative acceleration, the additional work performed by the bubbles in accelerating the liquid surrounding the bubble. While doing this, bubbles displace the liquid around them, and effectively experiencing a resistance to acceleration. You can represent the virtual mass by means of an enhanced gas density:

$$\rho'_G = \rho_G + C_{VM} \rho_L \quad \textbf{(27)}$$

For a rigid sphere, the virtual mass force is $C_{VM} = 0.5$.

2.3.1.4. Turbulent Dispersion Force

Drag and lift forces depend on the actual relative velocity between the phases, but the Reynolds-averaged equations of motion for the liquid only provide information regarding the mean flow field. To account for the random influence of the turbulent eddies, the concept of a turbulent dispersion force has been advanced. By analogy with thermal diffusion of air molecules in the atmosphere, this

force is set proportional to the local bubble concentration gradient. Lopez de Bertodano (1992)derived this relation as:

$$M_{TD,L} = -C_{TD}\rho_L k \nabla \alpha_G \tag{28}$$

Where k is the liquid turbulent kinetic energy per unit mass. In the analogy, the role of the thermal energy of the air molecules is played by the turbulent energy of the liquid. This consideration makes the force proportional to the turbulent kinetic energy and the liquid density, with the empirical coefficient set to $C_{TD} = 0.1$ (Anglart et al., 1993)

2.3.1.5 Turbulence Modeling

Launder and Spalding (1972) have developed a model based on the $k - \varepsilon$ formulation, to model the turbulence phenomena in the continuous phase (liquid) of the gas-liquid flow. The governing equations for the turbulent kinetic energy k and turbulent dissipation rate ε are:

$$\frac{\partial}{\partial t}(\alpha_L \rho_L k) + \nabla(\alpha_L \rho_L u_L k) = \nabla \cdot \left(\alpha_L \frac{\mu_{T,L}}{\sigma_L} \nabla k\right) + \alpha_L (G - \rho_L \varepsilon) + S_k \tag{29}$$

$$\frac{\partial(\alpha_L \rho_L \varepsilon)}{\partial t} + \nabla(\alpha_L \rho_L u_L \varepsilon) = \nabla \cdot \left(\alpha_L \frac{\mu_{T,L}}{\sigma_\varepsilon} \nabla \varepsilon\right) + \alpha_L \left(C_{\varepsilon 1} \frac{\varepsilon}{k} G - C_{\varepsilon 2} \rho_L \frac{\varepsilon^2}{k}\right) + S_\varepsilon \tag{30}$$

With standard model constants $C_{\varepsilon 1} = 1.44$, $C_{\varepsilon 2} = 1.92$, $C_\mu = 0.09$, $\sigma_k = 1$, $\sigma_\varepsilon = 1.3$. The term G is the production of turbulent kinetic energy.

$$G = \tau_L : \nabla \mu_L \tag{31}$$

2.4. Experiment (Fanneløp and Sjøen, 1980)

As the offshore oil-drilling activities in Norway increased, so did the awareness of risk related to uncontrolled blowouts. Fanneløp and Sjøen were interested in analyzing the hydrodynamics of underwater blowouts, to better be able to predict the fate of the released oil and gas.

The study included a theoretical study to be able to predict the deep-set bubble plumes expected from underwater blowouts. The results of their study were compared against other relevant data. In addition they also wanted to make an engineering analysis of the surface interaction zone, since this region largely determines the form and thickness of the oil slick resulting from the blowout. To support their analysis a number of experiments were performed. They used a ship model towing basing, with depth of 5.5m and 10m. These depth were not considered sufficient to verify all the predicted effects of bubble expansion, but the data from the interaction zone were expected to be valid also for deep-set blowout plumes. They performed a number of tests, among them 4 cases in the 10m pool with flow rates ranging from 0.005Nm3/s to 0.022Nm3/s. These are the ones that will be used for comparison in this thesis. The width of the tank was 10.5m and the length was 80m in the 10m part of the pool and additional 150m in the section of 5m depth. Since the dimension was large, the influence of the walls were considered not to influence the data. The parameters of interest are the vertical speed, plume radius and gas concentration throughout the trajectory of the bubble plume.

3. Case Setup

In this chapter I will present the basic setup for the cases used for comparison against the Fanneløp experiment (Fanneløp and Sjøen, 1980). This includes information on the mesh, boundary conditions, solution controls and other parameters of interest. Some settings are different for the 2D and 3D cases, e.g. the mesh, and this will be clearly shown. I will also give a short introduction to the CFD code OpenFOAM, followed by a detailed description of the solver used. A bubble plume is a complex phenomenon, and therefore it's not straight forward to set up a good case in OpenFOAM. I have decided to use the `bubbleFoam` solver in OpenFOAM for all the cases. This is based on theoretical study and testing of other solvers, e.g. `twoPhaseEulerFoam`. Of the solvers I tested, `bubbleFoam` seemed to be the most stable solver when introducing turbulence for this particular problem. `bubbleFoam` also has some limitations in that regards, which could lead to occasional divergence. By optimizing the time step control, I was able to get convergence for all the cases. The results of the cases are presented in Chapter 4.

3.1. Introduction to OpenFOAM

According to Hjertager (2009) OpenFOAM is an open source CFD software package written in C++. The development started at the Imperial College in London in the period 1990-1999. The function of OpenFOAM generally occurs via text files and unix style commands. A Linux operating system is required to run OpenFOAM. The case structure of OpenFOAM can generally be presented as in Figure 5. Each case is saved as a directory, and it must contain at least the three directories: 0, constant and system. In the constant directory you will find material properties, turbulence properties and mesh information. Solution controls, discretization schemes and time step controls are located in the system directory. In the 0 directory all the initial flow fields relevant for the solver used are found. Boundary conditions are set for each field. As the case is solved, new time dumps are written in their own directories. These directories show the updated field data for all the relevant fields.

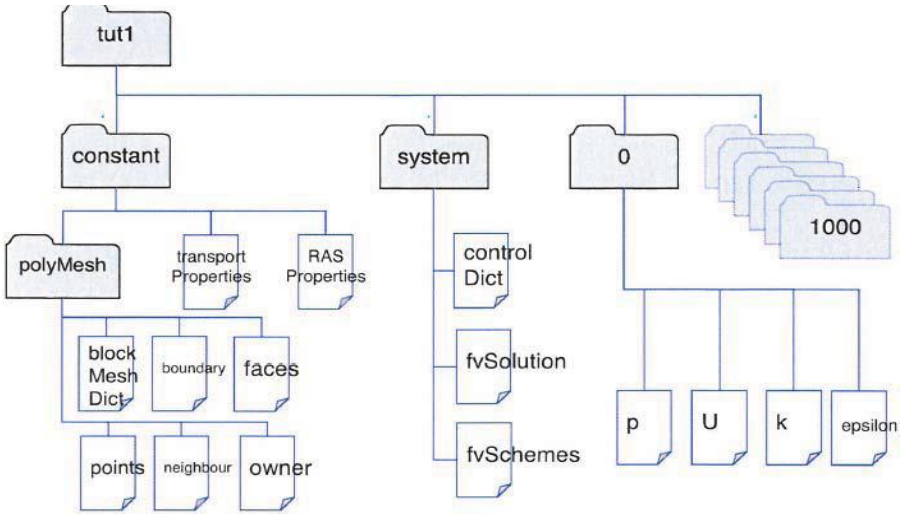

Figure 3 OpenFOAM case structure (Hjertager, 2009)

3.2. bubbleFoam

This section is largely drawn from openfoanwiki.net (2012). The `bubbleFoam` solver is a two-phase solver based on the Euler-Euler two fluid methodology (Drew, 1971, Weller, 2005, Drew, 1983, Enwald et al., 1996, Hill, 1998), suitable to compute dispersed gas-liquid flows. In the Euler-Euler two-fluid approach, the phases are treated as interpenetrating continua, which are capable of exchanging properties, like momentum, energy and mass one with the other. Typical applications of the two-fluid approach, as implemented in `bubbleFoam`, are:

- Bubble columns
- Stirred tank reactors
- Static mixers

```
bubbleColumn
|-- 0
|   |-- Ua
|   |-- Ub
|   |-- alpha
|   |-- epsilon
|   |-- k
|   `-- p
|-- constant
|   |-- RASProperties
|   |-- g
|   |-- polyMesh
|   |   |-- blockMeshDict
|   |   `-- boundary
|   `-- transportProperties
`-- system
    |-- controlDict
    |-- fvSchemes
    `-- fvSolution
```

Figure 4 Case structure of the bubbleFoam solver(openfoanwiki.net, 2012)

3.2.1. Capabilities

The `bubbleFoam` solver implements the two-fluid equations for the simulation of gas-liquid flows (Weller, 2005, Rusche, 2002). The model undergoes the following assumptions:

- Phases are incompressible
- The dispersed phase particle diameter is constant
- The flow is isothermal
- Only momentum exchange is accounted for in the momentum transport equations

The main features of the solver are the following:

- Capability to solve for dispersed two-phase flows with strong density ratio
- Robust solution algorithm, able to deal with complete flow separation
- Turbulence modelling through k-epsilon model and standard wall functions

3.2.2. Limitations

The `bubbleFoam` solver currently has the following limitations:

- Only the dispersed phase and a continuous phase can be described. It is not possible to account for multiple dispersed phases (i.e. represent a dispersed phase diameter distribution)
- The diameter of the particles constituting the dispersed phase is assumed to be constant. Aggregation, breakage and coalescence phenomena are not accounted for
- The drag coefficient is computed as a blend of the drag coefficients evaluated for each phase on the basis of the phase fractions, and no alternative drag models are available
- The interaction between the phases happens only through the momentum exchange term in the corresponding momentum equations:
    - It is not possible to model the heat transfer between phases
    - It is not possible to model the mass transfer between phases
    - No chemical reaction model is available

3.2.3. Governing Equations

In the two-fluid approach, a continuity and a momentum equation are solved for each phase present in the system. These equations can be derived by conditionally averaging the single phase flow equation. See e.g. Drew (1971). The continuity equations for each phase φ has the form:

$$\frac{\partial}{\partial t}\left(\alpha_\varphi \rho_\varphi\right) + \nabla \cdot \left(\alpha_\varphi \rho_\varphi U_\varphi\right) = 0 \qquad \textbf{(32)}$$

Where α_φ is the phase fraction of phase φ. ρ_φ is the density of the material constituting the same phase, and U_φ is the phase velocity. The phase momentum equation is:

$$\frac{\partial}{\partial t}\left(\alpha_\varphi \rho_\varphi U_\varphi\right) + \nabla \cdot \left(\alpha_\varphi \rho_\varphi U_\varphi U_\varphi\right) + \nabla \alpha_\varphi \tau_\varphi + \nabla\left(\alpha_\varphi R_\varphi\right) = \alpha_\varphi \nabla p + \alpha_\varphi \rho_\varphi g + M_\varphi \qquad \textbf{(33)}$$

Where τ_φ is the phase laminar stress tensor, assumed to be Newtonian, R_φ is the phase Reynolds stress tensor, and M_φ is the momentum exchange term. The laminar stress tensor is defined, for each phase:

$$\tau_{\varphi} = -\rho_{\varphi} v_{\varphi} \left[\nabla U_{\varphi} + \nabla^T U_{\varphi}\right] + \frac{2}{3}\rho_{\varphi} v_{\varphi} \left(\nabla \cdot U_{\varphi}\right) I \quad \textbf{(34)}$$

Where v_{φ} is the molecular kinematic viscosity of the fluid constituting phase φ, and I is the identity matrix. The phase Reynolds stress tensor is given by:

$$R_{\varphi} = -\rho_{\varphi} v_{\varphi,t} \left[\nabla U_{\varphi} + \nabla^T U_{\varphi}\right] + \frac{2}{3}\rho_{\varphi} v_{\varphi,t} \left(\nabla \cdot U_{\varphi}\right) I + \frac{2}{3}\rho_{\varphi} k_{\varphi} I \quad \textbf{(35)}$$

$Where\ k_{\varphi}\ is\ the\ phase\ turbulent\ kinetic\ energy. In\ bubbleFoam, the\ turbulent\ kinetic\ energy\ is$ assumed identical for both of the phases present in the system. $v_{\varphi,t}$ is the phase turbulent kinematic viscosity, defined as:

$$v_{\varphi,t} = C_{\mu} \frac{k_{\varphi}^2}{\varepsilon_{\varphi}} \quad \textbf{(36)}$$

C_{μ} is a constant, and ε_{φ} is the phase turbulent dissipation rate. The phase effective viscosity is calculated as the sum of the phase molecular viscosity and of the phase turbulent viscosity.

$$v_{\varphi,eff} = v_{\varphi} + v_{\varphi,t} \quad \textbf{(37)}$$

The momentum exchange term can be decomposed as a drag contribution, a lift force contribution and a virtual mass contribution:

$$M_{\varphi} = M_{\varphi,drag} + M_{\varphi,lift} + M_{\varphi,vm} \quad \textbf{(38)}$$

The terms are modelled according to Weller (2005). The drag term is described as:

$$M_{a,drag} = \frac{3}{4}\alpha_G \alpha_L \left(\alpha_L \frac{C_{D,G}\rho_L}{d_G} + \alpha_G \frac{C_{D,L}\rho_G}{d_L}\right)|U_r|U_r \quad \textbf{(39)}$$

Where d_a and d_b are the phase particle diameter, $U_r = U_a - U_b$ is the relative velocity vector, and $C_{D,a}$ and $C_{D,b}$ are the drag coefficients computed with respect to each phase, according to:

$$C_{D,\varphi} = \frac{24}{Re_\varphi}\left(1 + 0.15Re_\varphi^{0.687}\right) \quad \textbf{(40)}$$

The model for drag is hardcoded in the `bubbleFoam` solver, and no constants can be selected by the user.

The lift term is modelled as:

$$M_{\varphi,lift} = \alpha_G \alpha_L \left(\alpha_L C_{a,lift}\rho_L + \alpha_G C_{L,lift}\rho_G\right) U_r \times \nabla \times U \quad \textbf{(41)}$$

The user have to set the lift coefficient, C_l.

The virtual mass force term is evaluated as:

$$M_{\varphi,vm} = \alpha_G \alpha_L C_{vm}\rho_L \left(\left.\frac{dU_L}{dt}\right|_L - \left.\frac{dU_G}{dt}\right|_G\right) \quad \textbf{(42)}$$

The user have to set the virtual mass force, C_{vm}.

3.2.3.1. Turbulence Model

The `bubbleFoam` solver uses a two-equation $k - \varepsilon$ turbulence model for the continuous phase, and accounts for the influence of the turbulence on the dispersed phase by scaling the dispersed phase turbulent viscosity. The equation for the turbulent kinetic energy of the continuous phase is:

$$\frac{\partial}{\partial t}(\alpha_L k_L) + \nabla(\alpha_L U_L k_L) - \nabla\left(\frac{\nu_{L,eff}}{\sigma_k}\nabla k_L\right) = \alpha_L G - \alpha_L \varepsilon_L \quad \textbf{(43)}$$

Where σ_k is the turbulent Schmidt number and G is the production of the turbulent kinetic energy, given by:

$$G = 2v_{b,t}[\nabla U_L \cdot dev(\nabla U_L + \nabla^T U_L)] \quad \textbf{(44)}$$

The turbulent dissipation rate is determined by solving the transport equation:

$$\frac{\partial}{\partial t}(\alpha_G \varepsilon_L) + \nabla \cdot (\alpha_L U \varepsilon_L) - \nabla\left(\frac{v_{b,eff}}{\sigma_k}\nabla\varepsilon_L\right) = C_1 \alpha_L G \frac{\varepsilon_L}{k_L} - C_2 \alpha_L \frac{\varepsilon_L^2}{k_L} \quad \textbf{(45)}$$

Where C_1 and C_2 are constants of the turbulence model. The turbulence viscosity of the continuous phase is calculated from its definition:

$$v_{L,t} = C_\mu \frac{k_L^2}{\varepsilon_L} \quad \textbf{(46)}$$

The turbulent viscosity of the dispersed phase is evaluated as:

$$v_{G,t} = C_t^2 v_{L,t} \quad \textbf{(47)}$$

The coefficient C_μ and the turbulent response coefficient, C_t, are constants of the model. For the turbulence model, C_t is the only coefficient that has to be specified by the user. The other constants in the model are presented in Table 3:

C_μ	0.09
C_1	1.44
C_2	1.92
alphak	1
alphaEps	0.76923

Table 3 Constants for the turbulence model

Standard wall functions for the turbulence model are adopted to treat the zone next to the wall. These functions are hard coded in the `bubbleFoam` solver, and is not to be specified by the user. The constants for the wall functions are kappa = 0.41, and E = 9.8.

3.2.4. Solution Procedure

The numerical solution of the two-phase equation relies on a segregated algorithm based on the PISO procedure extended to two-phase flows (Oliveira and Issa, 2003). The momentum equations are manipulated to stabilize the system of equation at the limits of the range of volume fractions, to avoid singularities. For details on the implementation and numerical methodology of `bubbleFoam` the reader is referred to openfoanwiki.net (2012), which is based on Oliveira and Issa (2003), Rusche (2002) and Weller (2005). The solution procedure adopted in the `bubbleFoam` solver can be summed up as follows:

1. Solve the phase continuity equations
2. Update lift, drag and virtual mass coefficients
3. Construct the momentum equation matrix
4. Predict the phase velocity fields, without considering the pressure gradient at this stage
5. Solve the pressure equation
6. Correct the velocities with the new pressure field, and update the phase fractions
7. Solve the transport equations for the turbulence quantities

3.3. Case Description

As mentioned in Chapter 1, I will study how OpenFOAM can be used to simulate a gas-liquid flow from released gas. To do this I have set up a simple domain where air is injected at the bottom (dispersed phase) into the water phase (continuous phase). The air phase above the water is also included. This will make it possible to study how the interaction between the water and air phase will affect the behavior of the bubble plume. It could be of interest to see how OpenFOAM performs both in 2D and 3D. A case run in 2D is much cheaper computational wise, and therefore it is easier to perform a parameter study. The 2D case will be used for direct comparison against experiments, but it will also serve as basis for the 3D case.

3.3.1. Mesh Generation

The mesh is generated by modifying the `blockMeshDict` and running the `blockMesh` function in OpenFOAM. `blockMeshDict` is found under `constant` and `polyMesh`. `blockMesh` is a non-graphical multiblock mesh generator. Inputs are point coordinates and listings of point defining the limits of blocks and boundaries (Hjertager, 2009).

The 2D mesh is 20m wide and 12m high. It is build out of 3 blocks, where the bottom of the one in the middle represent the inlet. The mesh is separated into 100 cells in y direction and 162 cells in x direction. On average that is equal to about 8.33 cells/m in y direction and 8.1 cells/m in x direction. The complete `blockMeshDict` for the 2D case can be viewed in Appendix B.5.

The 3D mesh is 5m wide in both x and y direction, and has a height of 12m. The decrease in width from the 2D case is purely because of computational cost. It is assumed that the influence of the walls will not be significantly higher compared to a width of 10m. The mesh is built out of 9 blocks, whereas the bottom of the one in the middle act as an inlet. The mesh is separated into 100 cells in z-direction and 42 cells in x and y direction. This implies that the grid coarseness of the 2D and 3D case are similar. The complete `blockMeshDict` for the 3D case can be viewed in Appendix B.4.

According to Hjertager (2009) the mesh quality is crucial to the success and accuracy of the results. The command `checkMesh` crates a list of specific mesh quality measures. These measures are listed in Table 4. The max cell openness should be nearly zero, and the value of 1.4456e-16 and 1.156e-16 fulfills that requirement. The minimum and maximum face area and the minimum and maximum volume should be positive, and that is the case for both the 2D and 3D mesh. The mesh is based on exclusively tetrahedral cells, which indicates that the non-orthogonality and maximum skewness of the mesh is expected to be very low to zero. This is confirmed by a mesh non-orthogonality of 0 for both the meshes, and a very low maximum skewness. These values are well within the recommended limits. When running `chechMesh` both of the meshes are given “mesh OK” at the end, which according to Hjertager (2009) indicates that there are no major problems with the mesh.

	2D Mesh	3D Mesh
Points	32926	186749
Faces	65062	539364
Internal faces	32138	519036
Tetrahedra cells	16200	176400
Boundary patches	4	3
Faces inlet	2	4
Faces outlet	162	1764
Faces fixedWalls	360	18560
Max cell openness	1.162e-16	1.156e-16
Maz aspect ratio	2.4	2.45
Minimum face area	0.005	0.0025
Maximum face area	0.0149	0.015
Min volume	0.0006	0.0003
Max volume	0.00149	0.0018
Total volume	24	300
Mesh non-orthogonality	0	0
Max skewness	4.2e-12	8.34165e-12

Table 4 Mesh data taken from `checkMesh`

3.3.2. Boundary Conditions

In this sub-chapter I will specify the boundary conditions for both the 2D and 3D case. If there is a difference in the setup of the 2D and 3D case this will be highlighted. E.g. the velocity fields will be different, since upward will be the y-axis and z-axis respectively.

The boundaries used are:

- `Inlet` – defines the patch where air is entering the water at the bottom
- `Outlet` – defines the parch covering the top of the domain where air is allowed to leave the domain
- `fixedWalls` – defines the walls on the sides of the domain, and also the bottom surrounding the inlet

The boundary conditions can be viewed in full in Appendix A. The appendix is based on Case 1 from the Fanneløp and Sjøen (1980) experiment, indicating that the flow rate used is 0.005N$\boldsymbol{m^3}$/s.

3.3.2.1. alpha

The dictionary `alpha`, found under `0`, defines the dispersed phase, in this case injected air. The `inlet` is defined by a `fixedValue` of `uniform 1`. This means that 100% air is injected. The `outlet` and `fixedWalls` are defined as `zeroGradient`. OpenFOAM (2011) describes a patch using `zeroGradient` as "normal gradient of ϕ is zero", where ϕ is the patch field. This implies that it will balance the patch. If it allows air to flow out it will replace it to make the normal gradient equal to 0. In reality this will only apply for the `outlet`, since the `fixedWalls` are defined as wall and not patch.

Since both water and air phase are present in the domain the `internalField` is defined by a `nonuniform List<scalar>`. This is a list of all the cells given a value of either 0 (water) or 1(air). To set up the scalar field I have used the function `setFieldsDict` (See Appendix C.4) under `system`. This allows you to define a region of the domain that is e.g. defined as 0(water). To run this application you type the command `setFields` in the terminal and it will automatically update the `internalField` in `alpha`.

3.3.2.2. k and Epsilon

The dictionaries for `k` and `epsilon` defines the starting values for these properties at the different boundaries. The standard k-epsilon turbulence model is applied.

In both dictionaries the `internalField` are given the value `uniform 1e-8`, and the `inlet` are given a `fixedValue` of `uniform 1e-8`. These values are based on the assumption that there will be negligible turbulence before the air is injected, and negligible turbulence in the injected air. The values for `k` and `epsilon` in the domain will be calculated for each time step, and therefore change with time. It should be noted that the `k` and `epsilon` values are only calculated for the continuous phase (water).

The `outlet` and `fixedWalls` are defined as `zeroGradient`. According to openfoanwiki.net (2012) wall functions are adopted for the zone near the walls. This implies that `wallFunction` are hardcoded for `k` and `epsilon` in the `bubbleFoam` solver.

3.3.2.3. Pressure

The `internalField` are defined as `uniform 0`. The `inlet` and `fixedWalls` are defined `zeroGradient`. The `outlet` is given a `fixedValue` of `uniform 0`. This should allow the pressure to be controlled by the gravitational acceleration, meaning higher pressure when the water is getting deeper.

3.3.2.4. Velocity Fields

The velocity field of air is defined in the `Ua` file in the `0` directory. The `internalField` is set to `uniform (0 0 0)` since there will be no movement of air initially. The `outlet` is defined as `zeroGradient`, as you want air to be able to flow out of the domain. The `fixedWalls` are defined as a `fixedValue` of `uniform (0 0 0)`. To account for the flow rate of air you must define a velocity at the `inlet`. This is defined as a `fixedValue` and a velocity in the upwards direction. For the 2D case that would be the y direction, and z direction in the 3D case. The velocity is defined according to the flow rate of the different cases.

The velocity field of water is defined in the `Ub` file in the `0` directory. The `internalField` is set to `uniform (0 0 0)` since there will be no movement of water initially. The `inlet`, `outlet` and `fixedWalls` are defined as a `fixedValue` of `uniform (0 0 0)`, as no water will leave the domain at any point.

3.3.3. Transport Properties

The transport properties, meaning the properties of the materials/fluids, are defined in the `transportProperties` file in the `constant` directory. The full file can be viewed in Appendix B.1. When using the `bubbleFoam` solver you can specify two phases, phasea and phaseb. I have used phasea as air and phaseb as water. The two phases are defined by their kinematic viscosity (`nu`), density (`rho`) and bubble size (`d`). The values used are:

nua	1.6e-05
rhoa	1
da	3e-3
nub	1e-06
rhob	1000
db	1e-4

Table 5 Fluid properties of air and water

These values are taken from the `bubbleColumn` tutorial found in OpenFOAM 2.1.x. The bubble diameters are assumed constant for the solver used. For the injected air, a value of 3mm is chosen. This is based on the results of a parameter study (see section 3.4), but it could be argued that this value should be higher. E.g. Zhang et al. (2006) used a value of 4mm, while Tabib et al. (2008) used a value of 5mm.

For the `bubbleFoam` solver you also have to specify a number of coefficients in `transportProperties`. You need to define the virtual mass coefficient, `Cvm`. This value is set to 0.5. The lift coefficient, `Cl`, is set to 0.5 as proposed by (Deen, 2001). The turbulent response coefficient, `Ct`, is taken from the `bubbleColumn` tutorial found in OpenFOAM 2.1.x, and is given the value 1.

3.3.4. Turbulence Modelling

The file `RASProperties` allows you to specify the setting for the turbulence model, and is found under the dictionary `constant`. The `RASModel` is set to `kEpsilon` and `turbulence` is specified to be `on`. The starting values for `k` and `epsilon` are specified in section 3.2.2. The complete file can be viewed in Appendix B.2.

3.3.5. Environmental Properties

The only environmental property to be specified is the gravitational acceleration. This value is specified in the file `g` found under `constant`. You need to specify a value for the gravitational acceleration and the direction of it. Gravity is set to 9.81m/s2 in negative y and z direction for the 2D and 3D case respectively. The file `g` can be viewed in Appendix B.3.

3.3.6. Discretization Schemes

If not stated otherwise, this information on the different discretization schemes is taken from OpenFOAM (2011). The discretization schemes is found in the file `fvSchemes` in the dictionary `system`. The complete dictionary `fvSchemes` can be found in Appendix C.2. According to openfoanwiki.net (2012) this dictionary contains the definitions of the numerical schemes used in the simulations. Typical terms that must be assigned a numerical scheme in `fvSchemes` range from derivatives, e.g. gradient ∇, and interpolation of values from one set of points to another. The dictionary `fvSchemes` is in full taken from the `bubbleColumn` tutorial in OF 21x.

The `ddtSchemes` represent the choice of time scheme. `Euler` is adopted, and indicates a first order bounded implicit scheme. `gradSchemes` contains gradient terms. It is set to `default Gauss linear`. If a `default` scheme is specified, the settings apply to all the terms to which the sub-dictionary refers. `Gauss` represent the discretization, and indicates that values are interpolated from cell centers to face centers. The interpolation scheme is set to `linear`. The convection schemes are defined under `divSchemes`. The numerical schemes `div((-nuEffa*T(grad(Ua)))` and `div((-nuEffb*T(grad(Ub)))` are set to `Gauss linear`. All other numerical schemes under `divSchemes` are set to `Gauss upwind`. The laplacian scheme is defined by the sub-dictionary `laplacianSchemes`, and is applied to terms of the laplacian operator ∇^2. All the schemes are set to `Gauss linear corrected`. `Gauss` is the only scheme selectable, but it must be followed by an interpolation scheme and a surface normal gradient scheme. The interpolation scheme is set to `linear` and the gradient scheme is set to `corrected`. A `corrected` surface normal gradient scheme indicates a numerical behavior which is unbounded, conservative and of second order. The `interpolationSchemes` is set to `default linear`. `linear` is a centered scheme which indicates linear interpolation. The surface normal gradient schemes is defined by `snGradSchemes`, and is set to `default corrected`, which indicates it is provided with explicit non-orthogonal correction. The last field `fluxRequired` is set to `default no`.

3.3.7. Solution Controls

If not stated otherwise, this information on the different solution controls is taken from OpenFOAM (2011). The equations solvers, tolerances and algorithms are controlled from the `fvSolution` dictionary in the `system` directory. The complete dictionary `fvSolution` can be viewed in Appendix C.3. The dictionary `fvSolution` is in fully taken from the `bubbleColumn` tutorial in OF 21x.

The pressure velocity coupling scheme is set to `PIMPLE`. According to Hjertager (2009) there are two loops in PIMPLE, one inner and one outer. In the outer loop all equations are solved, while in the inner loop only the continuity equation is solved. It is important to ensure that the continuity error of the forgoing loop stays small. The max and mean Courant number should not grow large, but they may have higher values than that satisfied in the PISO pressure velocity coupling. Initial residuals should be kept small for the last outer loop. These remarks are important to ensure a reasonable converged solution at all times. For the PIMPLE pressure velocity coupling scheme you have to specify a number of settings. `nCorrectors` are set to `2`. `nCorrectors` indicates how many times the pressure equation is solved in the outer loop, and for strict time accuracy it is recommended to use minimum 2. The mesh non-orthoganality is 0, and therefore the `nNonOrthogonalCorrectors is set to 0`. A loop over the volume fraction is indicated by the keyword `nAlphaCorr`. It is recommended to use a value of 1-2 for transient flows and 0 for steady state flows. `nAlphaCorr` is set to `2`. Further `pRefCell` and `pRefValue` are set to `0`.

In the sub-dictionary `solvers` the method of number crunching for each discretized equation solved by `bubbleFoam` is specified. Both `alpha` and `k|epsilon` are using the `PBiCG` solver and the `preconditioner DILU`. `PBiCG` is short for preconditioned bi-conjugate gradient, which yields asymmetric matrices. `DILU` is short for Diagonal incomplete-LU. The tolerance for `k|epsilon` and `k|epsilonFinal` is set to `1e-5`, and the relative tolerance is set to `0.1` and `0` respectively. The tolerance for `alpha` and `alphaFinal` is set to `1e-10`, and the relative tolerance is set to `0.1` and `0` respectively. `tolerance` indicates that residuals lower than the given value will stop the solver. `relTol` indicates that a ratio of current to initial residual lower than the given value will stop the solver. The solver for the pressure, `p`, is set to `GAMG`, which is short for generalized geometric-algebraic multi-grid. The solver uses a `DIC` smoother, diagonal incomplete-Cholesky. The tolerance for `p` and `pFinal` is set to `1e-8`, and the relative tolerance is set to `0.1` and `0` respectively.

3.3.8. Time Step and Data Output

Details regarding time step and data output are set in the `controlDict` dictionary under `system`. The `controlDict` directory can be viewed in full in Appendix C.1. The velocities for the different cases will vary, and therefore it is of interest to have an adjustable time step. This way it is possible to have one common `controlDict` for all the cases. The initial `deltaT` is set to `0.002` and `adjustTimeStep` is set to `yes`. The time step is affected by a number of factors, and is controlled by the Courant number. In openFOAM the Courant number is defined as:

$$Co = \frac{\delta t|U|}{\delta X}$$

Where δt defines the time step, U the velocity and δX the distance through the cell in the velocity direction. The Courant number is specified by `maxCo`, and is set to `0.15`. This is a rather low value that is deemed necessary to account for the unstable turbulence model. In some of the 3D cases the value may need to be set even lower for parts of the simulation. The automatic time step control causes calculations to be performed at arbitrary time steps. You can however write results at fixed times. This is defined by the `writeInterval`, and is set to `1`. This implies that every second data will be written to the case directory. The runtime of the simulations are 50 seconds and is defined by `startTime` and `endTime`, which are accordingly set to `0` and `50` respectively. 50s runtime will allow the air particles to flow through the domain 3-5 times, which is recognized as minimum to have a stable solution. However, it could be of interest to consider a larger runtime, but this has proved to be difficult for both the 2D and 3D case. In the 2D case the velocity of water flowing laterally out from the surface region and down the walls will ultimately influence the inlet of air. It will push the flowing air to one side. In the 3D case the problem is related to the inflow of air at the outlet. The air leaving the domain has to be replaced by new air according to physical laws as well as the `zeroGradient` boundary conditions. Between a runtime of 50 and 100 seconds the inflow will reach extremely high values and complicating further simulation. Both these problems could be limited by using larger domains, but the gain from increasing the width of the domain by a factor of 2 is proved to only delay the problems by 10-20 seconds for the 2D case. For the 3D case a significant increase in width of the domain would mean intolerable computational costs for the computer I am using. When you start the solver, you can define where to start from. This is controlled by `startFrom`, and is set to `latestTime`. This means that it will start from the latest time dump that is available in the case directory. The `writeFormat` is set to `ascii`, which is default in openFoam. `writeCompression` is set to `uncompressed`.

When studying bubble plumes it is of interest to consider average values of velocity and void fraction over time. This is done by adding a `function`, `type fieldAverage` in the `controlDict`. Each of the fields set to be averaged with time, are defined by `mean on`, `prime2Mean off` and `base time`.

3.3.9. Sampling of Data

Data to be sampled are specified under the keyword `function` in the dictionary `controlDict` found under `system`. The `controlDict` can be viewed in Appendix C.1. Generally I want to define lines, at which I can read data for velocities and void fraction. To achieve this I have used the function `type sets`. The `setFormat` is defined as `xmgr`. The `interpolationScheme` is set to `cell`. `cell` indicates that the center values of the cells is interpolated to the line. To control the output of the data, `outputControl` is set to `outputTime`. This indicates that the data are collected in the same time intervals as defined by `writeInterval`.

The fields specified to read data from is defined under `fields`, and `UaMean`, `UbMean` and `alphaMean` are chosen. This means that for the lines specified it is of interest to study the water and gas velocity as well as the void fraction. The lines at which the data will be collected are specified under `sets`. I have defined 4 different lines. One will start from the `inlet` and end at the surface. The three others will cross the flow horizontally at 3 different heights. 3m, 6m and 9m are chosen. With data from these lines I will have sufficient information to study the different velocity profiles, as well as the void fraction. It will also provide information regarding the plume radius, as this is defined by the velocity profile. All the lines are specified by `type midpoint`, and the `axis` the specific

line follows. The direction and length are given by coordinates after the keywords `start` and `end`. The data collected are post-processed in xmgrace, as mentioned in chapter 3.3.1.1.

3.3.10. Running the Code

The code is run by typing `bubbleFoam`. Residuals are logged by using the command `bubbleFoam | tee output`.

3.3.11. Post-Processing

ParaView and Grace are used to post-process the results. Animations are produced in ParaView, while samples and residuals are plotted by the use of Grace. ParaView is an open-source program used for visualization, and is run by typing `paraFoam` in the terminal window. Grace is an open-source application used to plot two dimensional graphs, and is run by typing `xmrgace` in the terminal window.

3.4. Parameter Study

In this sub-chapter I will present the results from a parameter study performed. This study is not directly related to the work presented otherwise, but it is included because it gives a good indication on how the different parameters in OpenFOAM and `bubbleFoam` influences the results. This information could be of interest for the reader. The study is based on Case 1 of the experiments (Fanneløp and Sjøen, 1980). The values of the different parameters presented above for further use, are partly based on this study and partly based on other available papers and works. It should be noted that the cases run in this chapter are based on another mesh than the ones used for the main analysis presented in Chapter 4. A 2D mesh with 12 meter height and 10 meter width is used. The mesh coarseness is the same as the one used for the main simulations. It is proven that the mesh with 10m width are strongly influenced by the walls. For this reason I have used 20m width on the main simulations presented in chapter 4. Since the influence of the walls is larger for the cases presented here, the diameters reported are believed to be too large. This is not a significant problem since the goal of this study is to look at the general tendencies in a qualitative manner. The 2D mesh is chosen because it is much cheaper computational wise. Also, the instability of the turbulence model makes it challenging to use a 3D mesh. With the mesh as the exception, the case setup is as presented earlier in this chapter.

The parameters chosen to be subject of this study, are the ones that have to be set by the user and are believed to influence the results. This includes the lift, virtual mass and turbulent response coefficients. I have also considered how the size of the assumed constant diameter of air bubbles influence the solution. In addition I have included the effect of applying the slip function on the walls. The influence of the different parameters are presented in the following sub-chapters. Only the parameter specifically studied in each sub-chapter are subject to change. Data for diameter of plume and void fraction are plotted on 9 meter water height. The diameter of the plume is specified according to Fanneløp (1994), which indicates that the plume ends where the velocity has decreased

to half of the maximum velocity along the centerline. The simulation time for all the cases is set to 50 seconds and the flow rate is set to 0.005Nm3/s, as in case 1 of the experiment (Fanneløp and Sjøen, 1980). The inlet velocity is found by assuming a circular inlet of 0.1m. This gives a velocity of 0.322 m/s. By using a setup from an experiment it is easier to qualitatively evaluate the results.

3.4.1. Bubble Size

For the bubbleFoam solver you have to specify a constant bubble diameter of both the water and gas phase. In this section I have studied the effect of changing the bubble size for the air bubbles. Fanneløp and Sjøen (1980) observed a bubble diameter of 1cm as the air left the inlet. In a turbulent plume the bubbles tends to be unstable and break up into smaller sizes (Fanneløp and Bettilini, 2007). For that reason, it is of interest to look at the average bubble diameter. In Figure 5 the results can be observed. Vs is the slip velocity, D is the plume diameter, C is the maximum concentration of air and Ua is the velocity of air. The values used for the bubble size is between 1 and 5mm. The `bubbleColumn` tutorial for the `bubbleFoam` solver uses 3mm, and therefore it is of interest to investigate the values surrounding this value. Zhang et al. (2006) used a value of 4mm, while Tabib et al. (2008) used 5mm. I have not tried values above 5mm since the results for 5mm already indicates to large velocity.

Bubble size	MeanUa	EndTimeUa	MeanVs	EndTimeVs	MeanD	EndTimeD	MeanC	EndTimeC
1mm	1.21m/s	1.82m/s	0.11m/s	0.1m/s	1m	1m	0.013	0.034
2mm	1.67m/s	1.96m/s	0.2m/s	0.21m/s	1.4m	1.4m	0.016	0.026
3mm	1.83m/s	1.94m/s	0.3m/s	0.29m/s	1.5m	1.4m	0.0156	0.022
4mm	1.93m/s	2.05m/s	0.37m/s	0.36m/s	1.55m	1.45m	0.0156	0.020
5mm	2.02m/s	2.27m/s	0.45m/s	0.44m/s	1.6m	1.6m	0.0152	0.0192

Figure 5 Simulations using bubble size from 1-5mm

The effect of the bubble size is observed to be large. The velocity is clearly increasing with larger bubbles. When the velocity of gas is increasing so is the relative velocity between air and water, the slip velocity. The spreading of the plume is increasing with higher velocity, but that is largely believed to be because of larger influence from the walls with increasing velocity. Since the spreading is increasing with larger velocity, the void fraction is logically showing a decrease for the end time value. 3mm bubble size produces the best results for the slip velocity, which is believed to be around 0.3m/s (Fanneløp and Bettilini, 2007).

3.4.2. Lift Coefficient

A bubble traveling through a fluid in shearing motion will experience a lift force transverse to the direction of motion. A lift coefficient of $C_L = 0.1$ is suggested by e.g.Sheng and Irons (1995). Deen (2001) suggest a lift coefficient of $C_L = 0.5$. Therefore these values will be tested. I have also included the "extremes" 0 and 1. Figure 6 shows the results of using the mentioned values.

Cl	MeanUa	EndTimeUa	MeanVs	EndTimeVs	MeanD	EndTimeD	MeanC	EndTimeC
0	1.79m/s	1.91m/s	0.3m/s	0.29m/s	1.4m	1.4m	0.0194	0.024
0.1	1.82m/s	1.89m/s	0.3m/s	0.29m/s	1.4m	1.4m	0.0194	0.024
0.5	1.83m/s	1.94m/s	0.3m/s	0.29m/s	1.5m	1.4m	0.0156	0.022
1	1.756m/s	1.92m/s	0.3m/s	0.3m/s	1.7m	1.4m	0.0138	0.0193

Figure 6 Simulations using lift coefficients 0, 0.1, 0.5 and 1

The effect of the lift coefficient on the plume is limited. It could be noted that it shows some differences on the velocity profile along the centerline. With Cl=0.1 you will reach maximum mean velocity at 6m, while using Cl=1 you will reach the same point at 4.4m. It is also to be noted that the UaMean for Cl=1 is lower than the other cases, which would make it reach this velocity faster. The conclusion is that the lift coefficient will not influence the velocity profile or other parameters in a significant matter. You can see a decrease in maximum gas concentration at the Cl=1 case. This is most likely related to the larger diameter of the plume observed, which is expected to be caused by the instability using a high value for Ct and the influence of the wall.

3.4.3. Turbulent Response Coefficient

The turbulent response coefficient can be seen as the ratio of fluctuations between the dispersed phase and the continuous phase. The turbulent viscosity of the dispersed phase is calculated by taking the turbulent viscosity for the continuous phase and multiplying it with Ct^2 (openfoanwiki.net, 2012). The turbulent response coefficient is close to 1 for high phase fractions and higher than one for gas-liquid. The `bubbleColumn` tutorial for the `bubbleFoam` solver used a Ct value of 1. This implies that the value should not be set to lower than 1 for the phenomena I am studying, but I have decided to use values between 0.5 and 1.5 to observe if it will change the results. In figure 35 the cases run can be observed.

Ct	MeanUa	EndTimeUa	MeanVs	EndTimeVs	MeanD	EndTimeD	MeanC	EndTimeC
0.5	1.82m/s	1.91m/s	0.3m/s	0.29m/s	1.5m	1.35m	0.0156	0.022
0.9	1.83m/s	1.94m/s	0.3m/s	0.3m/s	1.5m	1.4m	0.0156	0.022
1	1.83m/s	1.94m/s	0.3m/s	0.29m/s	1.5m	1.4m	0.0156	0.022
1.1	1.83m/s	1.94m/s	0.3m/s	0.3m/s	1.5m	1.35m	0.0156	0.022
1.5	1.83m/s	1.94m/s	0.3m/s	0.29m/s	1.5m	1.35m	0.0156	0.022

Figure 7 Simulations using Ct values of 0.5, 0.9, 1, 1.1 and 1.5

The effect of the turbulence response coefficient on the parameters studied here is literally none. The vertical velocity profile is not effected in a significant manner.

3.4.4. Virtual Mass

The virtual mass force accounts for relative acceleration, the additional work performed by the bubbles in accelerating the liquid surrounding the bubble. While doing this, bubbles displace the liquid around them, and effectively experiencing a resistance to acceleration. For a rigid sphere, the virtual mass coefficient is 0.5 (Smith and Dhotre, 2007). A value of 0.5 is mostly used. I have here decided to use values from 0.1 to 0.9 to observe the effect on the results.

Cvm	MeanUa	EndTimeUa	MeanVs	EndTimeVs	MeanD	EndTimeD	MeanC	EndTimeC
0.1	1.85m/s	1.9m/s	0.3m/s	0.29m/s	1.55m	1.5m	0.014	0.021
0.4	1.83m/s	1.9m/s	0.3m/s	0.29m/s	1.5m	1.35m	0.0155	0.021
0.5	1.83m/s	1.94m/s	0.3m/s	0.29m/s	1.5m	1.4m	0.0156	0.022
0.6	1.83m/s	1.94m/s	0.29m/s	0.29m/s	1.5m	1.35m	0.0157	0.022
0.9	1.81m/s	1.95m/s	0.3m/s	0.3m/s	1.5m	1.3m	0.0159	0.022

Figure 8 Simulations using Cvm value of 0.1, 0.4, 0.5, 0.6 and 0.9

The virtual mass coefficient seems to have limited effect on the results. There are some small changes in end time diameter and mean gas concentration, but this seems more random with no clear trend. The virtual mass coefficient seem to have no significant effect on the vertical velocity profile.

4. Results and Discussions

In this chapter the results of the main simulations are presented and compared against the results from the experiment of Fanneløp and Sjøen (1980). The cases run is given in Table 6. There are four cases run in both 2D and 3D, and the cases are similar to the above mentioned experiment. First some general observations of the results will be presented, followed by a closer look at the parameters of interest. At the end the results will be summarized and discussed.

Flow rate	2D	3D
0.005N$\boldsymbol{m^3}$/s	c.1.2D	c.1.3D
0.01N$\boldsymbol{m^3}$/s	c.2.2D	c.2.3D
0.015N$\boldsymbol{m^3}$/s	c.3.2D	c.3.3D
0.022N$\boldsymbol{m^3}$/s	c.4.2D	c.4.3D

Table 6 Overview of cases

4.1. General Observations

The general observations presented in this sub-chapter is based on a flow rate of 0.01Nm^3/s, as in Case 2 of the experiment. The specific results, e.g. velocities and void factions, will be presented later in this chapter. Immediately after the start of the air injection a wide plume head (cap) will be created and rise toward the surface. This initial head will be wider than the established steady bubble plume, see Figure 9. This behavior is expected from a sudden release of air/gas in water. Figure 9-12 are all showing the void fraction on a scale from 0-0.1. From Figure 9 it can be observed that the spreading of the plume head is larger for the 2D case. The air is limited to move only in 2 dimensions, and this is expected to affect the spreading of the initial plume head. Figure 9 further indicates that the 3D case have a higher acceleration. The velocity profiles will be presented and discussed in more detail in section 4.3.

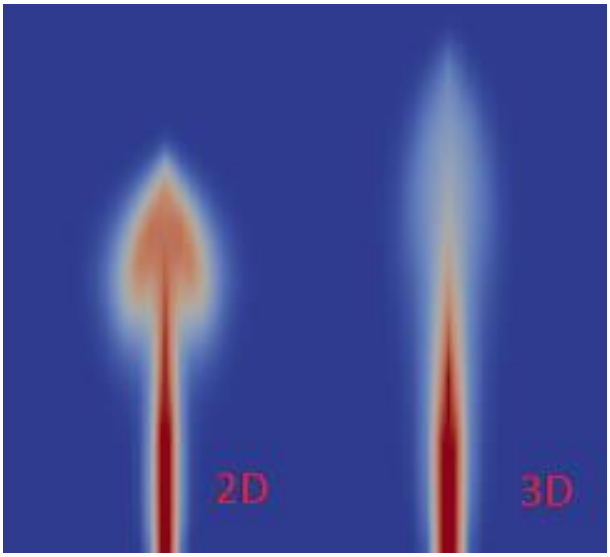

Figure 9 Initial bubble plume after 4 seconds

When the plume hits the surface the surface tension will create a strong lateral flow of both water and air. The bubble plume reaching the surface can be seen in Figure 10. The 2D case hits the surface after 8 seconds, while the 3D case first hits after 10 seconds. The 3D case have the highest acceleration, while the 2D case will have the largest velocity throughout the rise of the plume. After hitting the surface the bubble plume underneath will become more unstable, and you can observe some sway in flow. From Figure 11 we can see that this is markedly larger in the 2D simulations.

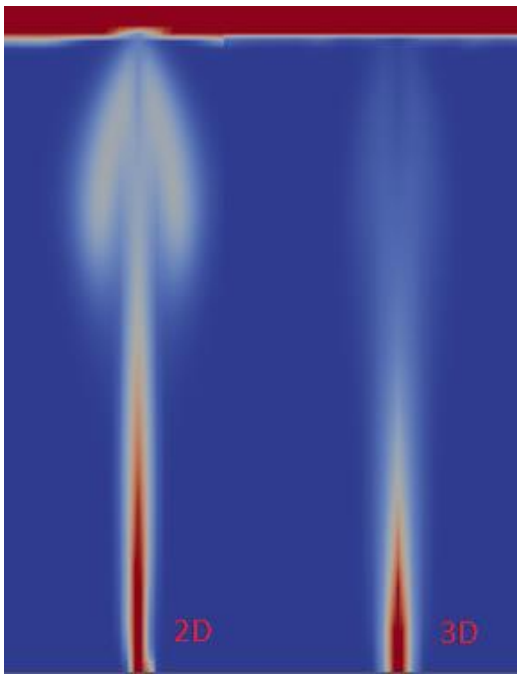

Figure 10 Bubble plume after reaching surface at respectively 8 and 10 seconds

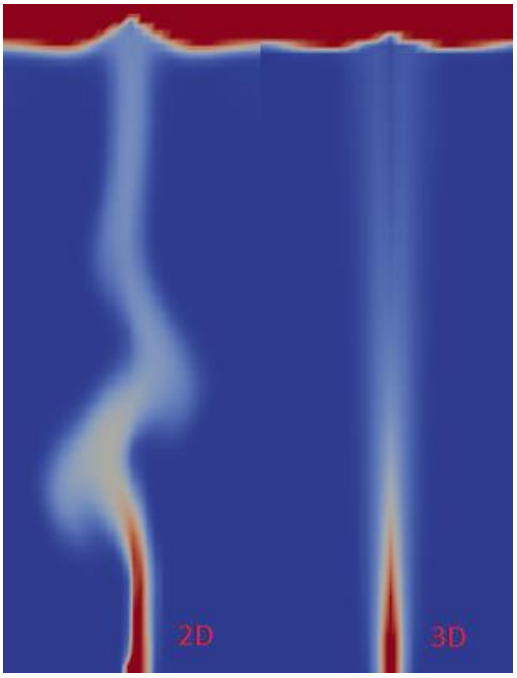

Figure 11 Bubble plume after 14 seconds

After some time the plume will stabilize and you can only see small swaying motions. From Figure 12 you can observe that after 30 seconds both 2D and 3D seams to flow relatively steady. Higher flow rates gives more and larger swaying motions, particularly for the 2D cases. Figure 12 also shows that the fountain at the surface is significantly larger for the 2D case. In fact, you cannot observe any disturbances in the water surface for the steady flow in the 3D case. The lateral flow of air and water at the surface will continue, but not as much as the immediate flow when the plume first hit the surface. The lateral flow will mostly consist of water, but some air will be dragged along and rise through the surface away from the boil region.

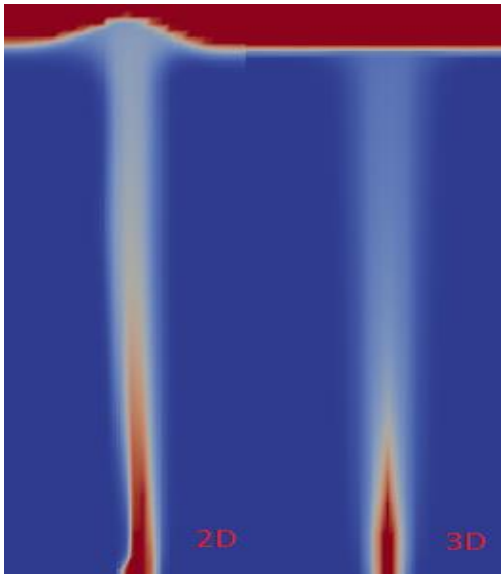

Figure 12 Bubble plume after 30 seconds

In Figure 13 you can see the gas velocity at the end time, 50 seconds. This also shows that the 2D case is generally more unstable. As mentioned above, one can observe that the 3D case have a higher acceleration, but the 2D case have a higher velocity throughout the plume. The higher velocity is indicated by a stronger red color. For the 2D case you can clearly see the lateral flow of air, while the 3D case seems to have a much smaller flow. Off course one have to remember that for the 3D case the air can flow in all directions, and this is just a cross-section of the plume. In Figure 14 you can see the velocity averaged over time. You can observe that the 3D case is very similar to the one in Figure 13, which further shows that this case is much more stable.

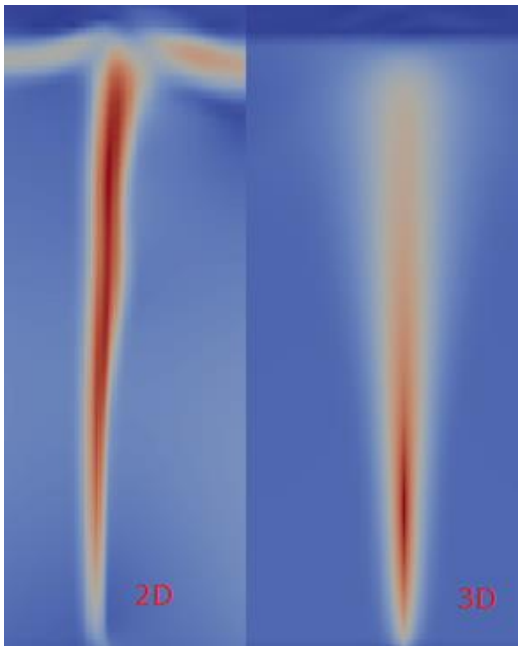

Figure 13 gas velocity after 50 seconds

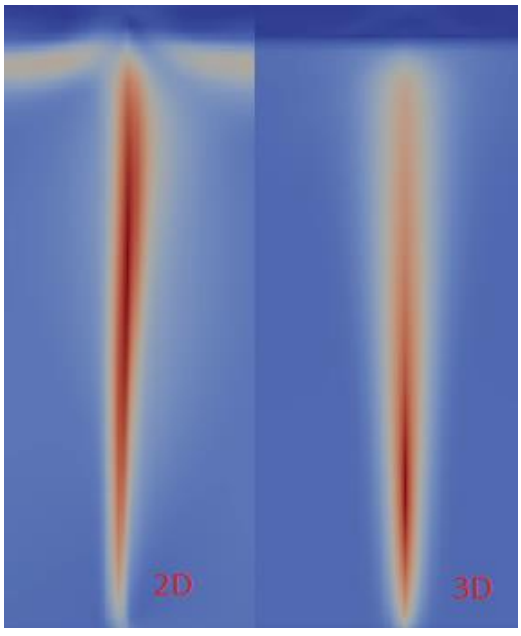

Figure 14 gas velocity averaged over time

The turbulence is calculated for the continuous phase. Figure 15 and 16 shows the values for turbulent dissipation rate (epsilon) and turbulent kinetic energy (k) a few seconds after the plume hits the surface. It can be observed that the values for the 3D case are 2-3 times higher than that of the 2D case. This could be explained by the fact that the 3D case have velocity vectors in all directions, and thus the turbulence will be higher at a given point. Figure 17 shows the epsilon values for the 3D case after 50 seconds. These figures clearly shows that turbulence is successfully calculated throughout the simulations. They also show that you get the highest values at the boiling region at the surface and some height above the inlet, which is expected.

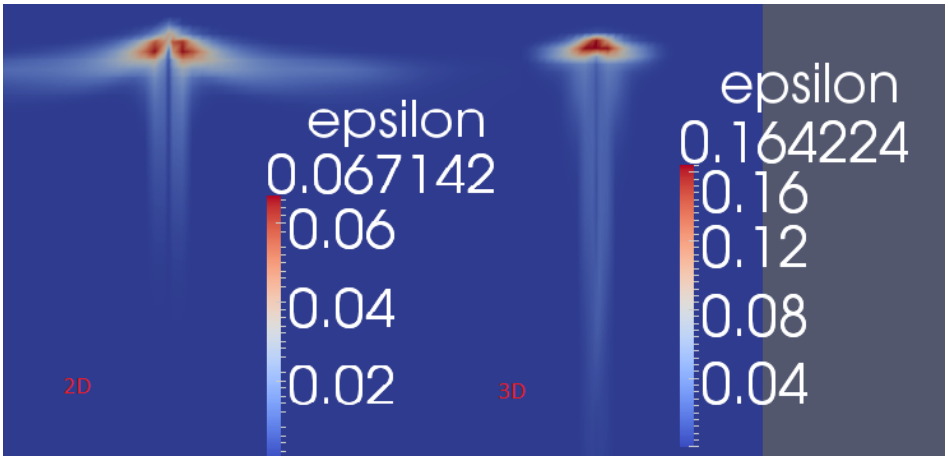

Figure 15 Epsilon values a few seconds after the plume hits the surface

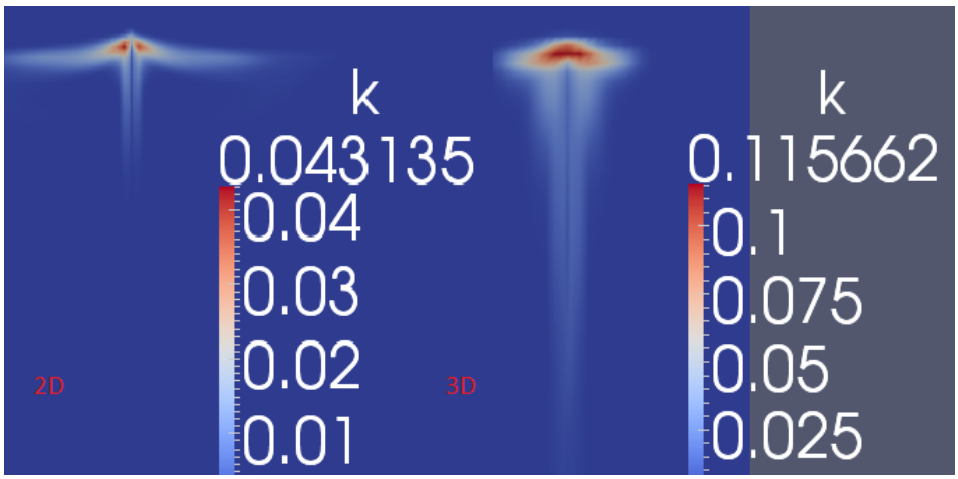

Figure 16 k values a few seconds after the plume hits the surface

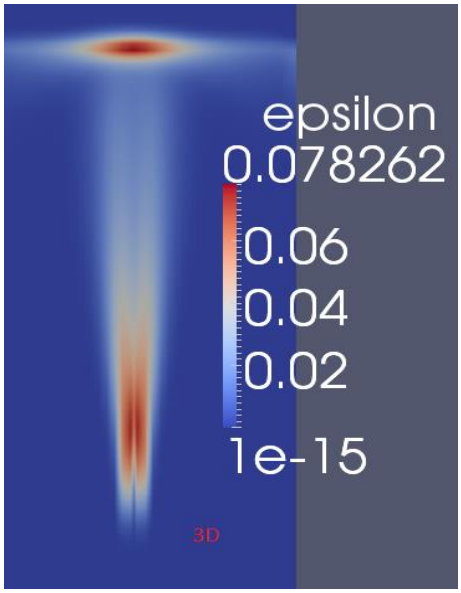

Figure 17 Epsilon values after 50 seconds for Case 2 in 3D

4.2. Velocity

In this section the velocity for the different cases will be evaluated. It is of interest to look at the velocity profiles both horizontal and vertically. All the data are taken from the velocity averaged over time. The horizontal profiles for both gas and liquid are measured at 9 meter water height. By looking at both the gas and liquid velocities, it is possible to evaluate the slip velocity for the different cases. Earlier it is mentioned that we assume a Gaussian profile for the velocity. This assumption is evaluated in the following. The vertical velocity profiles are compared against the Fanneløp and Sjøen (1980) experiment for each of the four cases.

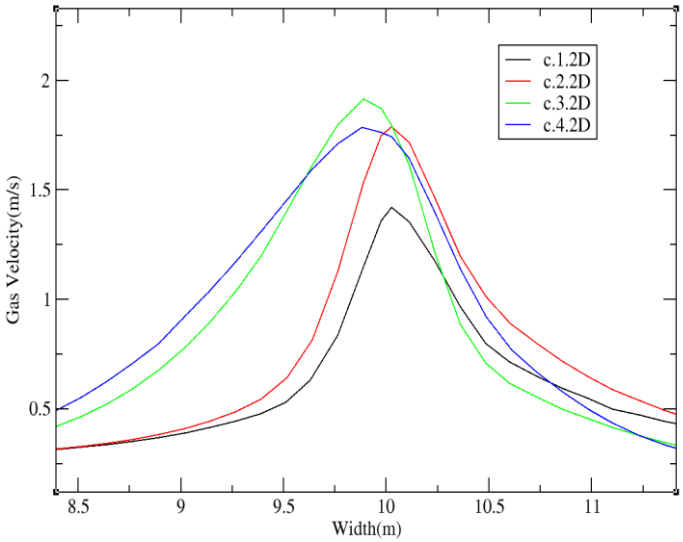

Figure 18 Gas velocity at 9m for the 2D cases

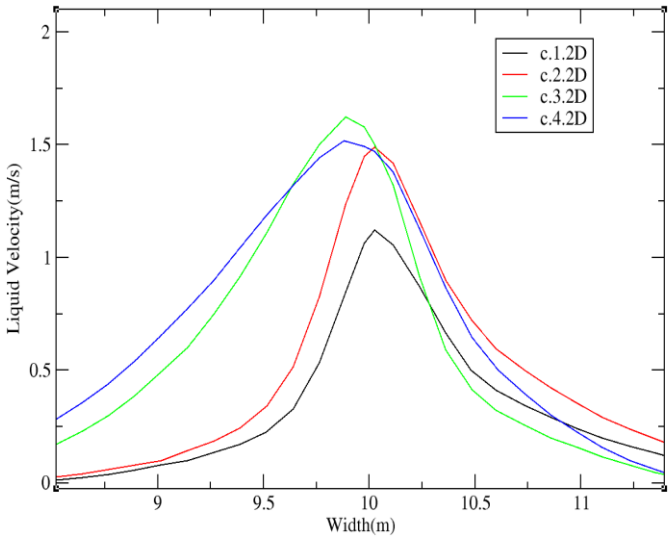

Figure 19 Liquid velocity at 9m for the 2D cases

Figure 18 and 19 shows the velocity of gas and liquid respectively for the 2D cases at 9 meter water height. It is clear that the 2D cases is not completely stable, and that you will see some fluctuations that will affect the velocity profiles. You can observe that some of the cases are off set in relation to the center of the width at 10m. For Case 1-3 you can see an increasing velocity, while Case 4 is actually showing lower velocity than Case 3. This is mainly because of more swaying and instability with increasing flow rate. The impact of the walls are expected to be larger with higher flow rates. The lateral flow of water at the surface will hit the walls, go down along the walls, and when it hits the bottom it will flow back into the plume and influence the flow. Based on this the results of case 4 is not considered to be accurate. If the domain were larger, one would expect Case 4 to show higher velocities than Case 3. All the profiles shows similarities to a Gaussian profile. The centerline velocity of gas and liquid shows that the slip velocity is approximately 0.3m/s for all the four cases, which is consistent with theory.

Figure 20 and 21 shows the velocity of gas and liquid respectively for the 3D cases at 9 meter water height. As seen in the general observations the 3D case showed a lot less swaying motion. This is clearly seen from the velocity profiles. The different cases with different flow rates is clearly indicated by the increasing centerline velocity. The maximum velocities are perfectly aligned with the centerline of the domain. By considering the centerline velocities for gas and liquid, it shows that also the 3D cases present a slip velocity of approximately 0.3m/s. All the profiles shows similarities to a Gaussian profile.

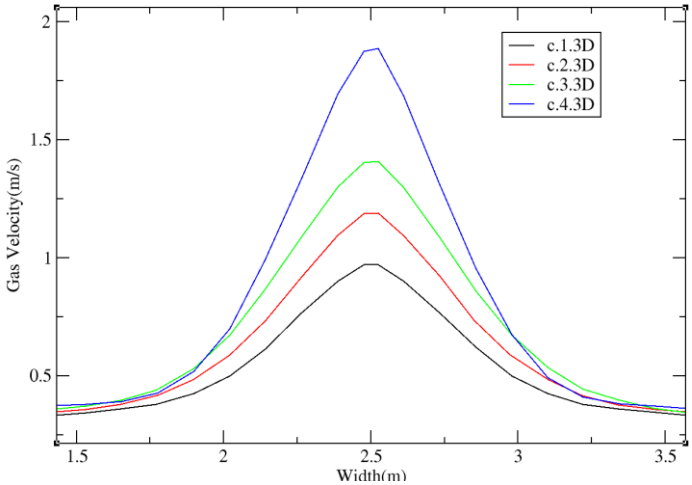

Figure 21 Gas velocity at 9m for the 2D cases

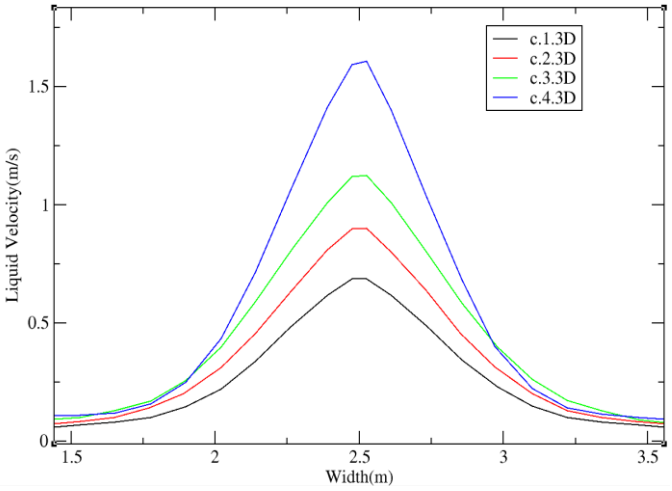

Figure 20 Liquid velocity at 9m for the 3D cases

Figure 22-25 shows the vertical gas velocity averaged over time along the centerline for cases 1-4. For each of the cases the results from the 2D case and the 3D case are plotted against the experiments of Fanneløp and Sjøen (1980). Z denotes the source distance, which is 10m. The data from the experiments are gathered at 0.75, 2.35, 3.95, 5.55, 7.15 and 8.75 meters. The results clearly show large discrepancy between the simulations and the experiments. As seen in the general observations, the 3D case has a higher acceleration, while the 2D case has a higher velocity overall. We can also observe that the 3D cases more quickly will decrease in velocity when reaching the maximum velocity. The exception here is case 4. By starting the decrease earlier, the 3D cases will get closer to the experimental results as the plume approaches the surface. The results of the experiments give the impression that the maximum velocity is far lower than the simulations. In

reality the experiment will have a higher initial velocity. This will happen very close to the source, and the plume will immediately slow down and buoyancy will take over. It seems clear that the 3D cases is at least producing the most realistic results. Further discussions on why the simulations are over-predicting the velocity throughout the plume is presented in section 4.6.

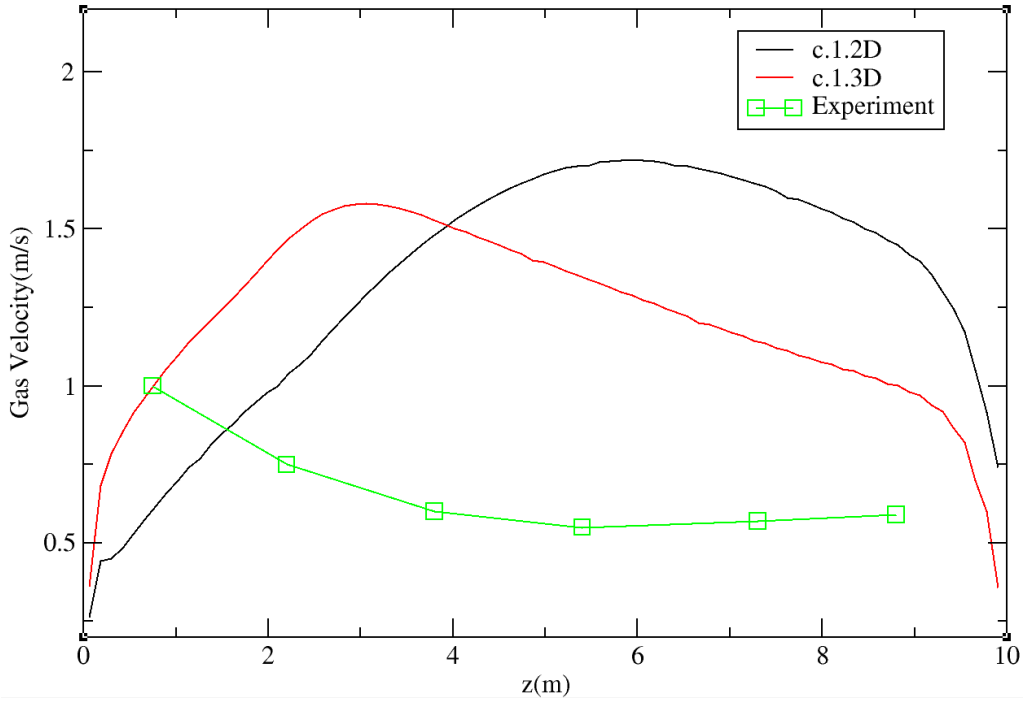

Figure 22 Vertical gas velocity averaged over time for Case 1

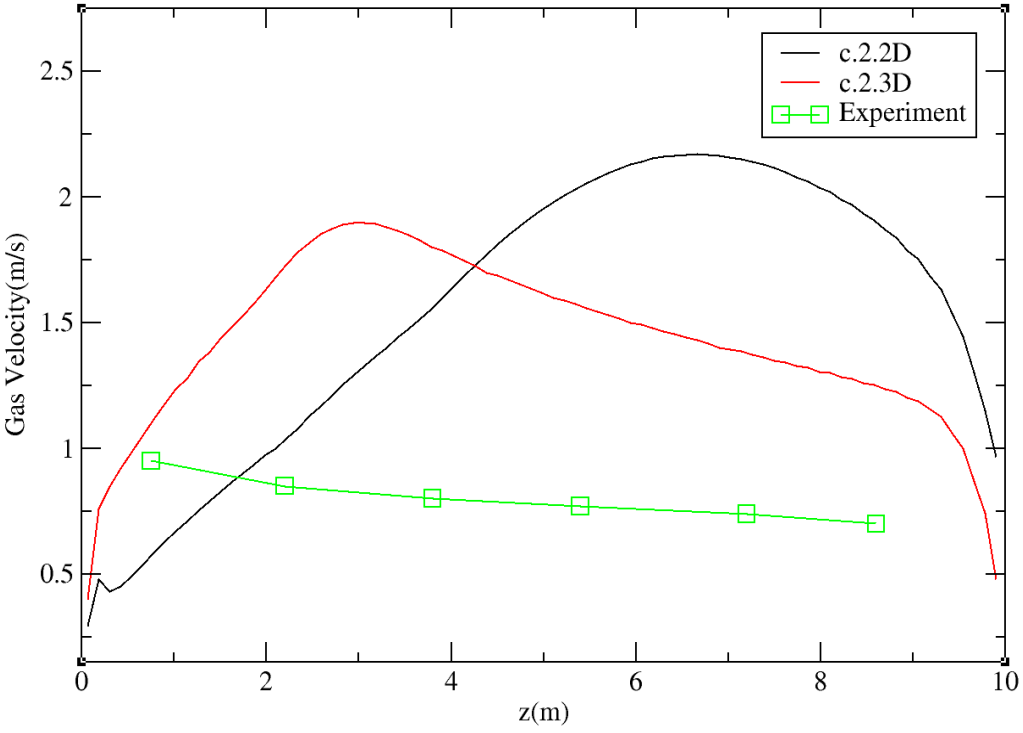

Figure 23 Vertical gas velocity averaged over time for Case 2

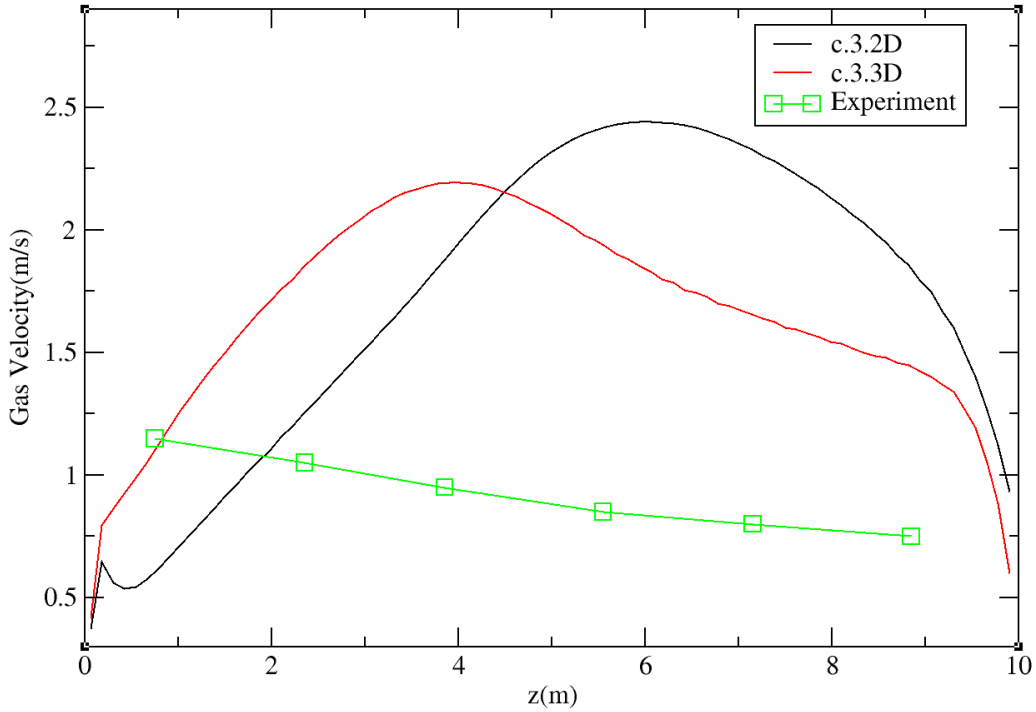

Figure 24 Vertical gas velocity averaged over time for Case 3

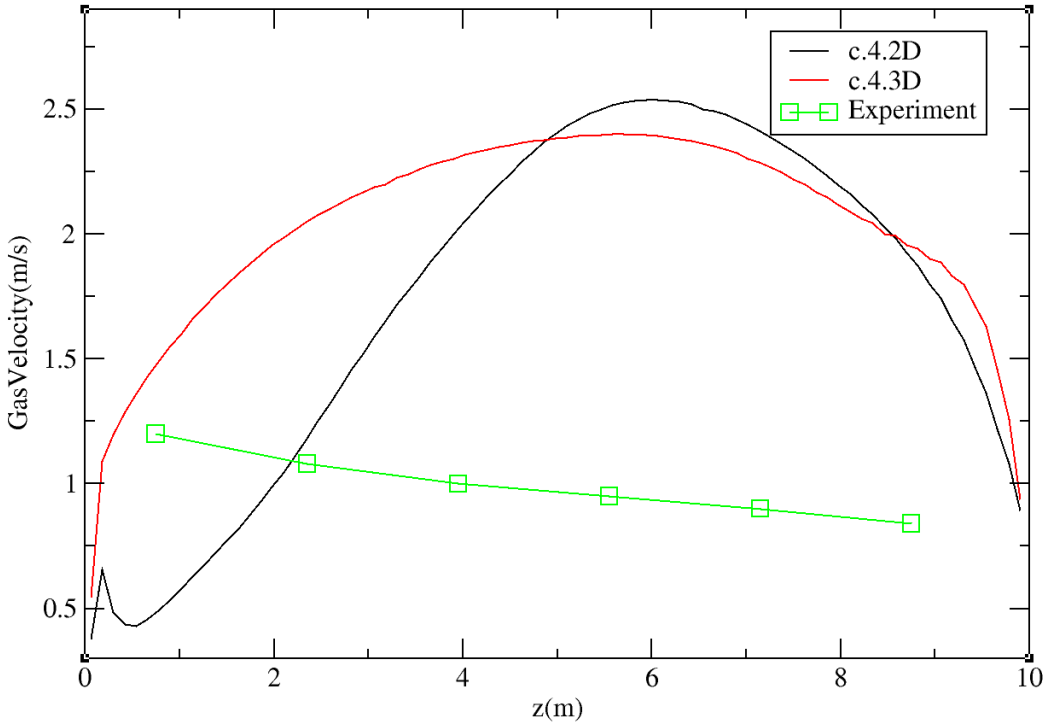

Figure 25 Vertical gas velocity averaged over time for Case 4

4.3. Void Fraction

In this section the void fraction for the different cases will be evaluated. It is of interest to look at the void fraction both in horizontal profile and vertical profile. All the data are averaged over time. The horizontal profiles are measured at 9 meter water height. Earlier it is mentioned that we assume a Gaussian profile for the velocity. This assumption is also known to be valid for the void fraction. The vertical void fraction along the centerline is compared against the Fanneløp and Sjøen experiment for each of the four cases.

In figure 26 and 27 you can see the void fraction at 9m for the 2D and 3D cases respectively. It can be observed that the profiles have some irregularities. There is no apparent reason for this. In the 2D cases there seems to be an elevation in void fraction close to the centerline that is not consistent with a normal Gaussian fit. In the 3D cases there is a clear dip in void fraction along the centerline. When not considering the irregularities, the profiles seems to have some similarities to a Gaussian fit. Case 4 in 2D seems to have unrealistic high void fraction at the sides. As mention for the velocity profiles, this is most likely because of instability caused by the high flow rate in combination with the influence of the walls. It can be observed that the void fraction for the 2D cases is about 2 times the value of the 3D cases. This could be explained by the fact that in 3D the air is allowed to spread in 3 dimensions, and the profile presented here is just a cross-section of the plume. For the 3D cases it can be observed some strange behavior in addition to the already mentioned irregularities. Case 2 and 3 seems almost identical, and it is actually Case 2 that shows the highest values for gas concentration along the centerline. There is no clear reason for this. The void fraction along the centerline for case 4 is similar to the ones of case 2 and 3. This behavior is not expected after observing the velocity profile and how stable the 3D cases are in general.

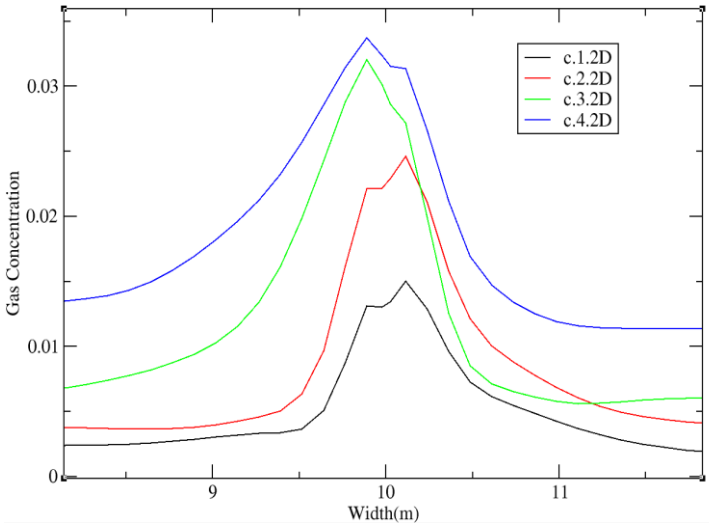

Figure 26 Void fraction for 2d cases at 9m

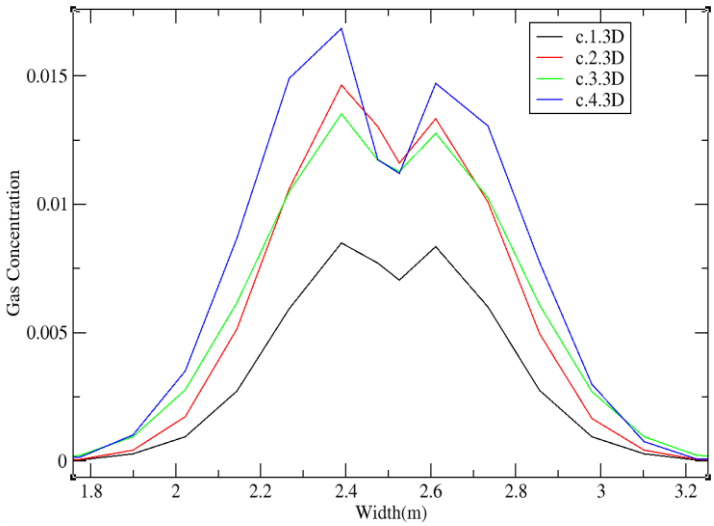

Figure 27 Void fraction for 3D cases at 9m

Figure 28-31 shows the void fraction along the centerline for all the cases compared to the Fanneløp and Sjøen experiment. The experiments are measured at a height of 2.35, 3.95, 5.55, 7.15 and 8.75 meters. Since there only are comparable data from 2.35 meters, the figures are focused on this area. Obviously all the profiles will experience a steep drop from 1 and down to where they are entering

the graphs displayed in the figures. As you also can see in Figure 26 and 27, the void fraction for the 2D cases are approximately a magnitude of 2 higher than the 3D cases along the centerline from about z=4. The 3D cases seems to have a profile similar to the experiments. The 3D cases over-predicts the void fraction, but it gets closer when approaching the surface.

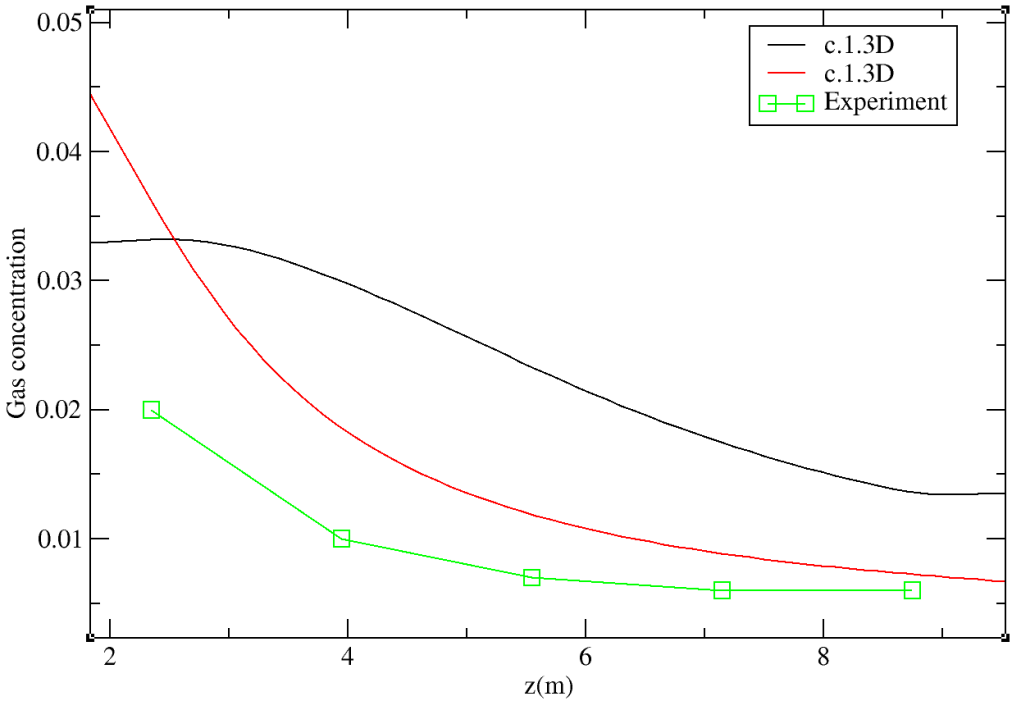

Figure 28 Void fraction averaged over time for Case 1

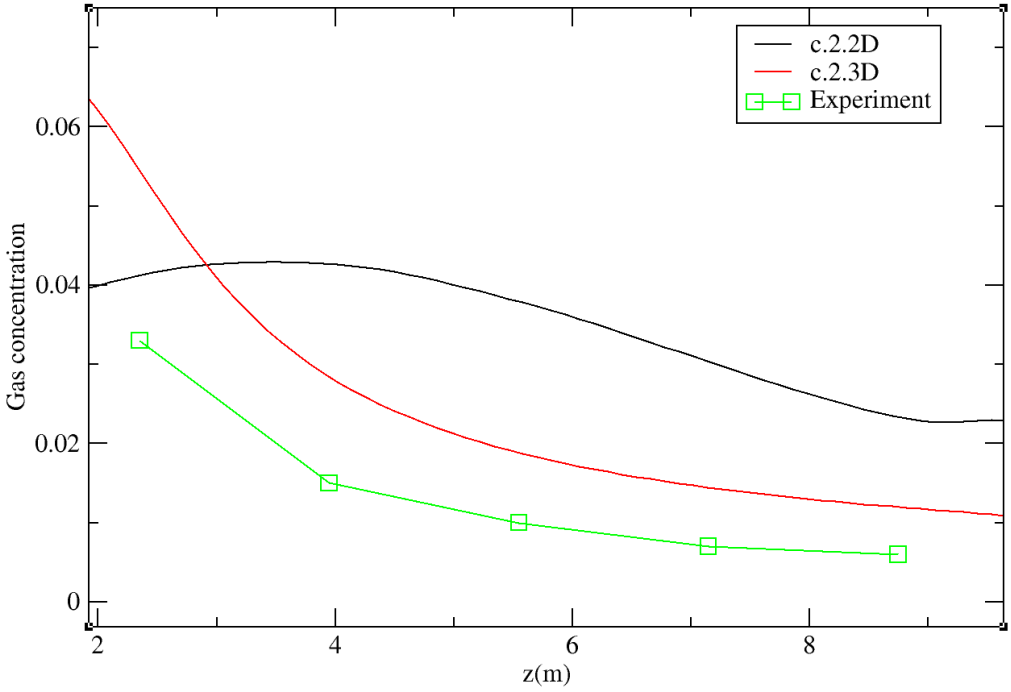

Figure 29 Void fraction averaged over time for Case 2

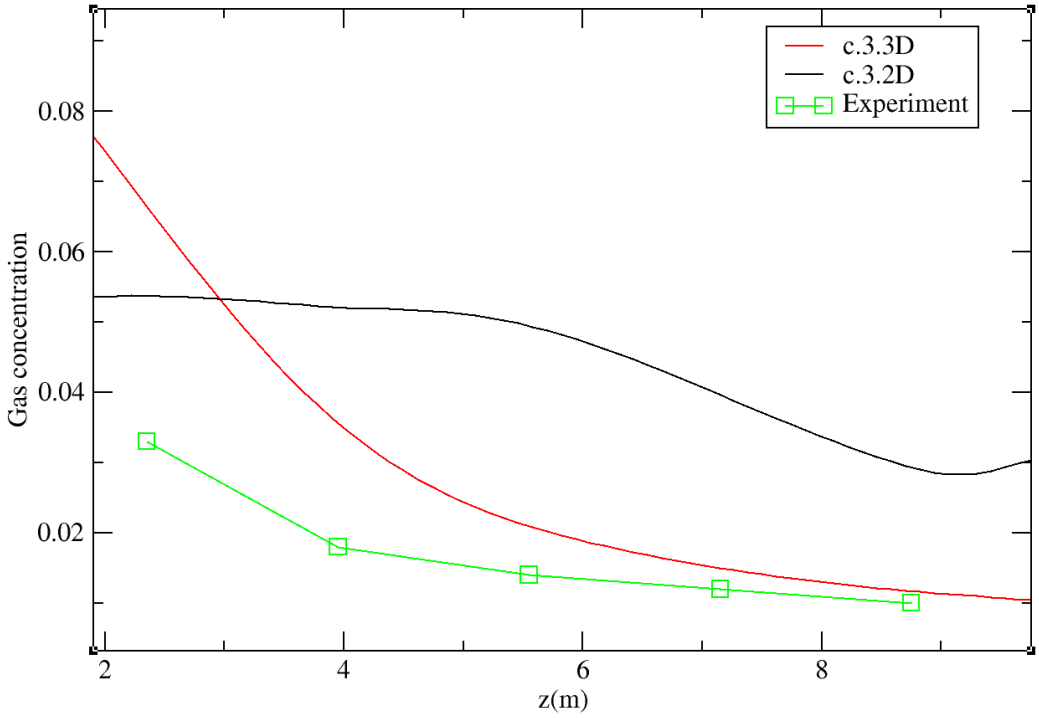

Figure 30 Void fraction averaged over time for Case 3

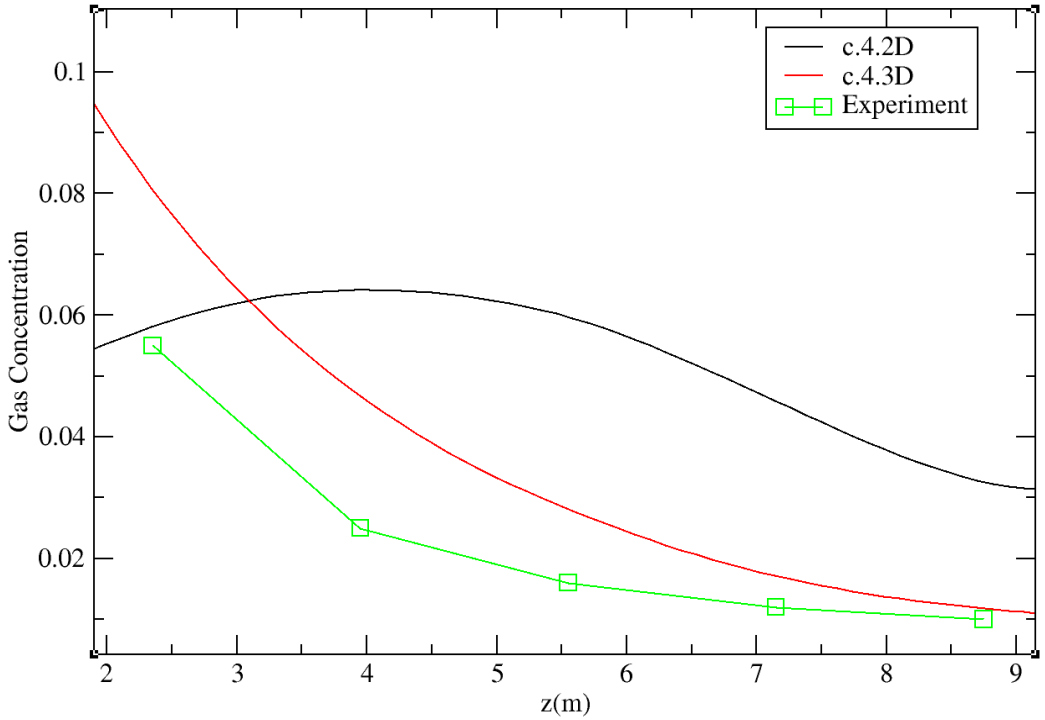

Figure 31 Void fraction averaged over time for Case 4

4.4. Plume Radius

In this section the plume radius of the different cases will be evaluated. The plume radius is an important measure, as it defines the boiling region at the surface. This parameter is then further used for consequence analysis. It is of interest to see how the 2D and 3D cases performs in relation to the experiments. Figure 32 shows the results of the Fanneløp and Sjøen (1980) experiments. The data from the experiments is based on the velocity profiles measured at 0.75, 2.35, 3.95, 5.55, 7.15 and 8.75 meters. From the figure you can clearly see a small increase in the radius following the increased flow rate. This indicates that also the entrainment coefficient is increasing slightly with increasing flow rates. The entrainment coefficient for the four cases where in the range of 0.07-0.1, where Case 1 was around 0.07 and Case 4 close to 0.1. I will come back to the entrainment coefficient in section 4.6. The plume radius in the experiments is in the range of 0.7-1.1m, where Case 1 is around 0.7m and Case 4 is around 1.1m.

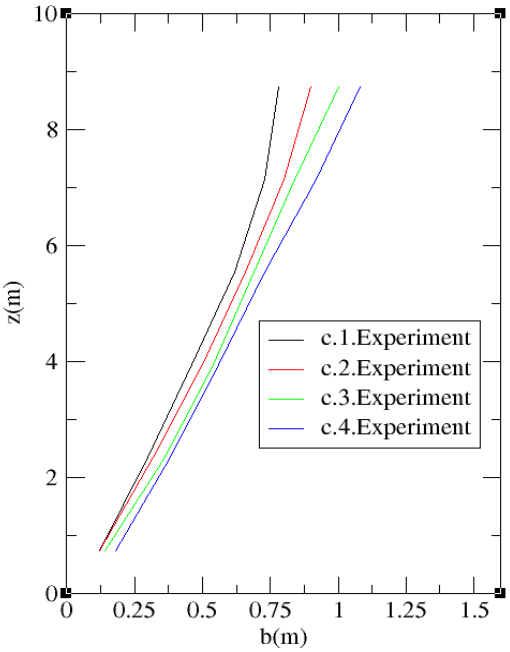

Figure 32 Plume radius for the Fanneløp experiment (Fanneløp and Sjøen, 1980)

Figure 33 and 34 shows the plume radius for the 2D and 3D cases respectively. The data is gathered at three different heights, namely 3, 6 and 9 meters. The plume radius is measured in accordance to Fanneløp (1994), which indicates that the plume ends where the velocity has decreased to half of the maximum velocity at the horizontal cross-section. It is clear that the simulations in general produce unrealistically low values for the plume radius, both in 2D and 3D. For the 2D cases the plume radius at 9 meter is between 0.45-0.55 meters. The exception here is Case 4, which is believed to be significantly influenced by the walls, leading to less accurate results. For the 3D cases, all seems to have a plume radius at 9 meter close to 0.5 meters. The experiments showed an increase in plume diameter as the flow rate increased, but the simulations does not seem to show the same tendency. You can observe some indications for the 2D cases, but not as clearly as in the experiments.

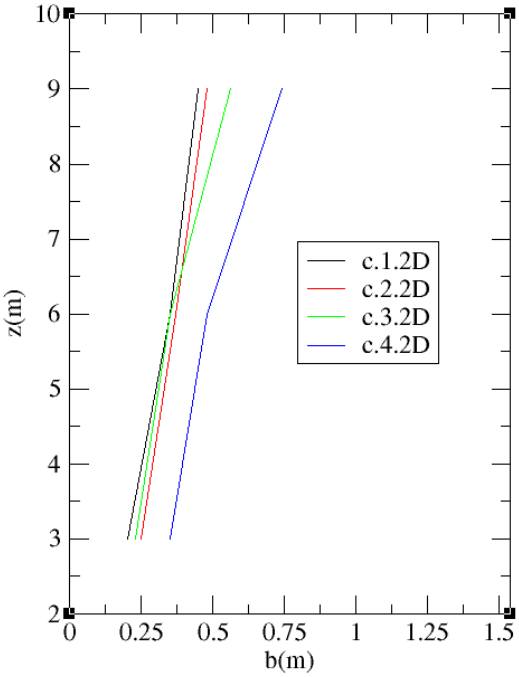

Figure 33 Plume radius for cases run in 2D

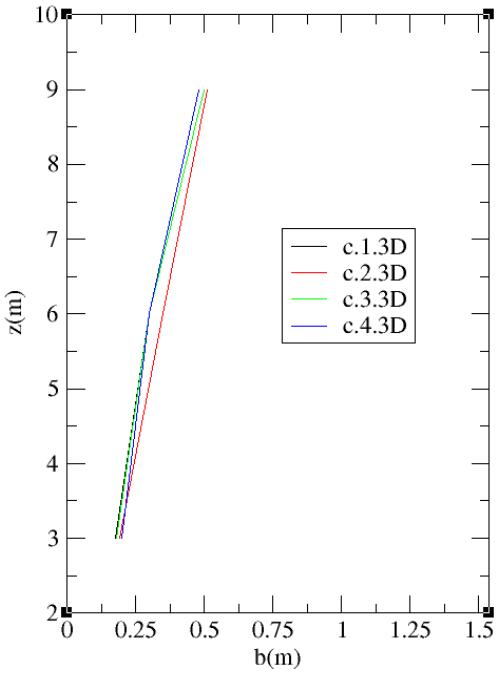

Figure 34 Plume radius for cases run in 3D

4.5. Summary and Discussion

In general the simulations over-predicts the velocity along the centerline of the bubble plume, and under-predicts the spreading of the plume. Since the spreading is much smaller for the simulations, the higher void fractions seen in section 4.4 is expected. It is clear that the 3D cases produces the best results. They show best fit to Gaussian profiles. The velocity accelerates faster and decreasing earlier, more similar to the experiments. The void fractions along the vertical centerline shows a similar curve to the experiments. The 2D cases seems more unstable and more influenced by the walls, especially with increasing flow rates. This implies that the 3D cases are best suited for further discussion on the results.

It is expected that there is a clear relation between the velocity profile and the spreading of the plume. The spreading will slow down the plume faster, and the buoyancy of the bubbles will take over. If water is not entrained as expected, there will be less resistance for the initial jet created at the source. Also the assumption of constant bubble size can affect the velocity. Fanneløp and Sjøen (1980) reported a bubble size of about 1cm at the inlet. The breakup of the bubbles and the entrainment of water will slow down the initial jet. In the simulations presented here, a 3mm constant bubble size is used. The assumed constant bubble size is believed to have some influence on the small spreading of the plume. The very important entrainment coefficients seems to be too low for the simulations. The experiments showed entrainment coefficients between 0.07 and 0.1. The entrainment coefficient has been found to increase with increasing flow rate. This is not clearly indicated by the simulations since the diameter off the plume is rather similar for all the cases, especially in 3D. It is possible to calculate the entrainment coefficients for the simulations in 3D by using the relation derived from Fanneløp and Sjøen (1980):

$$D = 2 * \alpha * z \quad \textbf{(48)}$$

Where D is the plume diameter, α is the entrainment coefficient and z is the source depth. The calculations is based on the data gathered at 9m water height, and therefore z is set to 9. Figure 35 shows the entrainment coefficient plotted against the flow rate for both the experiments and the 3D cases. This shows clearly what is suspected, the entrainment coefficient is too low and it does not show a trend of increasing with higher flow rate. It can be observed that there is a correlation in the differences in the plume diameters and entrainment coefficients when looking at the experiments compared to the 3D cases. This is off course expected since the relation in Eq.48 suggests that these parameters are dependent on each other. The low entrainment coefficient clearly indicates that there is too little lateral inflow of water into the plume. The reason for this could be a combination of the turbulence model used, the boundary conditions and inherent limitations in the `bubbleFoam` solver. E.g. the solver only has the drag model SchillerNaumann hardcoded. If one look at e.g. Dhotre et al. (2012), different approaches to drag are applied.The turbulence model has been very unstable for the phenomena studied, and it is uncertain whether or not it produces realistic values for k, epsilon and the viscosity. The boundary conditions defined as walls does not allow water to flow into the domain. This influence the waters ability to laterally flow into the plume. In an unconfined place the water is allowed to flow into the plume locally but also a large distance away from it. The experiments performed by Fanneløp and Sjøen reduced this problem by having "unlimited" distance in two of the directions. This combined with 10 meter width in the last two directions is believed to have reduced this problem significantly.

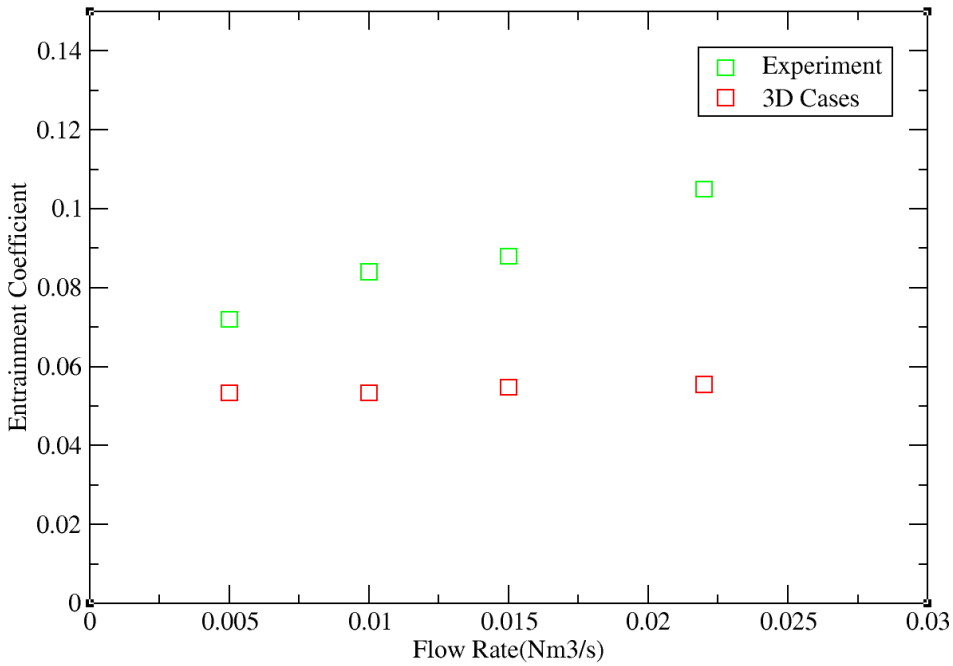

Figure 35 Entrainment coefficient plotted against flow rate for experiment and 3D cases

It is observed that both the 2D and 3D cases have a slip velocity at the centerline of about 0.3m/s. This is consistent with theory, e.g. Fanneløp and Bettilini (2007) and Levich (1962). The slip velocity is

not as important as the entrainment coefficient, but it is worth noticing that the results produce accurate values for this parameter.

To summarize the results the simulations have some similarities to the experiments. The slip velocity show promising agreement with the theory. On the other hand, the most important parameters are clearly under-predicted. The spreading of the plume and the entrainment of water is on average about half of what is reported in the experiments. This leads to a higher void fraction, on average a factor of 2. This is believed to be affected by the turbulence model used, the boundary conditions and inherent limitations in the `bubbleFoam` solver. Also the velocity profile along the centerline is clearly over-predicted in the simulations. This is believed to be affected by the same things mentioned above, but also the constant bubble diameter of 3mm used. Other limitations on the results is the case setup specified in Chapter 3. Fanneløp and Sjøen used a perforated plate with 150 1mm holes. In this setup I have used a squared inlet with 100% air injected. A different inlet condition could influence the results. This and other alternative approaches are considered in the next chapter. Further implications of the results are discussed in the conclusion in chapter 6.

5. Alternative Approaches

In Chapter 4 I have presented the results achieved with the `bubbleFoam` solver with a k-epsilon turbulence model. I have chosen the parameters and coefficients that is believed to give the best results. In this chapter I will look at some different approaches using OpenFOAM to simulate a bubble column. It could be of interest for future work to see how alternative approaches affect the results of the simulation. I will use Case 2 from the experiments (Fanneløp and Sjøen, 1980) to compare the alternative approaches against the 3D case presented in Chapter 4. All alternative cases will be run in 3D, using the same mesh as presented in Chapter 3.

Fanneløp and Sjøen (1980) used a perforated plate with a diameter of 100mm consisting of 150 holes of 1mm. This indicates that 0.015% of the plate is covered by the actual inlet. I do not have the possibility to create a similar perforated plate, but I can set the boundary conditions to injecting 0.015% of air and the rest water. This is based on a squared inlet of 100mm, as in the main simulations. The velocity of water is set to 0. To get the same flow rate I have to adjust the inlet velocity of air from 0.505 m/s to 33.66 m/s. It is of interest to see how the changed inlet conditions will affect the spreading of the bubble plume. Better initial spreading will slow down the plume and the bubbles buoyancy will take over.

It is further of interest to run a laminar case with increased viscosity of water. The goal of this approach is to get a more fluctuating flow of air, and see how that influences the spreading of the plume. This is an alternative to applying a large eddies simulation (LES) turbulence model, which is not available for the relevant solvers in OpenFOAM. LES might be a more realistic approach to turbulence, and is used in a number of resent research into bubble plumes, see e.g. Dhotre et al. (2012). The viscosity (`nu`) of the water phase is changed from 1x10-6 to 2x10-6. For the laminar flows I have used an alternative solver, the `twoPhaseEulerFoam`. This is a further development of the `bubbleFoam` solver that gives you some more functionality. The reason for this change is that I could not make the flow of air fluctuate as intended using the `bubbleFoam` solver, as seen in Figure 37. This is clearly a weakness of this solver. The results should provide some indications on further potential for using OpenFOAM to simulate bubble columns. In Table 7 the alternative cases and their characteristics can be viewed. The results of the cases are presented in the following subchapter.

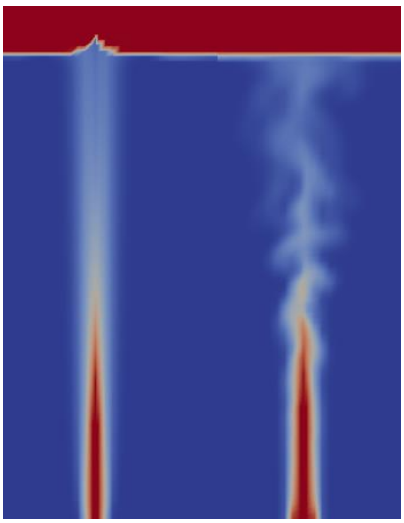

Figure 36 Laminar flow using `bubbleFoam` and `twoPhaseEulerFoam`

Case	Solver	Turbulence	Flow rate	Characteristic
c.2.3D.0.015air	bubbleFoam	k-epsilon	0.01Nm^3/s	Changed inlet conditions
c.2.tP.2nu	twoPhaseEulerFoam	no	0.01Nm^3/s	Laminar flow. Increased viscosity of water phase.
c.2.tP.2nu.0.015air	twoPhaseEulerFoam	no	0.01Nm^3/s	Laminar flow. Increased viscosity of water phase. Changed inlet conditions.

Table 7 Overview of cases run as alternative approaches

5.1. Results

In this subchapter the results of the alternative cases will be presented. The alternative cases are compared against Case 2 of the experiments (Fanneløp and Sjøen, 1980) and Case 2 from the simulation presented in Chapter 4.

5.1.1. Velocity

In Figure 38 one can observe the horizontal velocity profiles of the alternative cases and the original case from Chapter 4 at 9 meter. It is observed that by changing the inlet conditions for the original case, the velocity is reduced. This is expected since the initial spreading of the air is believed to be higher. By introducing the laminar flow with higher viscosity for the water phase, the velocity is further reduced. This is believed to be a result of the fluctuating air moving upwards, as seen on the right side in Figure 37. By combining the changed inlet conditions and laminar flow, the velocity is showing even lower values. From Figure 39 one can see that this value is actually very close to the experimental results at 9 meter. All the cases shows a profile similar to a Gaussian fit.

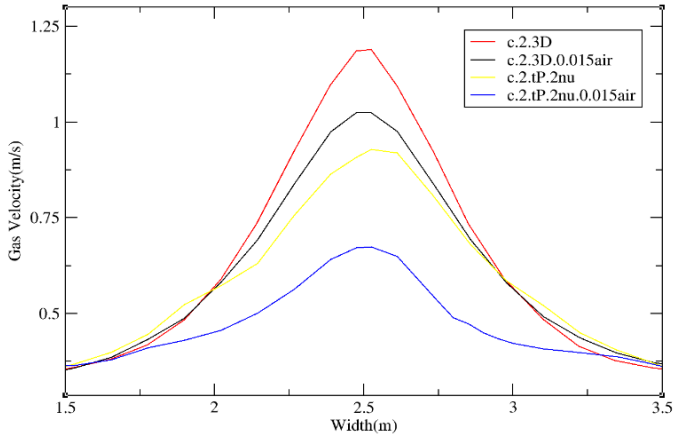

Figure 37 Velocity profile of alternative cases at 9m

Figure 39 shows the vertical velocity profile of the alternative cases and the original case plotted against the experiment. The original case with changed inlet conditions shows a small decrease in velocity along the centerline of the plume. The laminar case first shows a similar velocity as the original case, before it decreases from around z=5m. This clearly shows the point where the fluctuation of the flowing air starts. The start of the fluctuation can be observed on the right hand side of Figure 37. The laminar case with changed inlet conditions shows the best agreement with the experiment. It over-predicts the velocity close to the inlet and under-predicts the velocity around z=5, while at the end it gets very close to the experiment.

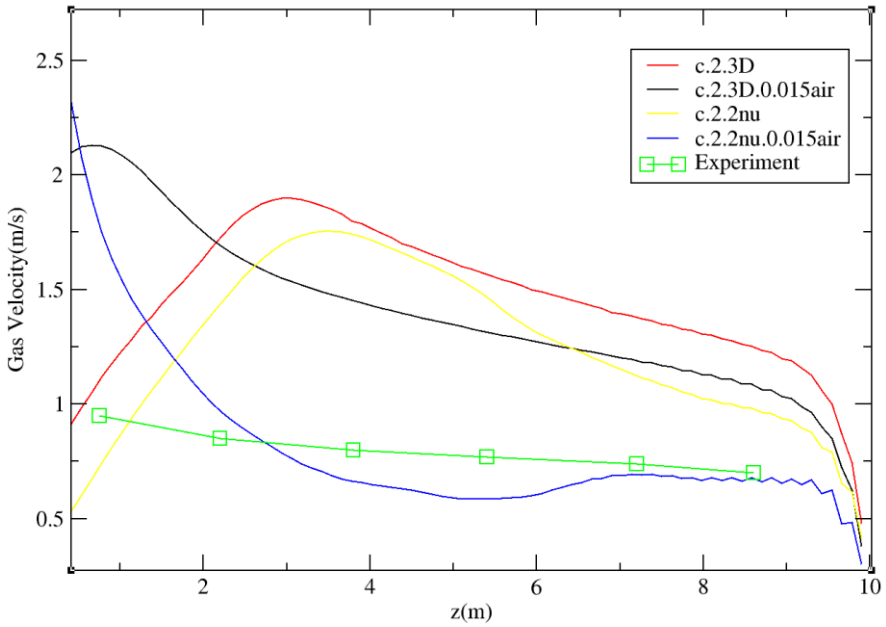

Figure 38 Vertical velocity profile of alternative cases compared against experiment

5.1.2. Void Fraction

In Figure 40 one can observe the horizontal void fraction of the alternative cases and the original case from Chapter 4 at 9 meter. The original case with changed inlet conditions shows very high values at the centerline. The laminar cases where there is a lot more fluctuation shows considerably smaller values than the original case. From Figure 41 it can be observed that the values of the laminar cases are close to the experiment at 9 meters.

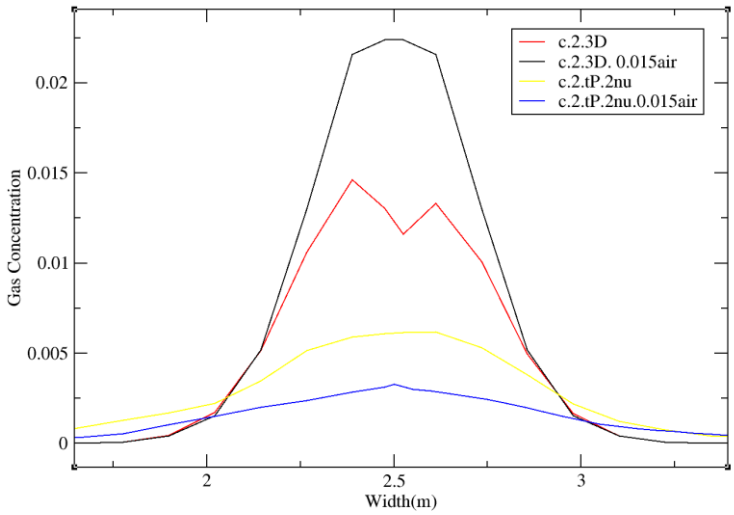

Figure 39 Void fraction of alternative cases at 9m

Figure 41 shows the void fraction along the centerline for the alternative cases compared against the experiment. As also seen in Figure 40, the original case with changed inlet conditions shows very high values throughout the whole plume. The laminar case without the changed inlet condition over-predicts the void fraction along the centerline, while the laminar case with changed inlet condition under-predicts the values along the centerline. All cases shows a similar curve to the experiment.

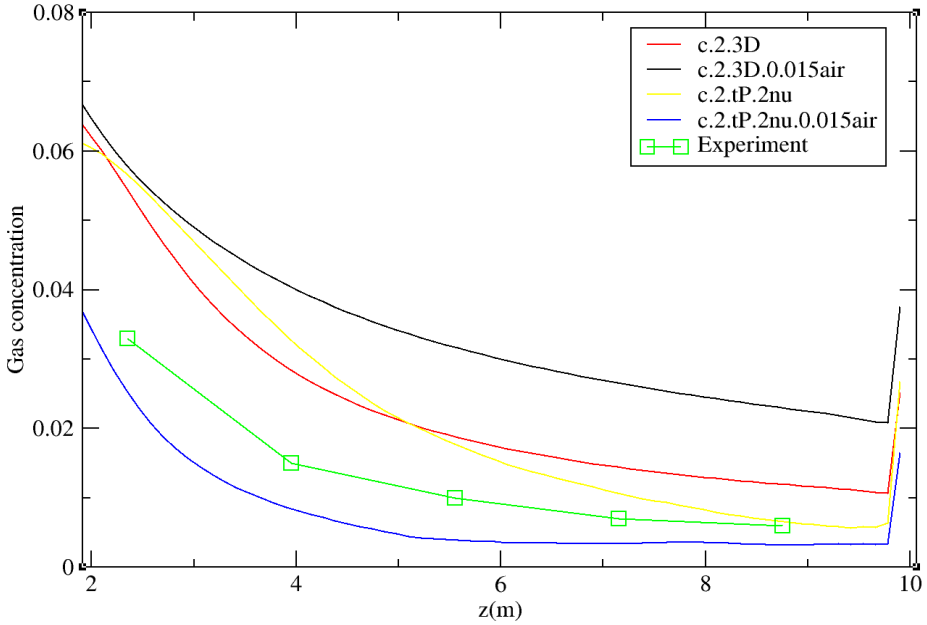

Figure 40 Void fraction along centerline for alternative cases compared against experiment

5.1.3. Plume Radius

Figure 42 shows the plume radius of the alternative cases, the original case from Chapter 4 and the experiment. The original case with changed inlet conditions shows a slight increase in the plume radius throughout the whole plume. This is believed to be caused by the increased initial mixing of air and water. The laminar case with original inlet conditions show a similar radius as the original case for the first part of the plume, while it increases at the end. This is believed to be caused by the increased fluctuation starting around z=5, as seen in Figure 37. The laminar case with changed inlet conditions shows a large plume radius throughout the whole plume, considerably larger than the experiment. This is believed to be caused by the fluctuation combined with the initial mixing of air and water.

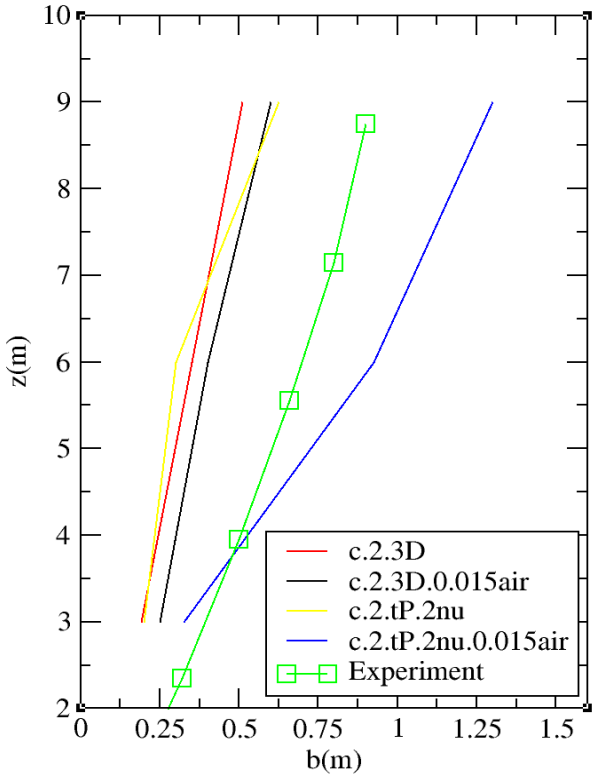

Figure 41 Plume radius of alternative cases compared against experiment

5.2. Summary and Discussion

In this chapter I have presented some alternative approaches to set up a bubble plume using OpenFOAM. I have presented a case where I have changed the inlet conditions to be more similar to that of the original experiment performed by Fanneløp and Sjøen (1980). I have in addition presented two laminar cases using the solver `twoPhaseEulerFoam`. For the laminar cases the viscosity of water is changed from 1x10-6 to 2x10-6 to make the flow of air fluctuate more. All of this is rather simple changes, where I focus on showing what can be done using the available solvers in OpenFOAM.

The case where I have changed the inlet conditions on the original case used in Chapter 4, shows no considerable improvements when comparing against the experiments. The velocity is slightly decreased, while the void fraction is increased. The best results is obtained from the laminar cases run with `twoPhaseEulerFoam`. The velocity of air and the void fractions are reduced. The laminar case with changed inlet conditions shows a vertical velocity profile close to the experiments. When it comes to plume radius, the same case shows a significant increase. It is considerably larger than the experiment, but it shows a good tendency. It is clear that the combination of laminar flow and changed inlet conditions shows the most promising results.

One could argue that some of the findings from this chapter could be used to optimize the cases used for comparison in Chapter 4. Unfortunately, there is no time for that. This chapter is meant as a quick look into what is possible to do in OpenFOAM, and inspire for future work. Further implications of the findings and the general conclusion can be found in Chapter 6. The findings from this chapter is also considered when discussing potential future work in Chapter 7.

6. Conclusion

In Chapter 4 one can observe the results obtained using the `bubbleFoam` solver in OpenFOAM. It becomes clear that the results are not in compliance with the experiment. In general the velocity of the bubble plume is too high throughout the plume, while the spreading of the air is too small. Too small spreading leads to high values for the void fraction. One of the most important parameter, namely the entrainment of water, is too small. While the experiment showed entrainment coefficients between 0.07 and 0.1 for the four cases, the simulations showed values around 0.05. For the simulations the entrainment coefficient did not increase with increasing flow rate, as in the experiments. Some of the less important parameters are showing decent agreement, like the slip velocity. The 3D cases showed better results than the 2D cases. They were more stable and showed a better fit to Gaussian profiles. The velocity and void fraction along the centerline showed better compliance with the experiment than the 2D cases. Since the spreading of the plume is too small, the deviation in void fraction is expected. It should be noted that if the spreading was similar to the experiments, the void fraction of the 3D cases is expected to be very close to the actual experimental results. The observed results seems very similar to the problems Fanneløp and Bettilini (2007) highlighted for numerical simulation of bubble plumes:

The rising fluid has the shape of a slender pencil and not that of a rapidly growing plume flow. The raising flow does not “entrain” fluid from the exterior water mass and there is too little inflow into what should be the plume. The problem appears to be related to the “distant” boundary conditions used in the simulation.

When looking at the results in Chapter 4 it is tempting to conclude that numerical simulation of a bubble plume is not good enough for practical application, but off course one have to remember that the results presented in this thesis is based on solvers available in OpenFOAM, and on the different conditions set by the user. I have limited background in fluid dynamics and programming. A more experienced user might have been able to modify the solver to better fit this specific phenomena. It should also be noted that for the simulations presented here a squared inlet is used, where 100% air is injected. In the real experiment a circular perforated plate was used. This may influence the initial mixing of gas-water that can slow down the plume and enhance the spreading. It should also be noted that other commercially available CFD codes might have the ability to provide better results. To study the effect of some of these limitations and to see if other approaches could enhance the results, I presented some alternative approaches. Especially the laminar case with changed inlet conditions using the `twoPhaseEulerFoam` solver showed some promising results. The vertical velocity profile was quite similar to the experiments. It also showed large spreading of the air, leading to a high plume radius. Further study should be done into this, but it clearly shows the potential of OpenFOAM to produce better results than the simulations presented in Chapter 4. Unfortunately I didn’t have time to further work with the alternative approaches. A lot of time was used just to get a good case setup that allowed for the different cases to converge.

One thing is clear, when the industry is moving into deeper water, they can no longer rely on the basic relation based on Fanneløp and Sjøen (1980). This relation assumes ideal gas behavior and that the gas is conserved in the plume. This is not the case for deeper water. Off course one can argue that they are conservative as long as they add a factor large enough to account for deep water and large plumes. Especially in consequence modeling and risk analysis this could be argued. But predicting unrealistically large clouds of flammable gas at the surface has its downsides. It could e.g. provide unnecessary costs to design and limit the activities of ships and the rig/platform. When a

blowout occurs, the intervention vessels are not allowed to work directly above the blowout. This complicates the job. If one could argue that there will not be flammable concentration of e.g. gas at the surface, this operation could be done a lot more effectively. The experimental database is very limited, but in June 2000 an experiment was executed in the Norwegian Sea at 840 meter water depths. Gas was released at a rate of 0.5kg/s, and none of the gas was observed at the surface. The echo sounders indicated that the methane gas disappeared at about 150 meter water depth, almost certainly because of dissolution into the ambient water (Johansen, 2003). This is a small release, but it still makes a good point. By applying the standard relation the expected diameter of boiling region would be around 180 meters, consisting of 100% methane gas. This will also be assumed to be directly above the release. If this is how this case would be solved in practice is uncertain. As mentioned, both the use of integral models and modification of the basic relation is used. Knowing that this relation is in use, we certainly cannot exclude the fact that this is how the fate of the released gas can be calculated in the industry. Integral models are the available tool today to tackle challenges like the mentioned release at 840 meters. The question is whether numerical simulation also can be an alternative for the future. To be able to compete with simpler integral models, much more research must be done. While I had problems setting up an ideal case in a 10 meter pool, an integral model like DeeBlow (Johansen, 2000) can take into account effects of cross-currents, non-ideal gas behavior, dissolution of gas, and formation and subsequent disintegration of hydrate. Further Fanneløp and Bettilini (2007) compared the integral model StarPlume against a number of available experiments with good results.

Based on the information gained through this thesis, the conclusion is that we must rely on integral models today and for the foreseeable future, especially for practical use in the industry. The immediate focus should be on lecturing the industry on the clear limitation inherent in the basic relation based on Fanneløp and Sjøen (1980). Using integral models for consequence modeling should be a requirement pushed by the safety authorities. That being said, research into numerical simulation of bubble plumes should also be prioritized. The alternative approaches in Chapter 5 showed some indications that it is possible to set up an ideal bubble plume using OpneFOAM. The challenge is to include non-ideal gas behavior, effects of cross currents, dissolution of gas and so forth. Also models that can include bubble breakage and coalescence should be considered. There is a clear lack of experimental data into deep set bubble plumes. Models based on numerical simulation can in the future help extending the available database. By verifying against experiments, a good model can be used to run a number of cases to build up a library of data to further support integral models. So it might not be directly used by the industry, but by different research institutes. In the next chapter I will discuss potential future work in relation to further development of numerical simulation of bubble plumes.

7. Future Work

By looking at the alternative approaches it is clear that OpenFOAM, and the available solvers, have potential of producing decent results when it comes to bubble plumes. By doing further research and modification of relevant solvers, it might be possible to get very good results. Some of the modifications that is possible is introducing the LES turbulence model and models for bubble breakage and coalescence. Some theoretical and practical work have been done in this regard. Dhotre et al. (2012) did a review of resent work into dispersed bubbly flow using large eddies simulation. This clearly shows that this turbulence model is available, and also the preferred turbulence model for resent research into bubble columns. In LES the large eddies are resolved directly on a numerical grid, while the small, unresolved eddies are modelled. This allows the flow to fluctuate and entrain the surrounding water. Acher et al. (2013) have validated a method to take bubble breakage and coalescence into account. Giving the bubbles the ability to break up and melt together is important to get the correct spreading of the plume. Fanneløp and Sjøen (1980) observed as big as 1cm bubbles by the inlet, before they stated breaking up into smaller bubbles. Most of the works mentioned are done in relation to the chemical and nuclear industry. More study should be done into bubble plumes from subsea releases of gas. This includes developing models for larger water depths. These models should be considered including sub-models for potential dissolution of gas, water currents, hydrate formation and other important phenomenon's that comes into play in deep water. Models like that could be used for engineering purposes in risk analysis and consequence modeling, or for validation of simpler models.

8. References

ACHER, T., DEMS, P., LENZ, S., GOBERT, C. & POLIFKE, W. Validation of a quatrature method of moments for polydisperse flow in bubble columns including poly-celerity, brakup and coalescence. 8th International Confrence on Multiphase Flow ICMF, May 26-31 2013 Jeju, Korea.

ANGLART, H., ANDERSSON, S., PODOWSKI, M. Z. & KURUL, N. 1993. An analysis of multidimensional void distribution in two-phase flows. *Proceedings og the Sixth International Topical Meeting on Nuclear Reactor Thermalhydraulics (NURETH 6), Grenoble, France, October 5-8*.

BETTELINI, M. & FANNELØP, T. K. 1993. Underwater plume from an instantaneously started source. *Applied Ocean Research,* 15**,** 195-206.

DEEN, N. G. 2001. An experimental and computational study of fluid dynamics in gas-liquid chemical reactors. *Ph.D thesis, Aalborg University, Esbjerg, Denmark*.

DHOTRE, M. T., DEEN, N. G., NICENO, B., KHAN, Z. & JOSHI, J. B. 2012. Large Eddy Simulation for Dispersed Bubble Flows: A Review. *International Journal of Chemical Engineering*.

DREW, D. A. 1971. Averaged foeld equations for two-phase media. *Studies in Apllied Mathemetacis* 133-165.

DREW, D. A. 1983. Continuum modeling of two-phase flows. . *In:* MEYER, R. (ed.) *Theory of dispersed multiphase flow.* Acedemic Press.

DREW, D. A. & LAHEY, R. T. 1987. The virtual mass and lift force on a sphere in totating and straining invidcid flow. *International Journal of Multiphase Flow,* 18**,** 371-395.

ENWALD, H., PEIRANO, E. & ALMSTEDT, A. E. 1996. Eularian two-phase flow theory applied to fluidization. *International Journal of Multiphase Flow,* 22**,** 21-66.

FANNELØP, T. K. 1994. Fluid Mechanics for Industrial Safety and Environmental Protection. *Industrial Safety Series, Elsevier*.

FANNELØP, T. K. & BETTILINI, M. 2007. Very Large Deep-Set Bubble Plumes From Broken Gas Pipelines. *Petroleumstilsynet (Ptil)***,** 70.

FANNELØP, T. K. & SJØEN, K. 1980. Hydrodynamics of underwater blowouts. *AIIA Paper 80-0219*.

FRIEDL, M. J. & FANNELØP, T. K. 2000. Bubble plumes and their interaction with the water surface. *Applied Ocean Research,* 22**,** 119-128.

HILL, D. P. 1998. The computer simulation of dispersed two-phase flow. *PhD. thesis, Imperial College of Science, Technology and Medicine, London*.

HJERTAGER, B. H. 2001. Computational Analysis of Fluid Flow Processes. *University of Stavanger*.

HJERTAGER, B. H. 2009. Lecture notes in OpenFoam. *University of Stavanger*.

ISHII, M. & MISHIMA, K. 1984. Two fluid model and hydrodynamic constitutive relations. *Nuclear Engineering Design,* 82**,** 107-126.

ISHII, M. & ZUBER, N. 1979. Drag coefficient and relativevelocity in bubbly, droplet or particulate flows. *I.I.Ch.E. Journal,* 25**,** 843.

JOHANSEN, Ø. 2003. Development and verification of deep-water blowout models. *Marine Pollution Bulletin,* 47**,** 360-368.

JOHANSEN, Ø. 2000. DeepBlow-a Lagrangian Plume Model for Deep Water Blowouts. *Spill Science & Technology Bulletin,* 6**,** 103-111.

LAUNDER, B. E. & SPALDING, B. 1972. Mathematical Models of Turbulence. New York: Acedemic Press.

LEVICH, V. G. 1962. Physicochemical Hydrodynamics. *Prentice-Hall*.

LOPEZ DE BERTODANO, M. 1992. Turbulent bubbly two-phase flow in a triangular duct. *Ph.D thesis, Rensselaer Polytechnic Institute, Troy, New York*.

MILGRAM, J. H. 1983. Mean flow in round bubble plumes. *J. Fluid Mech.,* 133**,** 345-376.

OLIVEIRA, J. P. & ISSA, R. I. 2003. Numerical aspects of an algorithm for Eulerian simulation of two-phase flows. *International Journal for Numerical Methods in Fluids,* 43**,** 1177-1198.

OPENFOAM 2011. The Open Source CFD Toolbox, User Guide. *OpenFOAM*.

OPENFOANWIKI.NET. 2012. *BubbleFoam* [Online]. Available: http://openfoamwiki.net/index.php/BubbleFoam [Accessed January 2013].
PFLEGER, D. & BECKER, S. 2001. Modeling and simulation of the dynamic flow behavior in a bubble column. *Chemical Engineering Science,* 56**,** 1737-1747.
RUSCHE, H. 2002. Computational fluid dynamics of dispersed two-phase flows at high phase fractions. *Ph.D thesis, Imperial College of Science, Technology and Medicine, London*.
SHENG, Y. Y. & IRONS, G. A. 1995. The impact of bubble dynamics on the flow in plumes of ladle water models. *Metallurgical Transactions* B 26**,** 625-635.
SIMIANO, M. 2005. Experimental investigation of large-scale three dimensional bubble plume dynamics. *Diss. ETH. No. 16220, Swis Federal Instiitute of Technology, Zurich*.
SINTEF & SCANDPOWER 2003. *Handbook for Fire Calculations and Fire Risk Assessment in the Process Industry*.
SMITH, B. L. & DHOTRE, M. T. 2007. CFD simulation of large-scale bubble plumes: Comparison against experiments. *Chemical Engineering Science,* 62**,** 6615-6630.
TABIB, M. V., ROY, S. A. & JOSHI, J. B. 2008. CFD simulation of bubble column - an analysis of interphase forces and turbulence models. *Chemical Engineering Science,* 139**,** 589-614.
TVEIT, O. J. 2006. Risiko knyttet til gassutslipp under vann.
VEERSTEG, H. K. & MALALASEKERA, W. 2007. *An Introduction to Computational Fluid Dynamics,* Essex, England, Pearson Education Limited.
WELLER, H. G. 2005. Derivation, modelling and solution of the conditionally averaged two-phase flow equations. *Technical report, OpenCFD Ltd*.
YAPA, P. D. & ZENG, L. 1997. Modelling oil and gas releases from deep water: A review. *Spill Science & Technology Bulletin,* 4**,** 189-198.
ZHANG, D., DEEN, N. G. & KUIPERS, J. A. M. 2006. Numerical simulation of the dynamic flow behaviour in a bubble column: a study of the closures for turbulence and interface forces. *Chemical Engineering Science,* 61**,** 7593-7608.

Appendix A

A.1 Boundary Conditions

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 0 0 0 0 0 0];

internalField   nonuniform List<scalar>
8200
(
0
0
0
0
0
0
0
0
0
0
.
.
.
.
+
+
.
.
.
1
1
```

```
1
1
)
;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 1;
    }
    outlet
    {
        type            zeroGradient;
    }
    lowerWalls
    {
        type            zeroGradient;
    }
    sideWalls
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
}


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

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      Ua;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0 0 0.253);
    }

    outlet
    {
        type            zeroGradient;
    }
    fixedWalls
    {
        type            fixedValue;
      value       uniform (0 0 0);
    }
    defaultFaces
    {
        type            empty;
    }
}

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

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      Ub;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    outlet
    {
        type            fixedValue;
        value            uniform (0 0 0);
    }
    fixedWalls
    {
        type            fixedValue;
      value      uniform (0 0 0);
    }
    defaultFaces
    {
        type            empty;
    }
}

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

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 1e-8;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 1e-8;
    }

    outlet
    {
        type            zeroGradient;
    }

    fixedWalls
    {
         type           zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}

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

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

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

internalField   uniform 1e-8;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 1e-8;
    }

    outlet
    {
        type            zeroGradient;
    }
    fixedWalls
    {
         type           zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}

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

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

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

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    fixedWalls
    {
      type       zeroGradient;
      value           uniform 0;
    }
    defaultFaces
    {
        type            empty;
    }
}

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

Appendix B

B.1 Transport Properties

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

// Air, alpha=1
nua              nua  [ 0 2 -1 0 0 0 0 ] 1.6e-05;
rhoa             rhoa [ 1 -3 0 0 0 0 0 ] 1;
da               da   [ 0 1 0 0 0 0 0 ]  3e-3;

// Water, alpha=0
nub              nub  [ 0 2 -1 0 0 0 0 ] 1e-06;
rhob             rhob [ 1 -3 0 0 0 0 0 ] 1000;
db               db   [ 0 1 0 0 0 0 0 ]  1e-4;

Cvm             Cvm [ 0 0 0 0 0 0 0 ] 0.5;

Cl              Cl [ 0 0 0 0 0 0 0 ] 0.5;

Ct              Ct [ 0 0 0 0 0 0 0 ] 1;


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

B.2 Turbulence Properties

```
/*---------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

RASModel        kEpsilon;

turbulence      on;

printCoeffs     on;


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

B.3 Gravitational Acceleration

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 1 -2 0 0 0 0];
value           ( 0 0 -9.81 );


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

B.4 BlocMeshDict-3DMesh

```
/*--------------------------------*- C++ -*----------------------------
------* \
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.7.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

convertToMeters 1;

vertices
(

 (0 0 0)
 (2.45 0 0)
 (2.45 2.45 0)
 (0 2.45 0)
 (0 2.55 0)
 (2.45 2.55 0)
 (0 5 0)
 (2.45 5 0)
 (2.55 0 0)
 (2.55 2.45 0)
 (5 0 0)
 (5 2.45 0)
 (2.55 2.55 0)
 (5 2.55 0)
 (2.55 5 0)
 (5 5 0)

 (0 0 12)
 (2.45 0 12)
 (2.45 2.45 12)
 (0 2.45 12)
 (0 2.55 12)
 (2.45 2.55 12)
 (0 5 12)
 (2.45 5 12)
 (2.55 0 12)
```

```
 (2.55 2.45 12)
 (5 0 12)
 (5 2.45 12)
 (2.55 2.55 12)
 (5 2.55 12)
 (2.55 5 12)
 (5 5 12)


);

blocks
(
 hex (0 1 2 3 16 17 18 19)     (20 20 100) simpleGrading (1 1 1)
 hex (3 2 5 4 19 18 21 20)     (20  2 100) simpleGrading (1 1 1)
 hex (4 5 7 6 20 21 23 22)     (20 20 100) simpleGrading (1 1 1)
 hex (1 8 9 2 17 24 25 18)     (2 20 100) simpleGrading (1 1 1)
 hex (2 9 12 5 18 25 28 21)    (2 2 100) simpleGrading (1 1 1)
 hex (5 12 14 7 21 28 30 23)   (2 20 100) simpleGrading (1 1 1)
 hex (8 10 11 9 24 26 27 25)   (20 20 100) simpleGrading (1 1 1)
 hex (9 11 13 12 25 27 29 28)  (20  2 100) simpleGrading (1 1 1)
 hex (12 13 15 14 28 29 31 30) (20 20 100) simpleGrading (1 1 1)
 );

edges
(
);

patches
(
 patch inlet (
            (2 5 12 9)
            )

 patch outlet (
             (16 19 18 17)
             (19 20 21 18)
             (20 22 23 21)
             (17 18 25 24)
             (18 21 28 25)
             (21 23 30 28)
             (24 25 27 26)
             (25 28 29 27)
             (28 30 31 29)

                  )

   wall fixedWalls
    (
     (0 16 19 3)
     (3 19 20 4)
     (4 20 22 6)
     (6 22 23 7)
     (7 23 30 14)
     (14 30 31 15)
     (15 31 29 13)
     (13 29 27 11)
     (11 27 26 10)
```

```
      (10 26 24 8)
      (8 24 17 1)
      (1 17 16 0)
      (0 3 2 1)
      (3 4 5 2)
      (4 6 7 5)
      (5 7 14 12)
      (12 14 15 13)
      (9 12 13 11)
      (8 9 11 10)
      (1 2 9 8)
      )


);

mergePatchPairs
(
);

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

B.5 BlockMeshDict-2DMesh

```
/*--------------------------------------------------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  1.6
|
|   \\  /    A nd           | Web:      http://www.openfoam.org
|
|    \\/     M anipulation  |
|
\*--------------------------------------------------------------------
------*/

FoamFile
{
    version         2.0;
    format          ascii;
    class           dictionary;
    object          blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

convertToMeters 1;

vertices
(
    (0 0 0)    //0
    (9.95 0 0)    //1
    (10.05 0 0)    //2
    (20 0 0)    //3
    (20 12 0)   //4
    (10.05 12 0)   //5
    (9.95 12 0)   //6
    (0 12 0)   //7
    (0 0 0.1) //8
    (9.95 0 0.1) //9
    (10.05 0 0.1) //10
    (20 0 0.1) //11
    (20 12 0.1)//12
    (10.05 12 0.1)//13
    (9.95 12 0.1)//14
    (0 12 0.1)//15
);

blocks
(
    hex (0 1 6 7 8 9 14 15) (80 100 1) simpleGrading (1 1 1)  //0
    hex (1 2 5 6 9 10 13 14) (2 100 1) smpleGrading (1 1 1) //1
    hex (2 3 4 5 10 11 12 13) (80 100 1) simpleGrading (1 1 1)  //2
);
```

```
edges
(
);

boundary
(
    inlet
    {
      type patch;
      faces
      (
            (1 9 10 2)
      );
    }
    outlet
    {
      type patch;
      faces
      (
              (7 6 14 15)
            (6 5 13 14)
            (5 4 12 13)
      );
    }
    lowerWalls
    {
      type wall;
      faces
      (
            (0 8 9 1)
            (2 10 11 3)
      );
    }
    sideWalls
    {
      type wall;
      faces
      (
              (0 7 15 8)
            (3 11 12 4)
      );
    }
    frontAndBack
    {
      type empty;
      faces
      (
            (0 7 6 1)
            (1 6 5 2)
            (2 5 4 3)
            (8 9 14 15)
            (9 10 13 14)
            (10 11 12 13)
      );
    }
);
mergePatchPairs
(
```

```
);

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

Appendix C

C.1 controlDict

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

application     bubbleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         50;

deltaT          0.002;

writeControl    runTime;

writeInterval   1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;
```

```
timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.15;

maxDeltaT       1;

functions
{
   fieldAverage1
   {
     type            fieldAverage;
    functionObjectLibs ( "libfieldFunctionObjects.so" );
    outputControl   outputTime;
    fields
   (
         Ua
         {
              mean        on;
               prime2Mean  off;
               base        time;
          }

          Ub
          {
               mean        on;
               prime2Mean  off;
               base        time;
          }

          alpha
           {
               mean        on;
                prime2Mean  off;
                base        time;
           }

          p
           {
                mean        on;
                prime2Mean  off;
                base        time;
          }
     );
    }
testSample
          {
            type sets;
            functionObjectLibs ("libsampling.so");
            setFormat xmgr;
            interpolationScheme cell;
            outputControl outputTime;

            fields
            (
```

```
            UaMean UbMean alphaMean
            );
           sets
           (
              line1
              {
                type midPoint;
                axis z;

                start (2.5 2.5 0);
                end (2.5 2.5 10);
               }
              line2
              {
                type midPoint;
                axis x;

                start (0 2.5 9);
                end (5 2.5 9);
              }
              line3
              {
                type midPoint;
                axis x;
                start (0 2.5 6);
                  end (5 2.5 6);
              }
              line4
              {
                type midPoint;
                axis x;
                start (0 2.5 3);
                end (5 2.5 3);
              }
                  );
         }


}

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

C.2 fvSchemes

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

/*
    div(phia,Ua)    Gauss limitedLinearV 1;
    div(phib,Ub)    Gauss limitedLinearV 1;
    div(phib,k)     Gauss limitedLinear 1;
    div(phib,epsilon) Gauss limitedLinear 1;
    div(phi,alpha)  Gauss limitedLinear01 1;
    div(phir,alpha) Gauss limitedLinear01 1;
    */
divSchemes
{
    default         none;
    div(phia,Ua)    Gauss upwind;
    div(phib,Ub)    Gauss upwind;
    div(phib,k)     Gauss upwind;
    div(phib,epsilon) Gauss upwind;
    div(phi,alpha)  Gauss upwind;
    div((-nuEffa*T(grad(Ua)))) Gauss linear;
    div((-nuEffb*T(grad(Ub)))) Gauss linear;
}
```

```
laplacianSchemes
{
    default         none;
    laplacian(nuEffa,Ua) Gauss linear corrected;
    laplacian(nuEffb,Ub) Gauss linear corrected;
    laplacian((rho*(1|A(U))),p) Gauss linear corrected;
    laplacian((alphak*nuEffb),k) Gauss linear corrected;
    laplacian((alphaEps*nuEffb),epsilon) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


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

C.3 fvSolution

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.1
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-08;
        relTol          0.1;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-08;
        relTol          0;
    }

    "(k|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
```

```
    }

    "(k|epsilon)Final"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }

    alpha
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-10;
        relTol          0.1;
    }

    alphaFinal
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-10;
        relTol          0;
    }
}

PIMPLE
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    nAlphaCorr      2;
    correctAlpha    yes;
    pRefCell        0;
    pRefValue       0;
}


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

C.4 setFieldsDict

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.1.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

defaultFieldValues
(
    volScalarFieldValue alpha 1
);
regions
(
    boxToCell
    {
        box (0 0 0) (5 5 10);
        fieldValues
        (
            volScalarFieldValue alpha 0
        );
    }
);

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

Appendix F

F.1 Content of Encolosed CD

The enclosed CD includes a PDF version of the thesis, as well as all the case files. The necessary sub-directories 0, constant and system are included for the 2D cases, 3D cases, Parameter Study and Alternative Approaches.