NTNU – Trondheim
Norwegian University of
Science and Technology

Two-Phase pipeflow simulations with OpenFoam

Nerea Herreras
Omagogeascoa
Jon Izarra Labeaga

Master's Thesis
Submission date: June 2013
Supervisor: Reidar Kristoffersen, EPT

Norwegian University of Science and Technology
Department of Energy and Process Engineering

NTNU

Norwegian University
of Science and Technology

Department of Energy
and Process Engineering

EPT-M-2013-52/58

MASTER THESIS

for

Stud.techn. Nerea Herreras and
Stud.techn. Jon Izarra

Spring 2013

Two-Phase Pipe flow Simulations with OpenFOAM

Tofase rørstrømning simuleringer med OpenFoam

Background and objective

OpenFOAM is an open-source CFD package which is a free alternative to expensive commercial CFD software. OpenFOAM has developed substantially during the last years, and it is interesting to investigate the possibilities this software has to engineering problems.

The following tasks are to be considered:

1. Simulate a free surface flow along an inclined plate. Compare with analytical solutions and empirical/experimental results.

2. Simulate two-phase flow in a horizontal pipe. Compare with experimental data.

3. Simulate two-phase flow in an inclined pipe.

-- ” --

Within 14 days of receiving the written text on the master thesis, the candidate shall submit a research plan for his project to the department.

When the thesis is evaluated, emphasis is put on processing of the results, and that they are presented in tabular and/or graphic form in a clear manner, and that they are analyzed carefully.

The thesis should be formulated as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents etc. During the preparation of the text, the candidate should make an effort to produce a well-structured and easily readable report. In order to ease the evaluation of the thesis, it is important that the cross-references are correct. In the making of the report, strong emphasis should be placed on both a thorough discussion of the results and an orderly presentation.

The candidate is requested to initiate and keep close contact with his/her academic supervisor(s) throughout the working period. The candidate must follow the rules and regulations of NTNU as well as passive directions given by the Department of Energy and Process Engineering.

Risk assessment of the candidate's work shall be carried out according to the department's procedures. The risk assessment must be documented and included as part of the final report. Events related to the candidate's work adversely affecting the health, safety or security, must be documented and included as part of the final report. If the documentation on risk assessment represents a large number of pages, the full version is to be submitted electronically to the supervisor and an excerpt is included in the report.

Pursuant to "Regulations concerning the supplementary provisions to the technology study program/Master of Science" at NTNU §20, the Department reserves the permission to utilize all the results and data for teaching and research purposes as well as in future publications.

The final report is to be submitted digitally in DAIM. An executive summary of the thesis including title, student's name, supervisor's name, year, department name, and NTNU's logo and name, shall be submitted to the department as a separate pdf file. Based on an agreement with the supervisor, the final report and other material and documents may be given to the supervisor in digital format.

□ Work to be done in lab (Water power lab, Fluids engineering lab, Thermal engineering lab)
□ Field work

Department of Energy and Process Engineering, 14. February 2013

Olav Boll

Olav Bolland
Department Head

Reidar Kristoffersen

Reidar Kristoffersen
Academic Supervisor

Research Advisors:
Aliresa Ashrafian, Weatherford.

PREFACE

This thesis has been written during our second semester in the NTNU and it is the last of our engineering study program in the Engineering school of Bilbao. It is also a continuation of a pre-project started the preceding semester.

The main aim of this thesis is to develop two-phase simulations using the CFD package OpenFOAM. Both open and closed channel flows are simulated in horizontal and inclined positions. During the specialization project we used how the multiphase cases in OpenFOAM and this served as a starting point for the current work. The problems that aroused in the pre-project were studied not to make the same mistakes again. During the work, implementation of cyclic boundary conditions was considered but due to some problem with the implementation in our cases, the project was finally abandoned.

This text is aimed at prospective students who are interested in working with OpenFOAM and therefore explains step by step how to work with multiphase cases and the reasons for the choices made when it comes to numerical schemes, turbulence models and so on. For this, open and closed channel flow are studied and analyzed and also the necessary theoretical background is given. Finally, it presents and discusses the results that have been obtained in various test cases and simulation and suggests pathways for further development.

Trondheim, 10th June, 2013

Nerea Herreras and Jon Izarra

ABSTRACT

The main purpose of this thesis is to develop two-phase simulations using OpenFOAM. Two different situations are studied: open and closed channel flow. Different parameters are changed in each case to obtain different results, such as the inclination of the channel and the values of the velocity inlets for each phase.

When dealing with the open-channel flow different inclinations are simulated and the influence of the Froude number is analyzed. The results obtained are compared with the analytical solution obtained from the Navier-Stokes equation and also with experimental results from studies done for open-channel flows.

For the closed channel flow, which is studied in horizontal, vertical and inclined position, different inlet velocities are given for each phase in order to create the different flow patterns characteristic of the multiphase flow and the results obtained are compared with experiments displayed in Tailter Dukler map that gives the transition between different flow regimes.

Finally, some possible future work is presented so that the person that wants to take a deeper study of the cases has some ideas to improve the simulations, ideas that could not be carried out in this study due to the lack of time and computational power.

All the files used in the software can be seen in the Appendix part of the thesis so that if the reader wants to test the simulations or just apply the conditions used in this study for a similar case, can use them as a base. Moreover, a CD is enclosed with the cases run in the thesis so that they can be directly run on a computer for testing or changing for further study.

SAMMENDRAG

Meningen med denne oppgaven er å utvikle 2-fase-simuleringer ved å bruke OpenFOAM. To situasjoner blir studert; åpen og lukket kanalstrømning. Forskjellige parametre blir variert i hvert tilfelle for å oppnå forskjellige resultater, som f.eks. helningsvinkel og inngangshastigheter for hver fase.

I tilfeller med åpne kanaler brukes forskjellige helningsvinkler, og effekten uttrykt ved Froude-tallet blir analysert. Testresultatene blir så sammenlignet med den analytiske løsningen av Navier-Stokes ligningen og eksperimentelle resultater.

I tilfeller med lukket kanalstrømning, hvor kanalen ligger henholdsvis horisontalt, vertikalt og skrått, brukes forskjellige inngangshastigheter til hver fase for å skape forskjellige strømningsmønstre karakteristiske for flerfasestrømning, og resultatene blir sammenlignet med andre eksperimentelle resultater hentet fra Tailter-Dukler kart som gir transisjonen mellom forskjellige strømningsregimer.

Til slutt presenteres fremtidige forslag til arbeid på området som inspirasjon til personer som ønsker å gjøre en dypere studie av problemstillingen og forbedre simuleringene.

Alle filene som er brukt finnes i Appendix-delen av oppgaven og kan brukes som utgangspunkt hvis leseren ønsker å teste simuleringene eller betingelsene fra dette studiet i lignende tilfeller. I tillegg er det lagt ved en CD med alle resultatfiler utført i oppgaven som kan kjøres direkte på en datamaskin for testing og videre-utvikling.

ACKNOWLEDGEMENTS

The thesis as had its ups and downs and therefore, we would like to thank our supervisor Reidar Kristoffersen for his valuable suggestions and our weekly meetings where different options and possibilities where discussed and where he helped us solving our problems.

Furthermore, we would like to thank our international friends in Trondheim that have always supported us and helped us carry on when we were feeling a bit blocked or under the weather.

The thesis was also carried in cooperation with Weatherford Petroleum AS and we will like to thank them for their interest in our study.

TABLE OF CONTENTS

APPENDIX

NOMENCLATURE

LATIN SYMBOLS

A	Cross sectional area	$[m^2]$
B	Width normal to the paper	$[m]$
$C, C_a, C_b,$ $C_1, C_2, C_\mu,$ $\sigma_k, \sigma_\varepsilon, C_{1\varepsilon}$	Constants	
c	Wave velocity	$[m/s]$
Co	Courant number	
D	Diameter of the pipe	$[m]$
Dm	Hydraulic mean depth	$[m]$
E	Specific energy	$[m]$
e_{ij}	Rate of deformation	$[s*m]^{-1/2}$
$E_{\min}$	Minimum Specific Energy	$[m]$
Fr	Froude number	
g	Gravitational acceleration	$[m/s^2]$
h_f	Friction losses	
K	Mean kinetic energy	$[m^2/s^2]$
k	Turbulent kinetic energy	$[m^2/s^2]$
L	Significant linear dimension	$[m]$
ℓ	Length scale	$[m]$
L_e	Entrance length	$[m]$
m	mass	$[kg]$
Ma	Match number	
P	Wetted perimeter	$[m]$
p	Pressure	$[Pa]$
p_0	Total pressure	$[Pa]$
p_d	Dynamic pressure	$[Pa]$
Q	Liquid flow rate	$[m^3/s]$
R	Hydraulic radius	$[m]$
Re	Reynolds number	
Re_{crit}	Critical Reynolds number	
t	Time	$[s]$
u	Velocity component	$[m/s]$

$\boldsymbol{U}$	Velocity vector	$[m/s]$
U_{drift}	Drift velocity between phases	$[m/s]$
U_g	Velocity of the gas	$[m/s]$
U_l	Velocity of the liquid	$[m/s]$
U_{gs}	Superficial gas velocity	$[m/s]$
U_{ls}	Superficial liquid velocity	$[m/s]$
U_m	Velocity of the mixture	$[m/s]$
V	Velocity of the fluid	$[m/s]$
$\vec{v}$	Velocity in x direction	$[m/s]$
V_g	Volume occupied by the gas	$[m^3]$
V_l	Volume occupied by the liquid	$[m^3]$
W	Weight term	$[N]$
$\boldsymbol{X}$	Position vector	$[m]$
Z	Channel water depth	$[m]$
z	Elevation of the free surface	$[m]$

GREEK SYMBOLS

α	Volume/void fraction	
ε	Rate of dissipation of turbulent kinetic energy	$[m/s^2]$
μ	Dynamic viscosity	$[Pa*s]$
μ_{air}	Dynamic viscosity of air	$[Pa*s]$
μ_{eff}	Effective dynamic viscosity	$[Pa*s]$
μ_t	Turbulent dynamic or eddy viscosity	$[Pa*s]$
μ_{water}	Dynamic viscosity of water	$[Pa*s]$
ρ	Density of the fluid	$[kg/m^3]$
ρ_{air}	Density of air	$[kg/m^3]$
ρ_g	Density of gas	$[kg/m^3]$
ρ_{water}	Density of water	$[kg/m^3]$
θ	Inclination of the channel respect the horizontal	
ϕ	Flux	$[m^3/s]$
γ	Specific gravity	$[N/m^3]$
ν	Kinematic viscosity	$[m^2/s]$
ϑ	Velocity scale	$[m/s]$

τ_w	Shear stress	$[N/m^2]$

ABBREVIATIONS

CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
EGL	Energy Grade Line
LES	Large Eddy Simulation
RAS	Reynolds Averaged Simulation
VOF	Volume Of Fluid

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Previous work

During the last semester the CFD tool OpenFOAM was introduced in the course Engineering Fluid Mechanics, Specialization course, and it was decided to start a project using this software to see how it worked and so, to analyze fluid problems worthy of interest. Once knowledge was acquired in multiphase flow in the same course, work began on the simulation of a closed channel line with two phases and to analyze the dependence of the flow pattern when changing speeds to the input phases. Although the results were not as expected, they showed that changes in the rates of entry, truly could create different flow patterns, and therefore, even if there were some problems in establishing both the boundary and the initial conditions, OpenFOAM gave different results according to combination of speeds.

Regarding the numerical analysis and the solution to the system of equations, these topics where also covered in the course Computational Heat and Fluid Flow. In these courses some knowledge was gained in classification and discretization of equations for different fluid flows, finite volume methods, numerical solutions to the equations, solution of algebraic system of equations, turbulent flows and so on.

1.2 Motivation

The first and most important reason why OpenFOAM is used is that it is an open source software. Commercial CFD tools are very expensive and therefore more inaccessible. In an academic setting, it is especially important not to have to pay for use, as it is not certain that the school agrees to pay such large sums of money for academic study. Apart from that, OpenFOAM has a wide range of use among different areas of engineering and science and it is written in C++ language, which is a modern and popular programming language. OpenFOAM can solve problems involving complex fluid flows with chemical reactions, heat transfer and turbulence, and can also work with solid dynamics and electromagnetics. OpenFOAM includes pre-processing and post-processing tools. It also offers the possibility to work in parallel, so that the user can take full advantage of the computer when working.

In order to use OpenFOAM the operating system Linux is recommended, so when working with the tool the user gets also acquainted with the operating system.

At the beginning of last semester Weatherford showed much interest in OpenFOAM and wanted to know what it is able to do. This is why the project is based on multiphase flow, since Weatherford is an oil and gas company. The study of flow in pipes is of vital importance for these companies, both horizontal, inclined and vertical.

As a verification test, the open-channel flow is also studied along with the influence of the Froude number, and the results are discussed with experimental and analytical results.

1.3 Thesis objective

The objective of the thesis is to develop a two-phase simulation in OpenFOAM along different sets, like an inclined plate or different closed channel positions, which can be used to understand better the way OpenFOAM works with multiphase flow. It is also an objective of this thesis to analyze the possibilities that OpenFOAM has to solve engineering problems. As explained above, different cases are examined, where the boundary conditions can also change in order to analyze the behavior of the fluid under different conditions, as well as the geometry. At the beginning, some simple cases were run, laminar cases with the same velocities for both phases. Once these simulations are established more complex cases were developed in order to achieve the expected results.

For the open-channel flow, at the beginning some more study has to be made due to a special boundary and its conditions, the atmosphere. Therefore, the tutorial case damBreak is going to be analyzed and the data for the atmosphere patch will be taken from here. Once the simulations are running and in order to compare the results with the analytical solution, a matlab code will be created to display the analytical distribution of the velocity and compare it with the velocity profile obtained in paraview.

When it comes to the closed channel flow results, the set up from the previous project is used, and using this base, the work is developed. Different velocity combinations are used to create the different patterns and then the results are compared with the ones obtained experimentally using maps that show the different combinations.

At some point, cyclic boundary condition wanted to be implemented in the simulations to analyze its behavior but after several tries, and some weeks not having any answer this idea was discarded.

2. LITERATURE STUDY

2.1 CFD

CFD, which stands for Computational Fluid Dynamics, is the system analysis consisting of fluid flow, heat transfer and chemical reactions with the help of a computer simulation [11]. Thanks to CFD, many engineering problems are covered, such as aerodynamics, gas turbines, turbo machinery, multiphase modeling, ship hydrodynamics and so on.

Many of the engineering problems shown above request some experimental results which are often impossible to realize or economically non-viable. These are the most important reasons why CFD is of such important when dealing with fluid flow problems, it is cheaper and less time consuming. When using CFD, some parameters that are difficult to measure in real experiments can be considered in the study. In recent years, it has become of common use both in the industry and the academic world, in order to work with fluid flow problems, due to its capability to show results faster and more accurately. However, validation and verification of the results is needed because usually CFD gives results with very nice displays that look very attractive but that can be wrong. Through validation and verification it is ensured that the results achieved are at least reliable.

CFD codes are based on numerical algorithms and consist of three main elements: pre-processing, processing and post-processing. In the pre-processing step the inputs are given to the CFD software for the ongoing problem, and this is made by easy-to-use interfaces. These inputs are usually the definition of the geometry, mesh generation (division of the domain into a number of cells), definition of fluid properties, the turbulence properties and of course the initial and boundary conditions. The accuracy of a CFD solution is affected by the number of cells in the grid, so the bigger the number of cells, the more accurate will the solution be. In the processing or solver part, the unknown flow variables are approximated by means of simple functions, then these approximations are introduced into the governing equations and they are discretized later on. Finally the algebraic equations are solved.

OpenFOAM uses the finite volume method streamline to define the way in which the flow variables are approximated and the discretization processes. The finite volume method was originally developed as a special finite difference formulation [11], and it is used in most of the commercial CFD packages. The numerical algorithm follows these steps:

- Integration of the governing equations over all the control volumes of the domain.
- Conversion of the integral equations into a system of algebraic equations by means of discretization.
- Solution of the algebraic equations.

Finally, in the post-processing step the data created from the previous step is analyzed. In the post-processing the geometry, the grid, different vector plots, contour plots or surface plots can be visualized.

2.2 Froude Number

The Froude number represents the ratio of inertia to gravitational forces and it is named after the naval architect William Froude [6]. It is generally presented as:

$$Fr = \frac{V}{\sqrt{gL}} \tag{2.1}$$

Here L represents the most significant dimension for the flow under study and V is the velocity of the fluid.

The Froude number is very useful when analyzing open-channel flow, due to the importance of both gravitational and inertia forces.

The Froude number is also known to be the ratio between the flow speed and the wave speed [4]. The pressure in an open-channel is constant along the free surface, so if the flow is disturbed, a surface wave will appear, and not a pressure wave as shown in the figure 2.1.

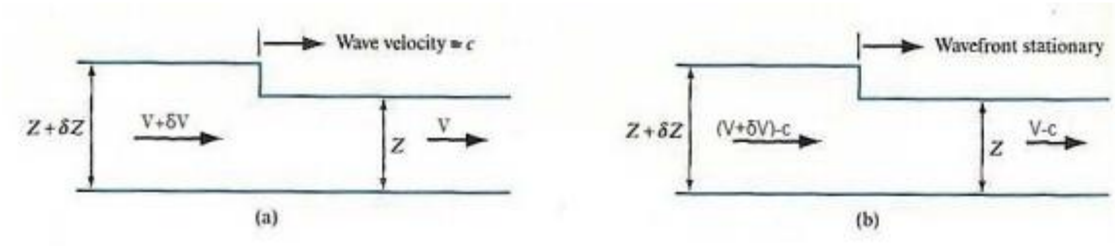


Figure 2.1. Surface wave. a) Unsteady flow observed from a stationary point of view. b) steady flow observed from a moving point of view. [4]

In order to derive the wave velocity the conservation of mass and Newton´s second law are used. Conservation of mass gives

$$\rho B\left(Z+\delta Z\right)\left(V+\delta V-c\right)=\rho BZ(V-c) \tag{2.2}$$

Here the *c* represents the wave velocity and V is the flow velocity. The B is the width normal to the paper plane.
The equation above states that the mass flow rate on the left hand side must equal the on the right hand side.
If the density is considered as constant, it can be cancelled:

$$Z\delta V+V\delta Z+\delta Z\delta V-c\delta Z=0 \tag{2.3}$$

Rearranging

$$(c-V)\delta Z=(Z+\delta Z)\delta V \tag{2.4}$$

If δZ increases, this is increasing the wave a hydrostatic force appears and it causes a higher wave velocity. If Newton´s second law is applied then,

$$\rho g\delta ZBZ=\rho BZ(V-c)(-\delta V) \tag{2.5}$$

Here the hydrostatic force must equal the mass flow rate times the change in velocity. Rearranging

$$\delta V=\frac{g\delta Z}{c-V} \tag{2.6}$$

Substituting the equation above in the equation (2.5)

$$(c-V)\delta Z=(Z+\delta Z)\frac{g\delta Z}{c-V} \tag{2.7}$$

Rearranging

$$(c-V)^2=(Z+\delta Z)g \tag{2.8}$$

If δZ is negligible, this means that the wave height is small.

$$(c-V)^2=gZ \tag{2.9}$$

Then the relative velocity of the wave respect to the fluid is held

$$c-V=\sqrt{gZ} \tag{2.10}$$

From the stationary point of view if the wave goes downstream its velocity is $\sqrt{gZ}+V$. If it goes upstream it is $\sqrt{gZ}-V$ instead.

If the velocity of the fluid is bigger than the wave velocity $(V>\sqrt{gZ})$, then it is impossible for the wave to flow in the upstream direction.

However, if the wave velocity is bigger than the fluid velocity $\left(V > \sqrt{gZ}\right)$, the wave can flow in both, upstream and downstream directions.

So finally, the Froude number is the ratio of the fluid velocity and the wave velocity

$$Fr = \frac{V}{c - V} \tag{2.11}$$

2.3 Open-channel flow

An open channel flow denotes flow with a free surface touching the atmosphere [12], which can be artificial (flumes, spillways, canals, weirs drainage ditches, culvers) or natural (streams, rivers, estuaries, floodplains). The closed ducts have no free surface because they are full of fluid, and they are driven by pressure forces, whereas in an open channel flow the only one is the gravity force. The basic force balance is between gravity and friction. The following information is taken from [17].

They are of special importance in civil and environmental engineering. The flow rates and water depths need to be predicted from the canal geometry. The common fluid in this kind of problems is water and the size of the channels is usually big and with turbulent flow.

The free surface which is usually at atmospheric pressure, is treated under constant pressure, which makes it the equivalent to the HGL or Hydraulic Grade Line. The depth profile changes with conditions and is computed as part of the problem, especially in unsteady problems involving wave motions.

An open channel is composed of two sides and a bottom and it satisfies the no-slip condition.

The measured velocity in an open channel will always vary across the channel section because of friction along the boundary. Neither is this velocity distribution usually axisymmetric, due to the existence of the free surface. It might be expected to find the maximum velocity at the free surface where the shear force is zero but is not the case. The profiles are complex and the maximum velocity occurs in the midplane around 20% below the surface. This is because of the presence of secondary currents which are circulating from the boundaries towards the section center and resistance at the air/water interface. However in very wide and shallow channels the maximum velocity is near the surface and, and the velocity profile is nearly logarithmic, from the bottom to the surface, see figure 2.2.

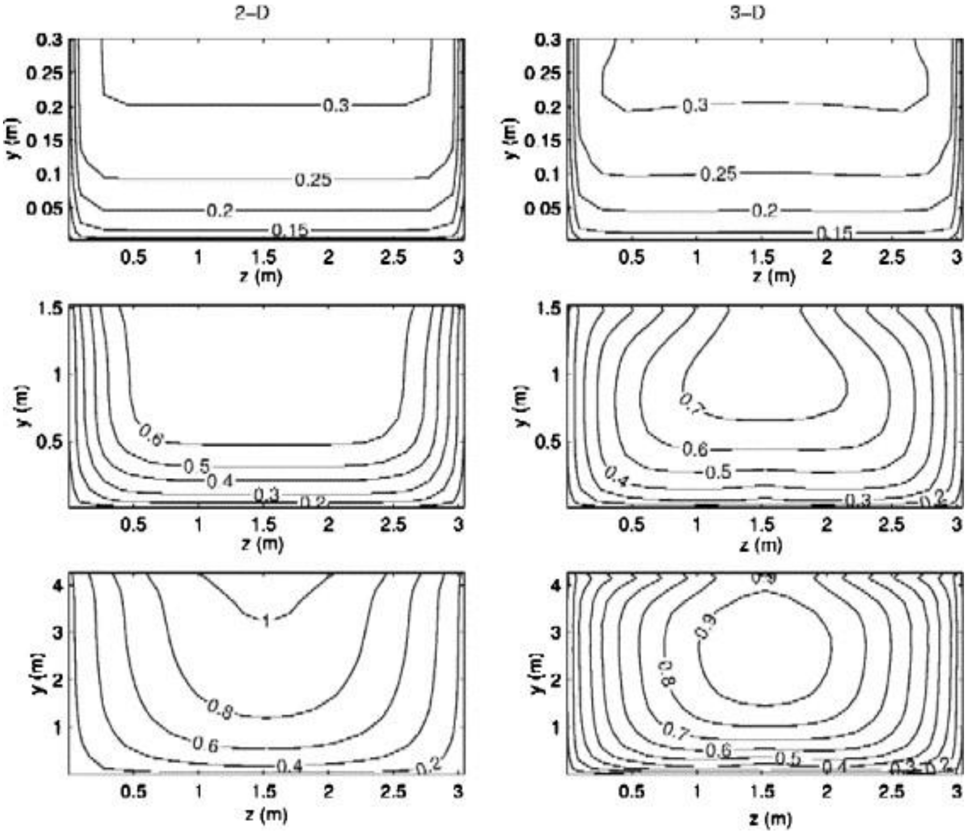


Figure 2.2. Different open-channel velocity profiles. Source, sciencedirect.com

In order to be able to make a numerical simulation of an open channel flow, a one-dimensional flow approximation is made. For the usual open-channel geometry see figure 2.3.

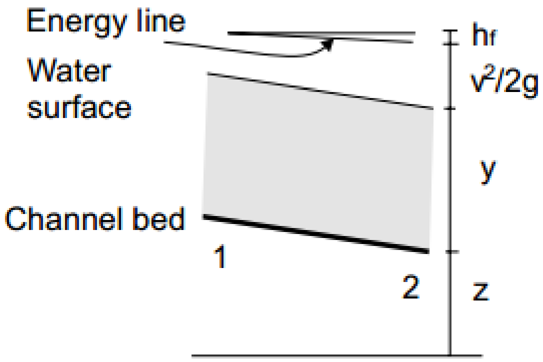


Figure 2.3. Open-channel geometry and notation

If the fluid is incompressible, the continuity equation is as follows,

$$Q = A*V = const \tag{2.12}$$

Where A is the cross sectional area and V is the mean velocity.

The energy or Bernoulli equation is also relates velocity and geometry. In the free surface, where the pressure is constant and equal to the atmospheric value, for steady flow,

$$\frac{V_1^2}{2g} + z_1 = \frac{V_2^2}{2g} + z_2 + h_f \tag{2.13}$$

Where h_f denotes the friction losses and z is the elevation of the free surface, which is the sum of the water depth plus the height of the bottom.

2.3.1 Properties of open channels

ARTIFICIAL CHANNELS

There are channels made by man. They include irrigation canals, navigation canals, spillways, sewers, culverts and drainage ditches. They are usually constructed in a regular cross-section shape throughout – and are thus prismatic channels (they don´t widen or get narrower along the channel). They are usually made out of concrete, steel or ground and have the surface roughness reasonably well defined, although it may change with time. As the channel is well defined, the results will be accurate.

NATURAL CHANNELS

The natural channels are usually not regular nor prismatic and even if they are mainly made out of land, the properties may vary widely. The surface roughness will change with time, distance and even elevation. Consequently, it becomes more difficult to accurately analyze and obtain satisfactory results for natural channels than it does for man-made ones.

For analysis, various geometric properties of the cross- sectional area of the channel are needed. For artificial channels these can usually be defined using simple algebraic equations given *Z* the flow depth.

The commonly needed geometric properties are defined as:
Depth (Z): the vertical distance from the lowest point if the channel section to the free surface.
Stage (z): the vertical distance from the free surface to the arbitrary datum.
Area (A): the cross sectional area of flow, normal to the flow-direction.
Wetted perimeter (P): the length of the wetted surface measured normal to the direction of flow.
Surface width (B): width of the channel section at the free surface
Hydraulic radius (R): the ratio of area to wetted perimeter (A/P).
Hydraulic mean depth (Dm): the ratio of area to surface width (A/B).

2.3.2 Flow classification by Depth Variation

The most common method of classifying open-channel flows is by rate of change of the free surface depth with respect to time and space.

- Steady and Unsteady (with respect to time)
  Flow is said to be steady if the depth of flow at a particular point does not change or can be considered constant for the time interval under consideration. The flow is unsteady if depth changes with time.

- Uniform flow (with respect to space)
  Open-channel flow is said to be uniform if the depth and velocity of flow are the same at every section of the channel. Hence, it follows that uniform flow only occur in prismatic channels. For steady uniform flows, depth and velocity is constant with both time and distance. This constitutes the fundamental type of flow in an open channel. It occurs when gravity forces are in equilibrium with resistance forces.

- Steady non-uniform flow
  Depth varies with distance but not with time. This type of flow may be either gradually varied or rapidly varied. The gradually varied flow uses the energy and frictional resistance equation, while the rapidly varied requires the energy and momentum equations.

- Unsteady flow
  The depth varies with both time and space. This is the most common type of flow and requires the solution of the energy, momentum and friction equations with time. However, in many practical cases the flow is sufficiently close to steady flow, therefore it can be analyzed as gradually varied steady flow.

2.3.3 Laminar and Turbulent flow

The flow in an open-channel may be either laminar or turbulent, as all flows. The criterion for determining the type of flow is the *Reynolds number, Re.* For open-channel flow,

$$Re = \frac{\rho u R}{\mu} \qquad (2.14)$$

Where u is the velocity of the flow and R is the hydraulic radius R=A/P. The analysis is not developed here but the open-channel limits for each type of flow become

Laminar: Re<500
Turbulent: Re>1000

In practice the limit for turbulent flow is not as well defined in channel as it is in pipes and so 2000 is often taken as the threshold for turbulent flow.

2.3.4 Flow classification by Froude number

Another very useful classification of open-channel flow is by the dimensionless Froude number Fr which, as explained before, is the ratio of channel velocity to the speed of propagation of a small-disturbance wave in the channel

$$Fr = \frac{V}{\sqrt{gL}} \qquad (2.2)$$

where *L* is the water depth. The flow behaves differently depending on these three flow regimes:

- Fr < 1.0 Subcritical flow
  Wave velocity > water velocity
  Upstream levels affected by downstream controls

- Fr = 1.0 Critical flow

- Fr > 1.0 Supercritical flow
  Wave velocity < water velocity
  Upstream levels not affected by downstream controls

There is here a strong analogy with the three compressible flow regimes of the Mach number: subsonic (Ma<1), sonic (Ma=1) and supersonic (Ma >1).

2.3.5 Specific Energy; Critical Depth

The total head of any incompressible flow is the sum of its velocity head $\frac{V^2}{2g}$, pressure head $\frac{p}{\gamma}$, and potential head z. For open-channel flow, surface pressure is everywhere atmospheric, so that channel energy is a balance between velocity and elevation head only. The final result is the quantity called *specific energy* E

$$E = y + \frac{V^2}{2g} \tag{2.15}$$

where y is the water depth.

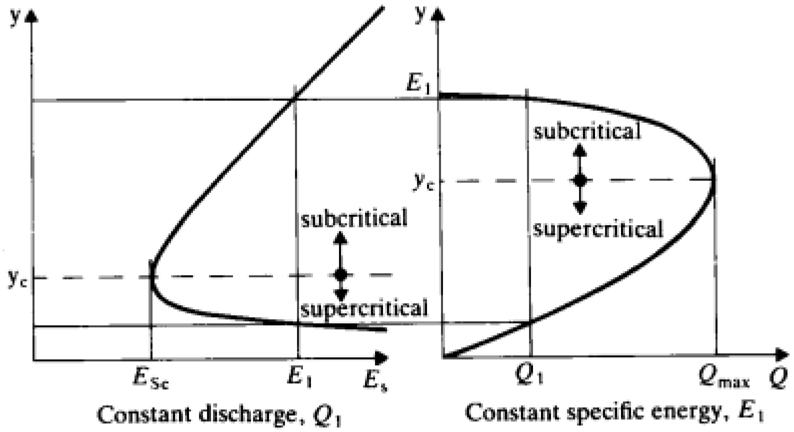


Figure 2.4. Variation of Specific energy and Discharge with depth.

E is the height of the *energy grade line* (EGL) above the channel bottom. For a given flow rate, there are usually two states possible, called *alternative states*, for the same specific energy. There is a minimum energy, $E_{\min}$ which corresponds to a Froude number of unity.

For $E < E_{\min}$ no solution exists and thus such a flow is impossible physically. For $E > E_{\min}$ two solutions are possible: subcritical and supercritical. In subcritical flow disturbances can propagate upstream because wave speed is is bigger than flow velocity. In supercritical flow, waves are swept downstream: Upstream is zone of silence and a small obstruction in the flow will create a wedge-shaped wave exactly analogous to the Mach waves. Small changes in E near E_{min} cause a large change in the depth y. Thus, critical flow is neutrally stable and is often accompanied by waves and undulations in the free surface. Channel designers should avoid long runs near critical flow.

2.3.6 Analytical solution of an open-channel flow

One of the objectives of the thesis is to compare the simulated results for the open-channel with the analytical solution.

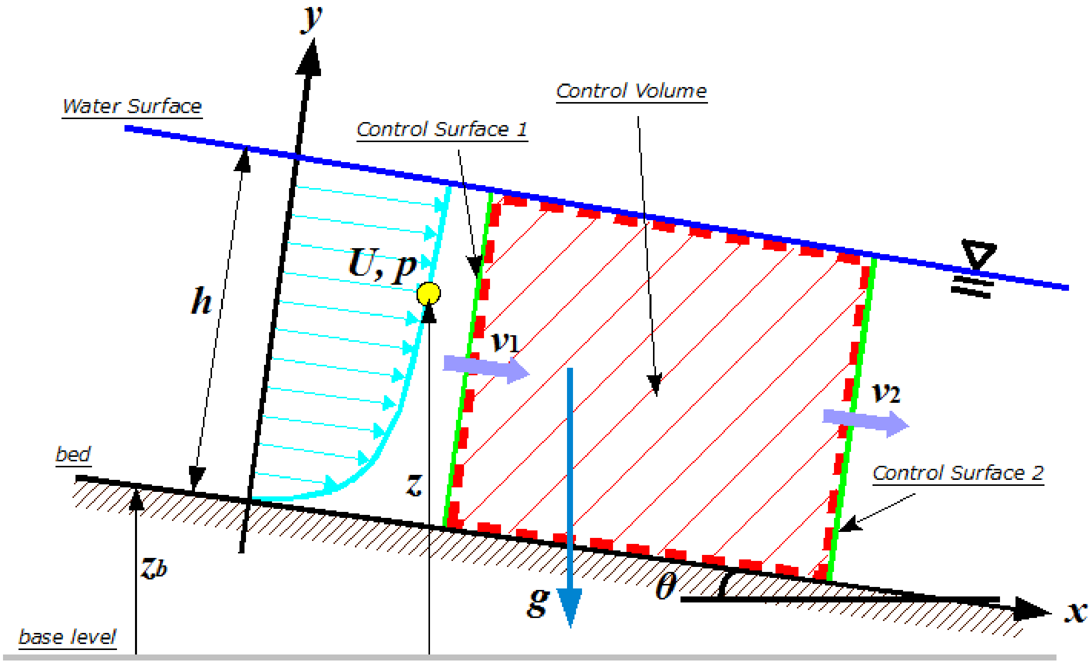


Figure 2.5. Open-channel flow parameters. source: commons.wikimedia.org

The velocity has just one component, which in this case the *u,* or the velocity in the x direction

$$\vec{v} = (u, 0, 0) \tag{2.16}$$

From the figure 2.5 it is deduced that the velocity *u*, is just a function of the coordinate *y,* and the divergence of $\vec{v}$ is zero

$$u = u(y) \tag{2.17}$$

$$\nabla\vec{v}=0 \tag{2.18}$$

The first component of the Navier-Stokes equation reduces to,

$$\frac{Du}{Dt}=0=-\frac{1}{\rho}\frac{\partial p}{\partial x}+g\sin\theta+\nu\frac{\partial^2 u}{\partial y^2} \tag{2.19}$$

This, as the u is a function of y, can only be rewritten as,

$$\frac{Du}{Dt}=0=-\frac{1}{\rho}\frac{\partial p}{\partial x}+g\sin\theta+\nu\frac{d^2 u}{dy^2} \tag{2.20}$$

And the second component,

$$\frac{Dv}{Dt}=0=-\frac{1}{\rho}\frac{\partial p}{\partial y}-g\cos\theta \tag{2.21}$$

Taking the x derivative of the second equation,

$$\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial y}\right)=\frac{\partial}{\partial y}\left(\frac{\partial p}{\partial x}\right)=\frac{\partial}{\partial x}(-\rho g\cos\theta) \tag{2.22}$$

and taking the x derivative of the first equation now,

$$\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right)=\frac{\partial}{dx}\left(\rho g\sin\theta+\rho\nu\frac{d^2u}{dy^2}\right)=\rho\nu\frac{d^2}{dy^2}\left(\frac{\partial u}{\partial x}\right)=0 \tag{2.23}$$

This equals to zero, because *u* just changes with *y* and so the *x* derivative is 0. This means that the *x* derivative of *p* is a constant:

$$\frac{\partial p}{\partial x}=C\rightarrow p(x,y)=C_a x+C_b \tag{2.24}$$

But here the first term must be zero because the pressure is just a function of the coordinate *y.*

$$p=p(y) \tag{2.25}$$

From equation 2.22, the value for the pressure is found,

$$\frac{\partial p}{\partial y} = \frac{dp}{dy} = -\rho g \cos\theta \rightarrow p = -\rho g y \cos\theta + C_1 \tag{2.26}$$

And from equation 2.20, the value for the velocity:

$$\frac{d^2u}{dy^2} = -\frac{g\sin\theta}{\nu} \rightarrow \frac{du}{dy} = -\frac{g\sin\theta}{\nu} y + C_2 \tag{2.27}$$

Finally, in order to solve the problem the boundary and initial conditions are needed. The boundary condition shall be to consider the shear stress zero in the interface between air and water

$$\tau_w\big|_{y=h} = \mu \frac{du}{dy}\bigg|_{y=h} = 0 \tag{2.28}$$

With this boundary condition the analytical solution for the steady state open-channel flow is solved.

If the equation 2.27 is solved, a second order equation is obtained, the equation of a parabola, as expected. First, equation 2.27 is integrated to obtain:

$$u - u_0 = \frac{-g\sin\theta}{\nu}\frac{y^2}{2} + C_2 y \tag{2.29}$$

And according to the no-slip condition at the bottom of the channel, u_0 is equal to zero. In order to obtain the value of the constant, the boundary condition 2.28 is used in equation 2.27. Finally, the value of the constant is:

$$C_2 = \frac{g\sin\theta}{\nu} h \tag{2.30}$$

So the equation for the velocity profile is:

$$u = \frac{-g\sin\theta}{\nu}\frac{y^2}{2} + \frac{g\sin\theta}{\nu}hy \tag{2.31}$$

2.4 Flow in conduits

A conduit is any pipe, tube, or duct that is completely filled with flowing fluid [2]. Examples include a pipeline transporting liquefied natural gas, a micro channel transporting hydrogen in a fuel cell, and a duct transporting air for heating of a building.

Piping systems are encountered in almost every engineering design and thus have been studied extensively. When dealing with pipe flow problems viscosity is of high importance, hence the fluid loses energy due to friction as fluid particles interact with one another and the pipe wall. The information is taken from [18].

2.4.1 Laminar and turbulent flow

As all other types of flows, pipe flows can also be classified into laminar or turbulent flows, with a small transitional region between the two. As said before, the Reynolds number, Re is used to determine which type of flow occurs:

$$Re = \frac{\rho u D}{\mu} \tag{2.29}$$

where u is the velocity and D is the diameter of the pipe.

For a pipe

Laminar flow: Re<2000

Transitional flow: 2000<Re<4000
Turbulent flow: Re>4000

It is important to determine the flow type as this governs how the amount of energy lost to friction relates to the velocity of the flow, and so, how much energy must be used to move the fluid.

Flow in pipes is usually turbulent, but there are some common exceptions such as high viscosity oils and blood flow. During the turbulent state random fluctuating movements of the fluid particles are superimposed on the main flow, and they are unpredictable. Therefore, no complete theory is available to analyze turbulent flow as it is a chaotic process. Most of what is known about turbulent flow has been obtained from experiments with pipes, so it is convenient to study it in this form and also because the pipe flow problem has significant commercial importance.

For a basic difference between turbulent and laminar behavior see figure 2.6.

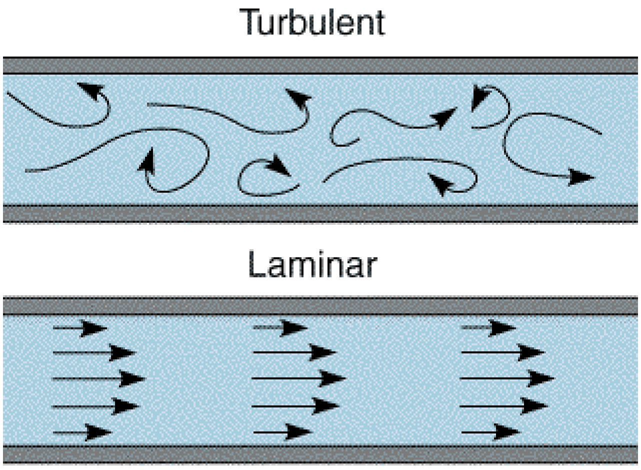


Figure 2.6. Laminar and turbulent flow. Source: www.ceb.cam.ac.uk

2.4.2 Developing flow and fully developed flow

Flow in a conduit is classified as being developing flow or fully developed flow. For example, considering a laminar fluid entering a pipe, as the fluid moves down the pipe, the velocity distribution changes in the stream wise direction as viscous effects cause the plug type profile to gradually change into a parabolic profile. This region of changing velocity profile is called developing flow. After the parabolic distribution is achieved, the flow profile remains unchanged in the stream wise direction, and the flow is called fully developed flow.
The distance required for a flow to develop is called the entrance length (L_e), and depends on the shear stresses that act on the pipe wall, see figure 2.7.

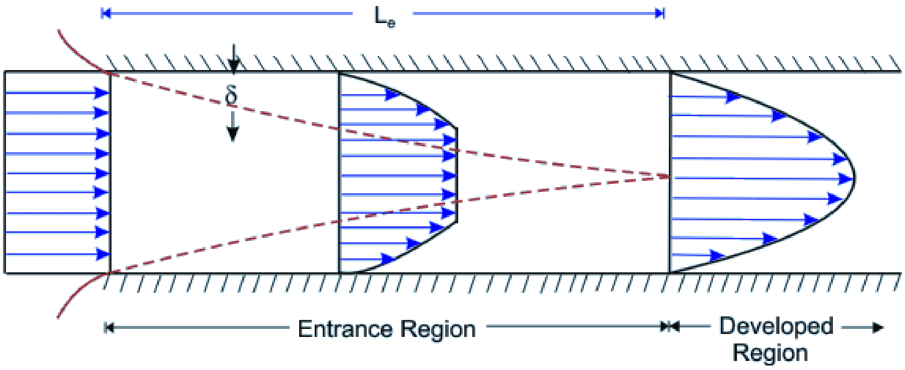


Figure 2.7. Developing flow and fully developed flow. Entrance length. Source: nptel.iitm.ac.in

Near the pipe entrance, the radial velocity gradient (change in velocity with distance from the wall) is high, so the shear stress is large. As the velocity profile progresses to a parabolic shape, the velocity gradient and the wall shear stress decrease until a constant value is achieved. The entry length is defined as the distance at which the shear stress reaches to within 2% of the fully developed value.

Correlations for entry length are:

Laminar flow:

$$\frac{L_e}{D} = 0.05Re \qquad (2.30)$$

Turbulent flow:

$$\frac{L_e}{D} = 4.4\ Re^{1/6} \qquad (2.31)$$

These values are only valid for a flow entering a circular pipe from quiet conditions. If some other elements like valves, elbows or pumps are present, different entrance lengths will occur.

2.4.3 Pressure loss due to friction in a pipeline

Consider a cylindrical element of incompressible fluid flowing in a pipe, see figure 2.8.

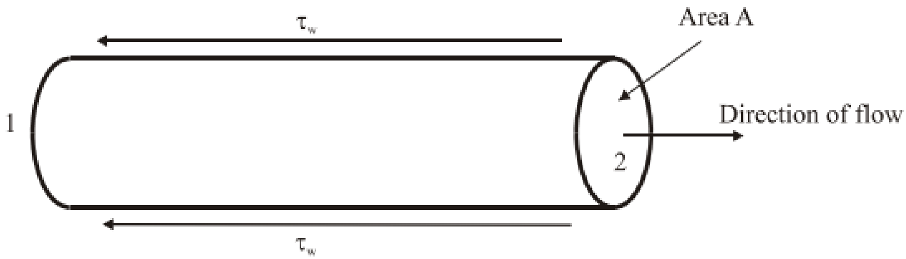


Figure 2.8. Element of fluid flow in a pipe

The pressure at the upstream end 1, is p , and at the downstream end, 2, the pressure has fallen by Δp to $(p-\Delta p)$.

The driving force due to pressure (F= Pressure x Area) can be written

Driving force = pressure force at 1 – pressure force at 2

$$pA-\left(p-\Delta p\right)A=\Delta pA=\Delta p\frac{\pi D^{2}}{4} \tag{2.32}$$

The retarding force is caused by the shear stress by the walls

Retarding force = shear stress x area over which it acts

$$\tau_{w}A=\tau_{w}\pi DL \tag{2.33}$$

As the forces should be balanced,

Driving force = Retarding force

$$\Delta p\frac{\pi d^{2}}{4}=\tau_{w}\pi DL \tag{2.34}$$

$$\Delta\frac{1}{2}p=\frac{\tau_{w}4L}{D} \tag{2.35}$$

So the pressure drop in the pipe depends on pipe lengths, diameter and the shear stress at the wall.
As explained before, the shear stress will vary with velocity of flow, and hence, with the Reynolds number. The relation between the pressure loss and the Reynolds number is shown in the following figure:

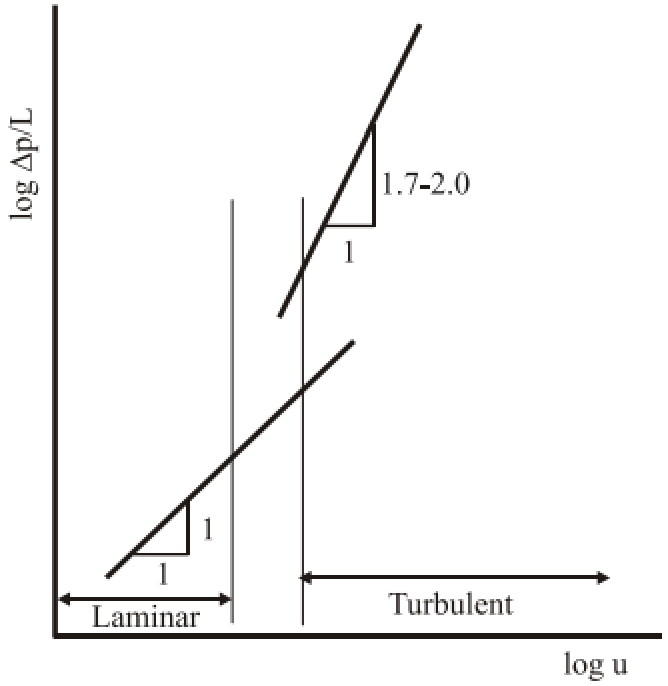


Figure 2.9. Relationship between pressure losses in pipes

From the graph above the following assumptions can be made:

For laminar flow: $\Delta p \propto u$

For turbulent flow: $\Delta p \propto u^a$

where 1.7<a<2.0

Of course, this turbulent relation is empirical, and they help finding the pressure loss from the velocity, but not determining the shear stress at the wall on a particular fluid.

2.4.4 Pressure loss during laminar flow in a pipeline

For laminar flow it is possible to calculate a theoretical value for a given velocity, fluid and pipe dimension. The pressure loss in a pipe with laminar flow is given by the Hagen-Poiseuille equation:

$$\Delta p = \frac{32\mu L u}{D^2} \tag{2.36}$$

2.4.5 Pressure loss during turbulent flow in a pipeline

In general, it is more common in engineering to have Re>2000, I.e. turbulent flow, in both closed and open-channel flows. However, analytical expressions are not available, so empirical relationships derived from experimental measurements are required.

Consider the element of fluid, shown in figure 2.10 below, flowing in a channel. It has length L and wetted perimeter P. The flow is steady and uniform so that acceleration is zero and the flow area at both sections is equal to A.

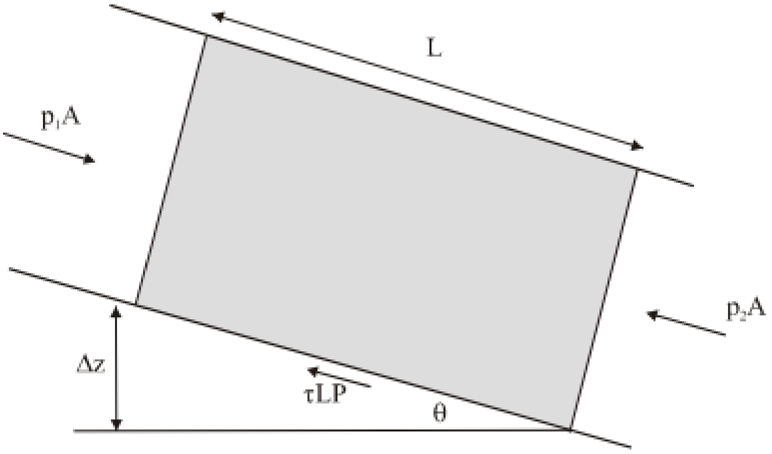


Figure 2.10. Element of fluid in a channel flowing with uniform flow

According to Newton´s second law:

$$p_1 A - p_2 A - \tau LP + W \sin\theta = 0 \qquad (2.37)$$

and writing the weight term as ρgAL and $\sin\theta = -\Delta z / L$ gives

$$A(p_1 - p_2) - \tau LP - \rho gA\Delta z = 0 \qquad (2.38)$$

This can be rearranged to be

$$\frac{[(p_1 - p_2) - \rho g\Delta z]}{L} - \tau\frac{P}{A} = 0 \qquad (2.39)$$

Where the first term represents the piezometric head loss of the length L or, if piezometric head is p*,

$$\tau = m\frac{dp^*}{dx} \qquad (2.40)$$

Where m=A/P is known as the hydraulic mean depth.

2.5 Multiphase flow

From the lecture notes of Fernandino [5] it is known that the multiphase modeling is part of the mathematical model inside the process of making a fluid flow simulation. Any fluid flow consisting of more than one phase or component is a multiphase flow, this can be, gas/liquid, liquid/liquid, gas/solid or liquid/solid.

The multiphase flows can be classified as dispersed flows, intermittent flows and separated flows. These types of flows are further explained below when talking about flow patterns.

What makes the multiphase flow modeling a challenge are plenty. For example, the Navier-Stokes equations describe the flow in each phase and the existence of interfaces makes the problem more complex. Furthermore, one face may be turbulent and the other laminar for example, and even, in the case of disperse flow, strong interaction occurs between particles and between particles and continuous phase.

The most important concepts of this type of modeling are the following:

1. Relative velocity between phases
2. Void fraction, which is the time average fraction of cross-sectional area or volume occupied by a phase, always between 0 and 1

$$\alpha \equiv \frac{V_g}{V} = \frac{V_g}{V_g + V_l} \tag{2.41}$$

3. Superficial velocity or the velocity of a phase as if flowing alone in a single phase model

$$U_{gs} = \frac{\dot{m}}{\rho_g A} \tag{2.42}$$

$$U_{gs} = \alpha U_g \tag{2.43}$$

Multiphase flow problems arise in all kind of engineering problems [3]: Multiphase transport, Cryogenic Heat exchangers, phase separation, chemical reactors and so on.

2.5.1 Flow patterns

In multiphase flow one of the key parameter are the different flow patterns and regimes. In this case the simulations are made either in horizontal, vertical or inclined closed channels. The two phase flow patterns in horizontal tubes are similar to vertical flows but the distribution of the phases is influenced by gravity that acts to stratify the heaviest phase to the bottom of the tube and the gas to the top, [10]. Flow patterns in inclined closed channels adopt the same flow regimes as for horizontal and vertical flow. However, certain regimes exist only in certain inclination intervals. Flow pattern maps for near horizontal flow displays similarities to the horizontal flow maps. As the angle increases similarities to vertical flow is found. The flow patterns for co-current flow of two phases that we are trying to simulate are:

Bubbly-Flow: This regime typically occurs at high oil flow rates and the gas bubbles are dispersed in the liquid and they concentrate in the upper half of the tube. The bubbles are often spherical or spheroidal in shape.

Stratified flow: The two phases separate at low velocities for both phases. The gas goes to the top and the oil goes to the bottom separated by an undisturbed horizontal interface.

Stratified-wavy Flow: When the velocity of the gas increases, waves are formed on the interface and travel in the direction of the flow. Even if the amplitude of the waves is notable, the crests do not reach the top of the tube. This amplitude depends on the relative velocities of the two phases. The waves climb up the sides of the tube and leave thin films of liquid on the wall.

Intermittent flow: If we keep increasing the velocity of the gas the amplitude of the waves increases too, and the crests intermittently wash the top of the tube. Smaller waves can appear in between the large ones. The top wall is wet almost continuously and thin liquid films are left behind. The intermittent flows are sub categorized into:

- **Plug flow:** In this case liquid plugs are separated by elongated gas bubbles. The diameter of the gas bubbles is smaller than the tube diameter and the liquid phase is continuous along the bottom of the tube. It is also known as the elongated bubble flow.

- **Slug Flow**: When we increase the gas velocity, the diameter of the gas bubbles increases too and they are separated by oil slugs. This bubbles have lengths greater than the tube diameter and often have hemispherical shaped tops and flattened tails, bullet shaped (ogival head and flat stern). It is also known as Taylor flow or bubble train flow. The gas bubbles occupy most of the tube section, separated from the channel wall by a thin liquid film.

Annular Flow: When we keep increasing the velocity of the gas, the oil forms a continuous annular film around the perimeter of the tube. The liquid film is thicker at the bottom than the top and it is disturbed by small amplitude waves and some droplets can be dispersed in the gas core too.

Different flow patterns are shown in the figure 2.11.

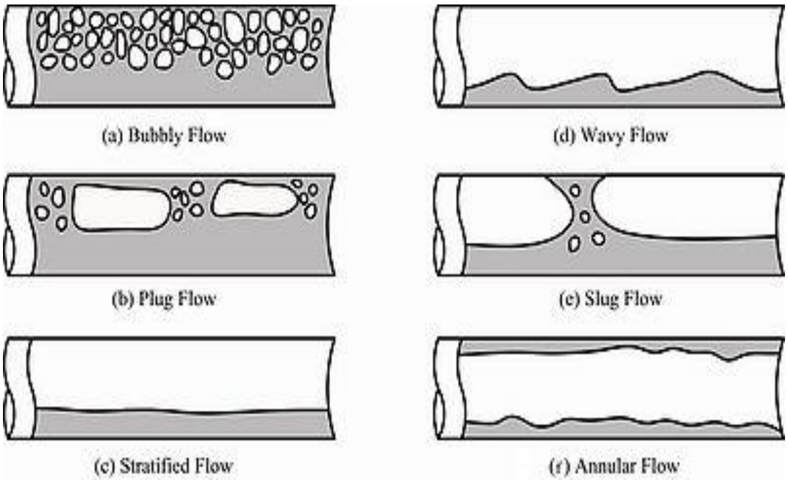


Figure 2.11. Different flow patterns

2.5.2 Multiphase flow models

The multiphase flow consists of gas, liquid and even solid interacting phases, and it is of interest to predict the formation, evolution and interaction between phases.

The flow can be described in two different ways: Lagrangian and Eulerian.

When using Lagrangian description the position and velocity of each particle is tracked. The motion is described based on Newton´s laws. The difficulty comes when trying to describe the interaction among particles and between particles and continuum. Although it cannot be used when there are a very large number of particles, it is very useful to describe the behavior of sprays, particles, bubbles etc. On the other hand, if Eulerian description is used, the domain is subdivided into control volumes and the flow is described when passing through the volume. The challenge with this type of modeling is to formulate closure models for interaction between phases.

When modeling multiphase flow there are several options depending on the case. They are divided into eulerian-eulerian, eulerian-lagrangian and eulerian for granular flow.

Inside the **Eulerian- Eulerian** the following are found:

Homogenous flow models

The main assumption in this type of modeling is that all phases have the same velocity. Thus, it is not applicable in problems with phases travelling at different velocities neither in counter current flows or flows with rapid accelerations and pressure changes.
This is for example the case for blood flow in large conduits (Homogeneous Newtonian fluid) or blood flow in smaller bifurcations (non-Newtonian mixture).

Drift-flux flow models

In this case it is assumed that there is a "drift" between the phases that is measurable, for example caused by different drive forces in each phase, and that there are strongly coupled. This is the case of counter current flows.

$$U_{drift} = U_g - U_m = f(Flow regime, Void\ Fraction, etc) \qquad (2.44)$$

where U_g is the velocity of the gas phase and U_m is the velocity of the mixture of both phases.

Two fluid Models

In this type of flow each phase can coexist in each space-time point. Phase topology is lost and replaced by closures. In this closure that is used as it was just a single fluid, a new parameter appears, α . For example if there is a two phase fluid that consists of liquid and gas, and the enclosure is formed by only the gaseous phase

$$\alpha = \frac{volume\,of\;gas}{total\,volume} \tag{2.45}$$

This model is not applicable when parts of the same phase behave differently.

Multifluid

Here, like in the two fluid models, each phase can coexist in each space-time point, but also can contain distinctive entities. However, it cannot be used if these entities interact changing their identity.

Volume of Fluid (VOF)

In this type of modeling the governing equations are solved for each separated phase, and the interface has to be tracked somehow. Together with the solution to this set of equations the reconstruction of the interface is made. In this type of modeling some extra parameters are needed such as surface tension, wetting angle, etc. It is considered separate flow model and is the one used in the simulations done for this project.

Among the **<u>Eulerian-Lagrangian</u>** models:

Particle tracking

When modeling using particle trajectory modeling, each particle is followed independently, so it is not possible to use it when a high concentration of particles exists.

Population balance

When using this type of modeling the entities can be described in a statistical sense, and also interact between them changing its identity. However, when velocity field changes drastically in these entities, it is not applicable. This is used for breakages or coalescences of droplets, for example.

Multifluid-population balance

This type of modeling is a combination of multifluid and population balance models, as the name suggests. So each phase can coexists in each space-time point, and can contain distinctive entities. And, of course, the entities can be described in a statistical way.

Finally, in the cases where the **Eulerian for granular flow** is used, the continuous phase is described in the eulerian way and the governing equations for the dispersed phase are based on kinetic theory.

2.5.3 Volume of Fluid (VOF)

The solver used in this case (interFoam), uses the VOF (Volume of Fluid) interface capturing approach, due to its good performance with stratified or free surface flows. This model doesn´t take into account the heat and mass transfer. However, it includes the surface tension. The continuity and transport equations for the volume of fraction are ([8], [9]),

$$\frac{\partial \alpha}{\partial t} + (\boldsymbol{U}\nabla)\alpha = 0 \qquad (2.46)$$

Where α is the volume fraction and it is conserved for space and time. It is a scalar value between 0 and 1, where 0 means gas phase and 1 indicates liquid phase.

$\alpha = 0$ Gas phase
$\alpha = 1$ Liquid phase
$0 < \alpha < 1$ The cell contains an interface

The volume fraction affects other material properties such as density or viscosity.

$$\rho\left(\boldsymbol{X},t\right)=\rho_{water}\alpha+\rho_{air}(1-\alpha) \tag{2.47}$$

$$\mu\left(\boldsymbol{X},t\right)=\mu_{water}\alpha+\mu_{air}(1-\alpha) \tag{2.48}$$

2.6 TURBULENCE & TURBULENCE MODELLING

2.6.1 Turbulence

All flows encountered in engineering practice become unstable above a certain Reynolds number. At low Reynolds numbers flows are laminar. At higher Reynolds numbers flows are observed to become turbulent. A chaotic and random state of motion develops in which the velocity and pressure change continuously with time within substantial regions of flow [11].

Many, if not most, flows of engineering significance are turbulent. The Reynolds number of a flow gives a measure of the relative importance of inertia forces and viscous forces. Values below the critical Reynolds number Re_{crit} indicate the flow is smooth. If the applied boundary conditions do not change with time the flow is steady. This regime is called laminar flow.

At values of the Reynolds number above Re_{crit} a complicated series of events takes place which leads to a radical change of the flow character. And finally, the flow behavior is random and chaotic and becomes unsteady even with constant boundary conditions. This is called turbulent flow.

The velocity in this flows can be decomposed into a steady mean value U with a fluctuating component u´(t). In general, the turbulent flows are characterized by the mean values of flow properties (U, V, W, P…) and the statistical properties of their fluctuations (u´,v´,w´,p´ etc.). Turbulent fluctuations always have a three-dimensional spatial character and their visualization reveals rotational flow structures, known as turbulent eddies, which can have a wide range of length scales. Particles of fluid which are initially separated by a long distance can be brought close together by the eddying motions in turbulent flows. And so, heat, mass and momentum change effectively. Large eddies are dominated by inertia effects and viscous effects are negligible. Their structure is anisotropic, directional, and flow dependent due to the strong interaction with the mean flow. Smallest eddies in a turbulent flow are isotropic, non-directional.

2.6.2 TURBULENCE MODELLING

The following information mainly is taken from Versteeg & Malalasekera [11]. A turbulence model is a computational procedure to close the system of mean flow equations. For most engineering problems it is just necessary to solve the effects of the turbulence on the mean flow, and not the details of turbulent fluctuation.

The most common turbulence models are the following. First, the classical models, which are the mixing length model, the k-ε model, the Reynolds stress equation model and the algebraic stress model. And then the large eddy simulations, based on space-filtered equations. Most classical models use the Reynolds averaged equations. Large eddy simulations are turbulence models where the time-dependent flow equations are solved for the mean flow and the largest eddies, and where the effects of the smaller eddies are modeled.

Of the classical models, the mixing length and k-ε are the most used and validated. They are based on the presumption that there exists an analogy between the action of viscous stresses and Reynolds stresses on the mean flow. Mixing length models attempt to describe the stresses by means of simple algebraic equations, and the k-ε model is more sophisticated and general, but more costly. The assumption for both of this models is that the turbulent viscosity μ_t is isotropic, i.e. that the ratio between Reynolds stress and mean rate of deformation is the same in all directions. When using a Reynolds stress equation model the cost of CFD increases substantially. The algebraic stress models are the most economical form of Reynolds stress model able to introduce anisotropic turbulence effects into CFD simulations by making some assumptions from the Reynolds stress equation models.

The model that is used in the present simulations is the k-ε model. The k-ε considers statements regarding the dynamics of turbulence and focuses on the mechanisms that affect the turbulent kinetic energy.

The instantaneous kinetic energy of a turbulent flow is the sum of the mean kinetic energy K and the turbulent kinetic energy k. in the same way, the rate of deformation e_{ij} is a sum of E_{ij} and $e_{ij}^{'}$.

Turbulent kinetic energy

$$\frac{\partial(\rho k)}{\partial t}+div(\rho kU)=div\left(-\overline{p'u'}+2\mu\overline{u'e_{ij}'}-\rho\frac{1}{2}\overline{u_i'.u_i'u_j'}\right)-2\mu\overline{e_{ij}'.e_{ij}'}-\rho\overline{u_i'u_j'}.E_{ij} \tag{2.49}$$

In order, each term of the equation is: rate of change of k, transport of k by convection, transport of k by pressure, transport of k by viscous stresses, transport of k by Reynolds stress, rate of dissipation of k and turbulence production.

The dissipation of turbulent kinetic energy is caused by work done by the smallest eddies against viscous stresses. The rate of dissipation per unit mass, whose dimensions are m^2/s^3, is of vital importance in the study of turbulence dynamics and is denoted by

$$\varepsilon = 2\nu\overline{e_{ij}{}'.e_{ij}{}'} \tag{2.51}$$

The standard k-ε model has two model equations, one for k and one for ε [11]. k and ε are used to define the velocity scale ϑ and the length scale ℓ representative of the large scale turbulence as follows:

$$\vartheta = k^{\frac{1}{2}} \tag{2.52}$$

$$\ell = \frac{k^{3/2}}{\varepsilon} \tag{2.53}$$

The eddy viscosity is defined as follows:

$$\mu_t = C\rho\vartheta\ell = \rho C_\mu \frac{k^2}{\varepsilon} \tag{2.54}$$

where C_μ is a dimensionless constant.

The standard model uses the following transport equations:

$$\frac{\partial(\rho k)}{\partial t} + div(\rho k U) = div\left[\frac{\mu_t}{\sigma_k} grad k\right] + 2\mu_t E_{ij}.E_{ij} - \rho\varepsilon \tag{2.55}$$

$$\frac{\partial(\rho\varepsilon)}{\partial t} + div(\rho\varepsilon U) = div\left[\frac{\mu_t}{\sigma_\varepsilon} grad\,\varepsilon\right] + C_{1\varepsilon}\frac{\varepsilon}{k} 2\mu_t E_{ij}.E_{ij} - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \tag{2.56}$$

Where each term of the equation is, in order, the rate of change ok k and ε, transport of k and ε by convection, transport of k or ε by diffusion, rate of production of k or ε and finally the rate of destruction of k or ε.
In these equations there are several constants that are adjustable $C_\mu, \sigma_k, \sigma_\varepsilon$, $C_{2\varepsilon}$ and $C_{1\varepsilon}$.
Production and destruction of turbulent kinetic energy are always closely linked. The dissipation rate ε is large where production of k is large.

The model equations for k and ε are elliptic, and like other elliptic flow equations, the following boundary conditions are needed:

- Inlet: values and distribution of k and ε must be given.
- Outlet: $\frac{\partial k}{\partial n} = 0$ & $\frac{\partial \varepsilon}{\partial n} = 0$
- Free stream: k=0 and ε=0
- Solid walls: depends on Reynolds number, wall functions etc.

The k-ε is the most widely used and validated turbulence model and performs particularly well in confined flows where the Reynolds shear stresses are most important. However, it shows only moderate agreement in unconfined flows and also has problems in swirling flows and flows with large, rapid extra strains since it doesn´t describe the effects of curvature.

In general, the advantages of the k-ε model are that it is the simplest turbulence model for which only initial and/or boundary conditions are needed, it has an excellent performance for many relevant flows in industry and it is well established. On the other hand, the disadvantages are that it is more expensive to implement that the mixing length model and that it has a poor performance in some cases like unconfined flows, rotating flows or flows with large extra strains.

3. NUMERICAL MODEL

3.1 Case structure

The cases in OpenFOAM have always the same structure. Initially inside the folder of each case 3 other folders are found: *0/, constant/* and *system/*. The folder *0/* contains all the initial and boundary conditions for the different flow fields. Inside *constant/* data from the geometry and the mesh are found, as well as the transport properties and turbulence properties and specifications. Finally, inside the *system/* directory all the data regarding the numerical scheme and time step information are found.

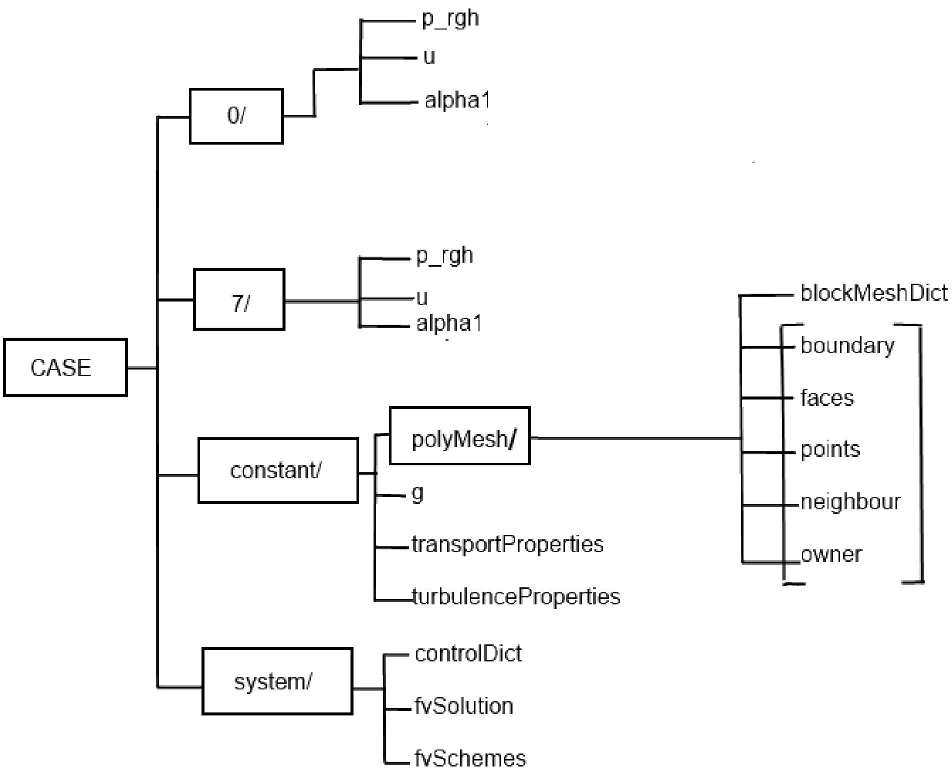


Figure3.1 Example of an OpenFOAM case with final time 7 seconds

3.2 PRE-PROCESSING

3.2.1 CASE GEOMETRY

The simulations are simplified to two dimensional cases. In both types of simulations the geometry is 1 m high for closed channels and 1 or 2 meters high for the open channel, but the length of the channel varies for each case depending on how long it takes for each simulation to develop a reasonable answer. Sometimes a channel that is 20 meters long is enough to recreate a valid answer but some other times it is necessary to have a 100 meters long channel to have some result. The geometry and mesh features are specified in the *blockMeshDict* file and an example of this file can be viewed in APPENDIX A.

The inlets and outlets are defined as `patch` because the fluid enters and leaves the geometry through this faces. The `lowerWall` defined as `wall` because the fluid must no go through this face. For the open-channel flow the `topWall` is defined as a `patch` because it represents the atmosphere and so it must not be a wall. However, in the closed channel flow cases the `topWall` is a `wall` representing the top of the channel.

3.2.2 MESH GENERATION

OpenFOAM always operates in a 3 dimensional Cartesian coordinate system and all geometries are generated in 3 dimensions [13]. OpenFOAM solves the case in 3 dimensions by default but can be instructed to solve in 2 dimensions by specifying a 'special' *empty* boundary condition on boundaries normal to the (3rd) dimension for which no solution is required.

The mesh generator supplied with OpenFOAM, called `blockMesh`, generates meshes from a description specified in an input dictionary, *blockMeshDict* located in the *constant/polyMesh* directory for each given case. An example of the *blockMeshDict* entries for the cases can be found in APPENDIX A.

The mesh quality is essential for the success and accuracy of the results. By typing `checkMesh` in the terminal the specific quality measures are created in the screen. This command ends with the script `mesh OK`. Some data of some of the cases are shown in the tables below. The first table shows the data for some closed channel flow cases.

	PH1-1	**PH20-1**	**PV1-1**	**PV1-20**
Points	2142	6342	20582	16482
Faces	4070	12170	40290	32240
Internal faces	1930	5830	19710	15760
Hexaedra cell	1000	3000	10000	8000
Boundary patches	7	7	7	7
Faces inlet_water	10	10	20	20
Faces inlet_air	10	10	20	20
Faces outlet_water	10	10	20	20
Faces outlet_air	10	10	20	20
Faces lowerWall	50	150	250	200

Faces topWall	50	150	250	200
Faces defaultFaces	2000	6000	20000	16000
Max cell openness	8.67362e-17	8.67362e-17	8.67362e-17	8.67362e-17
Max aspect ratio	8	16	16	20
Minimum face área	0,01	0.005	0.005	0.005
Maximum face area	0,08	0.08	0.08	0.1
Min volume	0,004	0.002	0.002	0.0025
Max volume	0,004	0.006	0.002	0.0025
Total volume	4	12	20	20
Max non-otrthogonality	0	0	0	0
Average non-orthogonality	0	0	0	0
Max skewness	6.25616e-13	1.2502e-12	1.2502e-12	1.00006e-12

Table 1. checkMesh data for closed channel flow cases

The next table shows the data for some open channel cases.

	C3g1H	**C10g1H**	**C25g2H**	**C45g2H**
Points	12382	12382	12382	12382
Faces	24190	24190	24190	24190
Internal faces	11810	11810	11810	11810
Hexaedra cell	6000	6000	6000	6000
Boundary patches	7	7	7	7
Faces inlet_water	20	20	20	20
Faces inlet_air	20	20	20	20
Faces outlet_water	20	20	20	20
Faces	20	20	20	20

outlet_air				
Faces lowerWall	150	150	150	150
Faces topWall	150	150	150	150
Faces defaultFaces	12000	12000	12000	12000
Max cell openness	1.04083e-16	1.04083e-16	1.04083e-16	1.04083e-16
Max aspect ratio	13.3333	13.3333	6.66667	6.66667
Minimum face área	0.005	0.005	0.01	0.01
Maximum face área	0.0666667	0.0666667	0.066667	0.066667
Min volumen	0.00166667	0.00166667	0.00333333	0.00333333
Max volumen	0.00166667	0.00166667	0.00333333	0.00333333
Total volumen	10	10	20	20
Max non-otrthogonality	0	0	0	0
Average non-orthogonality	0	0	0	0
Max skewness	1.50066e-12	1.50066e-12	7.51314e-13	7.51314e-13

Table 2. checkMesh data for open channel cases.

All the values that checkMesh gives seem to be good for the correct performance of the simulations. The maximum and minimum values of the areas and volumes are positive. Non-orthogonality, or the angle between the center to the center line between to neighbor faces and the normal line from the common border of the faces, is equal to zero, and the max skewness remains below a value of 10, which is the critical value for obtaining convergence. Therefore, the values for non-orthogonality and skewness won´t affect the stability of the solution.

3.2.3 BOUNDARY CONDITIONS

The boundary faces are `inlet_water` and `inlet_air` for the inlets, `outlet_water` and `outlet_air` for the outlets and `lowerWall` for the bottom which are common for both cases, closed and open-channel flow. The top wall in the closed channel flow case is known as `topWall` and in the open-channel case is named `atmosphere` due to its different nature.

The inlet of the velocities is the value that changes in the various cases that are under study. The values for the inlets in the closed channel flow of both phases are fixed values that vary depending on the case. In the open channel flow the value for the inlet is set to `zeroGradient`. The outlets of both cases are defined as `zeroGradient`, which means that the velocity normal to the flow outlet is zero. At the walls, the velocity is also a `fixedValue` but it is set to 0, due to the no slip condition of the fluid in the walls. However, when dealing with the open-channel case the top face called `atmosphere` is set to `pressureInletOutletVelocity` which is a combination of `pressureInletVelocity` and `inletOutlet` boundary conditions [16].

For the pressure, since the velocity is fixed in the inlets and the walls, the dynamic pressure is set to `zeroGradient`, so that the gradient normal to the faces is zero, which enables the pressure to float as the velocity is fixed. The outlet boundary condition for the pressure is also set to `zeroGradient`. In the case of the atmosphere boundary, it is set to `totalPressure`. Thus, Total pressure $p_0 = p + \frac{1}{2}\rho|U|^2$ is fixed; when U changes, p is adjusted accordingly [16]. The values down it are settings for this boundary.

The values of k and ε that are to be used in the turbulence modeling are also specified in the boundary conditions.

The model of the boundary conditions that is used in the case is based in the one from the damBreak tutorial.

These and other boundary conditions can be viewed in the APPENDIX B.

3.2.4 FLUID PROPERTIES

The fluid properties are defined in the *transportProperties* dictionary, and can be viewed in APPENDIX C. Water and air are defined in the sub dictionaries of respectively `phase1` and `phase2`. For each phase the keyword `transportModel` is set to `Newtonian`. A Newtonian fluid is characterized by a constant kinematic viscosity which is kept unchanged with the rate of deformation [6]. In this directory the word `nu` defines the kinematic viscosity and is set to 1.48e-05 for air and 1e-06 for water. Density, which is named as `rho`, is set to 1000 kg/m3 for water and 1 kg/m3 for air. The surface tension between air and water is defined as `sigma` and its value is specified to 0.07, adopted from the dam break tutorial.

3.2.5 GRAVITY

In OpenFOAM the gravity is a uniform vector field across the computational domain and can be set in the dictionary *g*. The absolute value of the gravity is always 9.8 m/s2 in every case, but the inclination angle is changed in some cases. This is done in order to give different inclinations to the channels. An example of the gravity directory can be viewed in the APPENDIX D.

3.2.6 TURBULENCE MODELING

The choices for turbulence can be made in the directory called *turbulenceProperties*, under `simulationType`. According to OpenFOAM the choices are `laminar, RASmodel` or `LESmodel`. If `RASmodel` or `LESmodel` are chosen another directory is created, called either *RASproperties* or *LESproperties* where additional model coefficients are defined.

For the open-channel flow the `simulationType` is set to `laminar` for the laminar case Laminar cases are run because the solution to the Navier-Stokes equations to get the analytical solution for the flow are based on a laminar case, so, in order to compare the results, laminar cases have to be developed .

For the closed channel flow cases `RASmodel` is chosen, which means that a Reynolds averaged simulation is used. Inside the *RASproperties* directory the choice of the turbulence model is made. For these simulations the k-ε is the one used, due to his wide validation in engineering problems and because it is supposed to perform well in the cases under study. `kEpsilon` is typed to set it. However, OpenFOAM offers a wide range of different options of turbulence models which can be used instead of the k-ε, like the k-ω, k-ω-SST, RNG-K-ε, and much more that can be found in [15] together with the ones for the LESmodels and others.

These dictionaries can be viewed in APPENDIX E

3.2.7 TIME STEP AND DATA OUTPUT CONTROL

According to the OpenFOAM foundation [8], the surface tracking algorithm in interFoam is more sensitive to the Courant number than other models. It is recommended that the Courant number is equal or less than 0.5 in the interface. In the *controlDict* file the time adjustments can be made, which can be studied in APPENDIX F Here it is not useful to have a fixed time step because the propagation of the velocity is not easily predicted, so `adjustTimeStep` is set to `yes`. `maxAlphaCo` sets the maximum Courant number applied to alpha1 and is set to `0.5`. `maxCo` is the same as the previous keyword but it is applied to other fields such as `p_rgh` and `U`, and is also set to `0.5`. The Courant number is defined as

$$Co = \frac{\Delta t \left| U \right|}{\Delta x} \tag{3.1}$$

where Δx is the width of the cell in the velocity direction, Δt is the time step. The maximum Courant number is a result of a small cell and a high velocity.

`maxDeltaT` determines the upper limit to the time step, and it is set to `1`, like in the dam break tutorial. The keyword `writeInterval` sets the time when the results are written, even if in OpenFOAM the calculations are performed at arbitrary time steps, and is set to `0.05`. `writeControl` is set to `adjustableRunTime` to allow this.

Then the `startFrom` keyword is set to `startTime`, and `startTime` is set to `0`, so the first fields data input is read from the *0/* directory. `endTime` is a variable that has to be adjusted from case to case, because depending on the result needed more time will be needed. In these cases it varies from 10 to 20 seconds. Finally `stopAt` is set to `endTime`. There are keywords regarding time control.

`writeFromat` is set to the default value `ascii`, but it could also be written in the binary format by typing `binary`. The keyword `writeFormat` is set to `6`, adopting the value from the dam break tutorial and `writeCompression` is set to `uncompressed`, because the data files are not too big, but if the reduction is needed it can be set to `compressed` and the space occupied by the cases will be smaller. Finally, the `timeFormat` is set to `general` and `timePrecision` is set to `6` which are default values for OpenFOAM.

3.2.8 fvSolution

The pressure velocity coupling is set to PIMPLE under the *fvSolution*. The PIMPLE algorithm has two loops, inner and outer. In the outer loop all equations are solved while in the inner loop only the continuity one. It is important to have the continuity error under control. It is important also for the max and mean Courant number not to increase too much, but they will still have higher values than when using the PISO algorithm. Also, the initial residuals should be kept small for the last outer loop. All of these are important in order to get a reasonable converged solution.

In the *fvSolution* directory, the `nCorrectors` indicates how many times the pressure equation is solved in the outer loop, and it is recommended to use minimum 2 to get strict time accuracy. `nNonOrthogonalCorrectors` are used if the mesh is sufficiently non-orthogonal. When time steps are small and `nCorrectors` are in use this value may be 0 generally. Both of these values should be kept low, since they would increase the computational cost if not. `nAlphaCorr` indicates the loop over the volume fraction and it should be 1-2 for transient flows and 0 for steady state flows. `nAlphaSubCycles` represents the number of sub cycles `alpha1` is calculated. In this case is set to 2 for laminar cases and 4 for turbulent cases which means that alpha is calculated 2/4 times during each time step. `cAlpha` sets the compression term in the modified transport equation, the compression at the interface. 1 means conservative compression, while it is deactivated at a value 0. If a value greater than 1 is given, the compression is enhanced. In this case the value 1 is used, conservative compression. In the sub-directory solvers, the method for solving the discretized equations is specified. The equations solved are the pressure correction `pcorr`, the first pressure loop `p_rgh`, the second and last pressure loops `p_rghfinal` and the velocity equation `u`. Solver is set to `PCG` for the equations regarding pressure which stands for preconditioned conjugate gradient, and can be used with symmetric matrices. If this solver is applied to an asymmetric matrix OpenFOAM will create and error message. For the velocity equation `u`, solver is set to `PBiCG`, which stands for Preconditioned Bi-Conjugate Gradient, which can be applied to asymmetric matrices. These solvers generate iterative solutions, so residuals are created. The residual indicates the error in the solution. If the residual is lower than the `tolerance`, the ratio of current to initial residual is lower than `relTol` or if the number of iterations is bigger than `maxIter`, the solver will stop running. The values of tolerance and relative tolerance in this case are:

	p_corr	p_rgh	p_rghFinal	"(U|k|epsilon)" (Turbulent)	"(U|k|epsilon)Final" (Turbulent)	U (Laminar)
Solver	PCG	PCG	PCG	PBiCG	PBiCG	PBiCG
preconditioner	DIC	DIC	DIC	DILU	DILU	DILU
tolerance	1e-10	1e-7	1e-7	1e-06	1e-08	1e-06
relTol	0	0.05	0	0	0	0

Table 3. Data for the solver, preconditioner, tolerance and relative tolerance

maxIter is an optional value and it is not defined in this case.

For the preconditioner keyword, DIC is selected for all the pressure equations, which means Diagonal Incomplete-Cholesky, useful for symmetric matrices, and DILU for the velocity equations, Diagonal Incomplete-LU, for asymmetric matrices.

The *fvSolution/* directory can be viewed in APPENDIX G.

3.2.9 DISCRETIZATION SCHEMES

Inside this directory, the keyword *divSchemes* defines the convection schemes in the momentum equation, $\nabla * \rho UU$ defined as div (rho*phi, U) in OpenFOAM. Here the scheme GausslimitedLinearV 1 is used, where the number 1 expresses the coefficient ϕ, which is the flux phi that is used in the interpolation of the convection term, and is the most stable option. This scheme ensures good accuracy. Following, the numerical scheme div (phi, alpha), $\nabla * U\alpha$, is set to Gauss vanLeer and the div(phirb, alpha), $\nabla * U_{rb}\alpha$, is set to Gauss interfaceCompression.
According to OpenFOAM [13], the *ddtShemes* represents the choice of time scheme. In this case Euler is adopted and indicates a first

order bounded implicit scheme, which is sufficiently accurate due to the small time steps created by the Courant number restriction. The keyword `gradSchemes` defines the discretization schemes for the gradient terms, and in this case is `default Gauss linear`. Default means that the Gauss linear scheme will be applied to all the gradient terms. When a Gauss discretization is used the values are interpolated from cell centers to face centers. `laplacianSchemes` defines the laplacian scheme, and it is applied to the terms with the laplacian operator ∇^2 . In this case it is set to `default Gauss linear corrected`. Here the word corrected expresses the surface normal gradient scheme and indicates a unbounded, conservative and second order numerical behavior. For the interpolation schemes, OpenFOAM offers centered, upwind convection, TVD (total variation diminishing) and NVD (normalized variable diagram). Here a centered interpolation scheme is used under `interpolationSchemes`, `default linear`, which is the one most generally used. `snGradSchemes` indicates the surface normal gradient schemes, and evaluates the gradient normal to the face center shared by two cells. In this case it is `default corrected`. Finally, `fluxRequired` defines the fields that are solved before the flux, which are `p_rgh, pcorr` and `alpha1.`

The *divSchemes* directory can be found in APPENDIX H.

3.3 PROCESSING. RUNNING THE CODE

Prior to the running of the solver, some other steps have to be done and this are ran by using the script `runScript.sh`, for example, erasing the previous data on the case. Due to the iterative character of the simulations it is really important to erase the data previously obtained because if not, the results could be just partially overwritten to the previous simulation and maybe not correct. For this purpose, `foamClearPolyMesh` and foamCleanTutorials are used. This way, every time the simulation is ran, it starts clean, without previous data.

During the simulation the setFields utility reads fields from the files and after re-calculating, writes them back to the file. Because the files are then overridden, it is recommended that a backup is made before setFields is executed. The alpha1 field is initially stored as a backup *only*, named alpha1.org. Before running setFields, the user first needs to copy alpha1.org to alpha1 [13].

After that, by running blockMesh, the mesh is generated and any mistakes in the *blockMeshDict* file are picked up by blockMesh and the resulting error message directs the user to the line in the file where the problem occurred [13].

Finally, setFields is executed to set the phase fractions into the domain and as explained above, reads fields from the files and after re-calculating, writes them back to the file.

The script runScript.SH is used, which can be viewed in APPENDIX I.

When the next step is interFoam the simulation starts running. Functions specified in *controlDict* are sampled and written to different directories.

Execution time of the cases considerably limits the amount of cells in the cases. As the cell size decreases, so does the time step, due to the fixed Courant number of maximum 0.5. Increased number of cells will reduce the time step further in addition to the extra computational time to compute the governing equations for additional cells.

It is also possible to run in parallel, but this is not done in any of the cases, because the number of cells is not that big. Dividing the mesh also requires the computer to communicate results across the divided parts. It is recommended when cell number in the case exceeds 100000.

3.3.1 SOLVER: INTERFOAM

The simulation is meant to be a multiphase problem. Thus, the solver needed has to be able to deal with these kinds of problems. The solver used in this case is called interFoam. InterFoam is a two-phase solver for incompressible, isothermal and immiscible fluids using a VOF (volume of fluid) based interface capturing approach [8].

When using interFoam, the phase fraction varies from 0 to 1, but the interface between both phases is never very sharp, it occupies an area that should be just a line in the plane.

InterFoam solves the Navier-Stokes equations for fluid flow. The Reynolds-average momentum equation looks like

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \boldsymbol{U}\boldsymbol{U} = -\nabla P + \rho \boldsymbol{g} + \nabla[(\mu + \mu_t)(\nabla \boldsymbol{U} + \nabla \boldsymbol{U}^{\boldsymbol{T}})] \qquad (3.2)$$

$$P = \rho \boldsymbol{g}\boldsymbol{X} + p_d \qquad (3.3)$$

Here $\rho \boldsymbol{g}$ is the acceleration of the fluid because of the gravity.

Taking the gradient,

$$\nabla P = \nabla\left(\rho \boldsymbol{g}\boldsymbol{X}\right) + \nabla p_d \qquad (3.4)$$

$$\nabla P = \rho \boldsymbol{g} + \boldsymbol{gX}\nabla\rho + \nabla p_d \tag{3.5}$$

In the interface the gradient of density is very big but at places that are far from this the change in density can be considered as 0. If this is used in the Reynolds-average momentum equation,

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \boldsymbol{UU} = -\nabla p_d - \boldsymbol{gX}\nabla\rho + \nabla[(\mu+\mu_t)(\nabla \boldsymbol{U} + \nabla \boldsymbol{U^T})] \tag{3.6}$$

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \boldsymbol{UU} = -\nabla p_d - \boldsymbol{gX}\nabla\rho + \nabla[\mu_{eff}(\nabla \boldsymbol{U} + \nabla \boldsymbol{U^T})] \tag{3.7}$$

where μ_{eff} is the effective dynamic viscosity and equal to $(\mu+\mu_t)$.

Developing the last term in the equation (3.7)

$$\nabla\left[\mu_{eff}\left(\nabla\boldsymbol{U}+\nabla\boldsymbol{U^T}\right)\right] = \nabla\left(\mu_{eff}\nabla\boldsymbol{U}\right) + \nabla\boldsymbol{U}\nabla\mu_{eff} + \mu_{eff}\nabla\left(\nabla\boldsymbol{U}\right) = \nabla\left(\mu_{eff}\nabla\boldsymbol{U}\right) + \nabla\boldsymbol{U}\nabla\mu_{eff} \tag{3.8}$$

Inserting this in the Reynolds average momentum equation,

$$\frac{\partial \rho U}{\partial t} + \nabla UU = -\nabla p_d - gX\nabla\rho + \nabla\left(\mu_{eff}\nabla U\right) + \nabla U\nabla\mu_{eff} \tag{3.9}$$

In order to create a linear system to solve U in the three directions, interFoam just takes into account the convective and viscous terms. InterFoam uses also the PIMPLE algorithm. PIMPLE algorithm is a combination of PISO and SIMPLE algorithms; see APPENDIX J The structure of PIMPLE is based on the original PISO, but allows equation under-relaxation to ensure the convergence of equations, like in SIMPLE.

3.4 POST-PROCESSING

The main post-processing tool provided with OpenFOAM is ParaView, an open-source visualization application. ParaView uses the Visualization Toolkit (VTK) as its data processing and rendering engine and can therefore read any data in VTK format. OpenFOAM includes the foamToVTK utility to convert data from its native format to VTK format, which means that any VTK-based graphics tools can be used to post-process OpenFOAM cases [14]. This provides an alternative means for using ParaView with OpenFOAM. ParaView is launched by writing paraFoam in the terminal. ParaView also gives the option of doing animations with the simulations.

4. RESULTS AND DISCUSSION

4.1 General comments

In the open-channel cases the input and output conditions for both the velocity and the pressure are set to zeroGradient, which means that $\frac{\partial u}{\partial x} = 0$ and $\frac{\partial p}{\partial x} = 0$ in both sides. For the closed channel cases, the input velocities are constant values, different for each phase, and the outlet velocity condition is set to zeroGradient, like in the open-channel flow. The pressure conditions here are also the same for the inlet and the outlet.

The domain is split in two blocks so that the different phases can have different velocities set.

When the simulations start it takes some time to reach a steady state, or sometimes it was not able to reach it without increasing the computational cost to limits that were too high for the computer power available. However, after several time steps some results could be intuited. Based on these results some conclusions were reached and are described case by case.

The turbulence model used in every case is the k-ε, and there was no time to test other models. All the open-channel cases are simulates using a laminar base, but just one of them gives a laminar profile because the viscosity of the water was set higher in order to get a laminar Reynolds number value. The rest, even if they were simulated using a laminar set up, always show turbulent velocity profiles.

The influence of the change in the inlet values of k and ε was also analyzed. When changing these values the following changes occurred:

10-1 Kwater-Kair	**K water**	**K air**	**Epsilon water**	**Epsilon air**	**K max**	**Epsilon max**
100%-100%	0.08	0.003	0.05	0.0003	86.55	1898.31
75%-75%	0.06	0.002	0.04	0.0002	+10%	+4.6%
75%-100%	0.06	0.003	0.04	0.0003	+2.1%	-4.3%
100%-75%	0.08	0.002	0.05	0.0002	-2.9%	-25.9%
50%-50%	0.04	0.001	0.02	0.0002	+3.6%	-5.4%
50%-100%	0.04	0.003	0.02	0.0003	+19.7%	-10.1%
100%-50%	0.08	0.001	0.05	0.0002	+2.4%	-9.6069%
125%-125%	0.09	0.003	0.06	0.0004	+2.9%	-12.08663%
125%-100%	0.09	0.003	0.06	0.0003	+7%	+1.5%
100%-125%	0.08	0.002	0.05	0.0004	+4.7%	-1.2%

Table 4. Influence of the changes in the inlet values of k and ε

As it can be seen in table 4, the change in the inlet values of k and ε cause no remarkable alterations in the results, even if the changes made are quite big except in some isolated cases. So, it can be concluded that these turbulence model quantities don´t influence the results of the simulations in a big scale and therefore standard input values are used, based on the damBreak tutorial.

The Reynolds numbers and the Entry lengths are calculated and shown in the table 5.

	Re Air	**Re Water**	**Le Air (m)**	**Le Water (m)**
LAMINAR	0	78.6	0	7.85
C3G1H	2413	3.75e8	1689	3.75e7
C5G2H	2815	4.16e8	1689	3.57e7
C45G2H	2815	2.58e9	1689	2.14e7
PH1-1	4826	7.1e7	36.1	179

PH10-1	26594	3.9e8	48	238
PH1-10	77220	1.1e8	57.3	193
PH15-1	22924	3.4e8	46.9	232
PV1-1	135135	8.6e8	441	271
PV10-1.5	164092	4.3e8	455	271
PV1-10	144787	5e8	446	241
PI3G	19305	4.3e8	319	248
PI5G	12065	4.3e8	295	242
PI10G	14478	3.4e8	304	242

Table 5. Reynolds numbers and entry lengths

For the open-channel cases, which have been run using a laminar setup, the case where the viscosity was changed, the laminar case, is the only one that gives values that are under a limit that can be realizable. The rest of the cases give entry length values way too big for the program to develop a flow completely. This may be because the cases, which are turbulent, were imposed a laminar set up in OpenFOAM, and so the values obtained don´t match the reality, as the formulas for calculating the entry length vary for laminar and turbulent cases.

For the closed-channel cases, the values of the Reynolds numbers indicate turbulent flows, and so were simulated. However, the entry lengths that this Reynolds values give are bigger than the geometry created for the simulations, and it can be concluded that the flows are not fully developed.

4.2 Open-channel flow

The results of the laminar simulation are to be compared with the analytical results obtained in the literature study part 2.3.6.

If this equation 2.31 is analyzed, the only variable is the height of the water from the bottom of the channel. The velocity is zero at the bottom of the channel and it develops in a parabola form as the height increases.

Calculating the maximum velocity of an open channel flow:

$$u_{\max} = u(y=h) = \left(\frac{-g\sin\theta}{2\nu} + \frac{g\sin\theta}{\nu} \right) h^2 = \frac{gh^2\sin\theta}{2\nu} \tag{4.1}$$

If the laminar case is analyzed, the viscosity that appears in the equation 2.31 and 4.1 has to be adjusted in order to keep the Reynolds number smaller than 500. After that, this equation is used to build a matlab code that creates a velocity profile for a certain water depth and inclination; so that the analytical solution can be compared with the one obtained using OpenFOAM.

FROUDE NUMBER

In order to calculate the Froude number for these cases the formula 2.1 is used. The values obtained are the following:

Case	**v**	**Froude number**
c3g1h	9	4.066
c5g2h	6.4	2.044
c45g2h	31	9.903

Table 6. Froude Numbers

In this case then, as the Froude number is higher than 1, the flow is considered to be supercritical. This means that the water velocity is higher than the wave velocity. Also, it can be said that the upstream levels are not affected by the downstream controls.

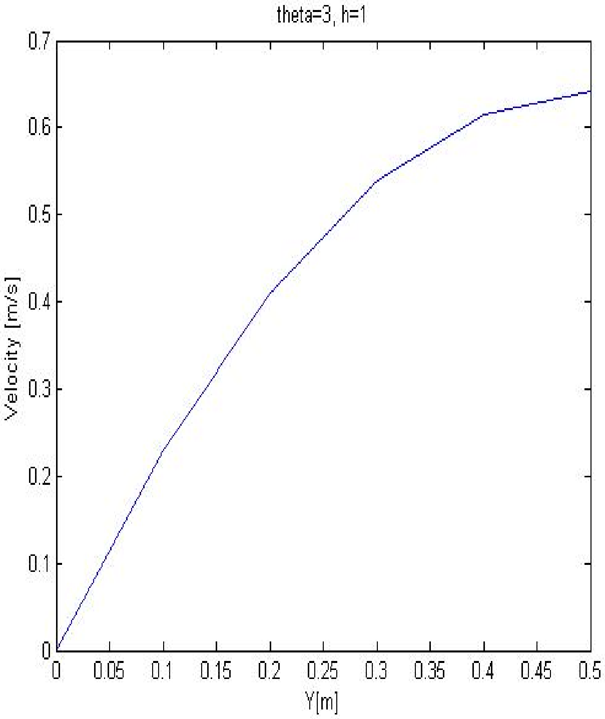
The cases simulated here are not very explanatory when it comes to analyze the behavior of the wave, because due to the necessity of making the simulation files as small as possible the channel was not big enough to develop a behavior that was able to give much information about the wave propagation.

CASE LAMINAR C3G1H

This case is an open-channel simulation of 50 meters long and 1 meter high channel with 3 degrees inclination.

Figure 4.1 Steady state for a laminar open-channel flow

It is a laminar stratified flow. The water viscosity value has set in 0.1 m²/s in order to get a low Reynolds number to ensure it is a laminar flow.



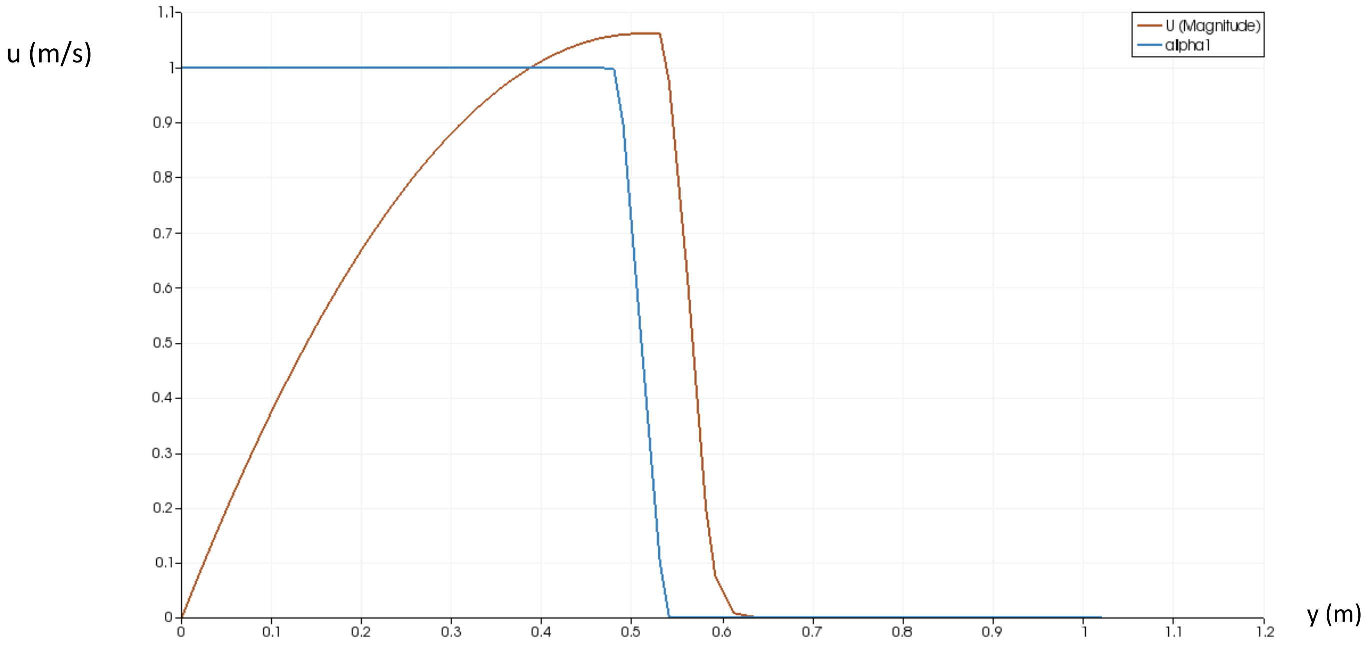


Figure 4.2 a) Comparison between OpenFOAM results and analytical results using matlab

As it can be seen in the Figure 4.2 the flow develops a laminar curve with similar values as the ones obtained by the analytical solution. The laminar open-channel flow compares favorably with the analytical solution, both parabolic profiles with the maximum velocity at the interface, $u_{\max,analytical} \simeq u_{\max,OpenFOAM}$.

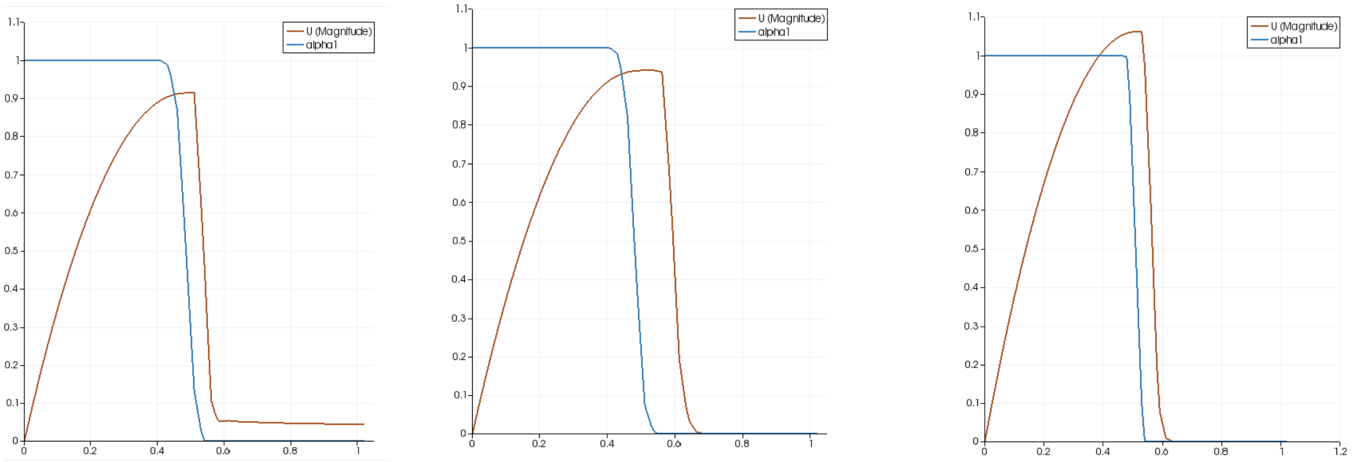


Figure 4.2 b) Evolution of velocity profile. Time=10sg. Distance X=0.5, 7 and 25m

The Reynolds number of 78 for this case indicates a laminar flow indeed and almost a completely developed velocity profile can be seen as the entry length is 7.85m. However, in the results OpenFOAM gives it changes slightly from 7 meters to 25 meters, but not in a very notorious way.

CASE C3G1H

This first case is an open-channel simulation of 50 meters long and 1 meter high 2D channel with 3 degrees inclination.

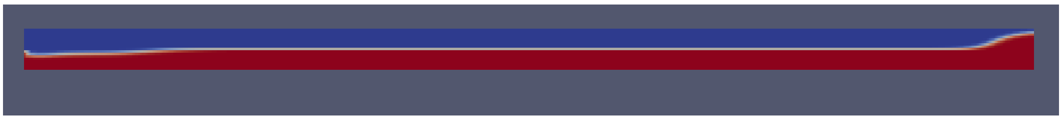


Figure 4.3. CASE C3G1H phase distribution at t=4s

As shown in figure 4.3 the flow is stratified as expected. The water moves as a consequence of the gravitational force and not because of the pressure variation.
Initially the upper side is only with air over the water layer but at the end it can be seen how water starts to fill the right part, and the air fills the rest.
The velocity is bigger in the water than in the air and they respect the no-slip condition in the surface of the wall. The biggest velocity is achieved when the interface starts not to be just water.
The pressure values do not change a lot. As explained above, the top wall is open to the atmosphere and the value of the pressure there is constant and only the value in the liquid section has a considerable change. It depends on the depth as equation 2.26 explains.

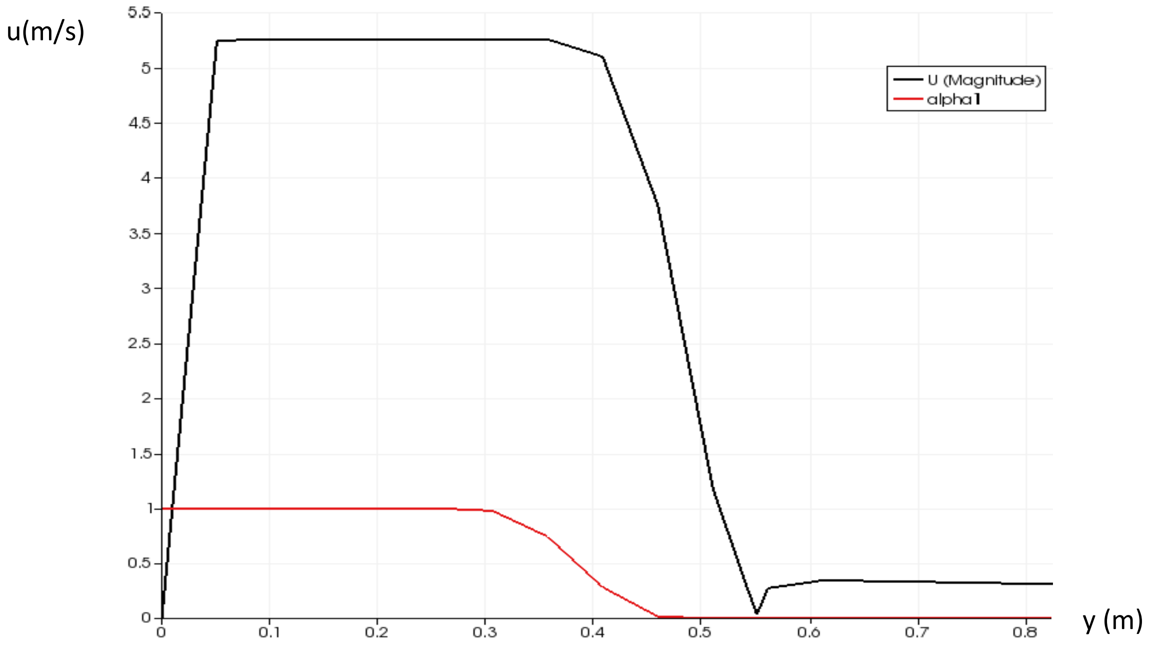


Figure 4.4. CASE C3G1H Velocity values given by OpenFOAM at t=6.85 and x=25m

As it can be seen in figure 4.4, the maximum velocity occurs at the water-air interface, as expected. The velocity profile is a turbulent velocity profile.

CASE C5G2H

This case is of an open-channel of 50 m long and 2 m height with 5 degrees inclination.

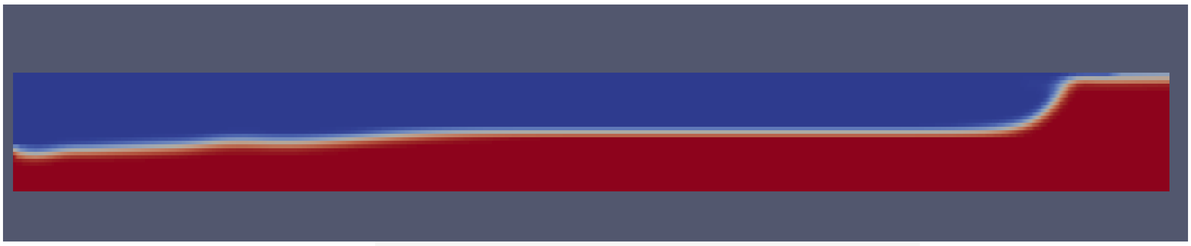


Figure 4.5. CASE C5G2H phase distribution at t=4s.

It is similar to the previous case but with some instability and the end. The liquid phase gets the top of the channel in the right part. Also it is difficult to take measures as the volume of liquid has a considerable variation in time. The initial distribution changes and finally almost all the water is at the end of the channel and all the air at the beginning.

Values of 4.75 m/s velocities are obtained in the middle of the channel, as it can be seen in figure 4.6, just before the instabilities occur as a wave starts to move around the whole channel and disturbs the previous measures.

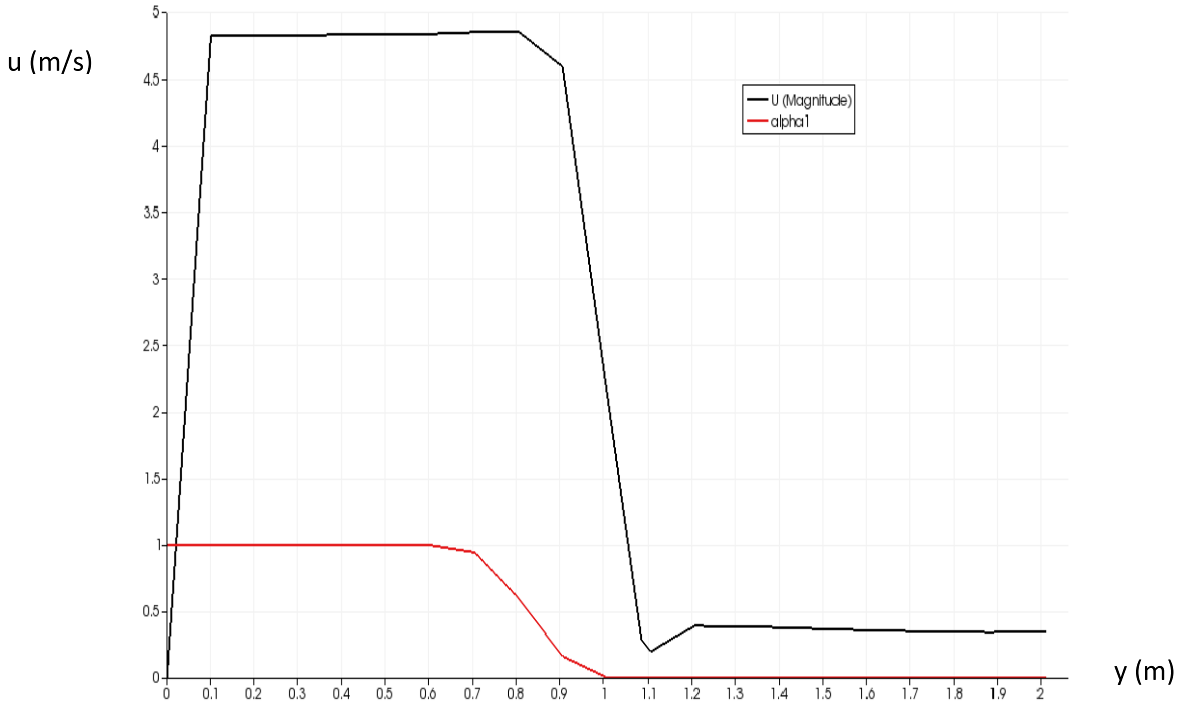


Figure 4.6. CASE C5G2H Velocity values given by OpenFOAM at t=6.35s and x= 25m

CASE C45G2H

This simulation is from a channel of 50m long, 2m high and an inclination of 45 degrees.

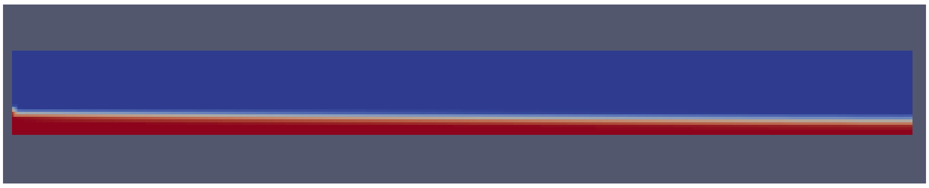


Figure 4.7 CASE C45G2H phase distribution at t=9s.

This flow has the problem that the inclination is too big. The inclination is such big that makes all the liquid phase abandons the channel in a bigger speed than the inlet one. This makes the channel end almost without water. This effect should not happen but even using zeroGradient boundaries for the inlets and outlets it is not possible to maintain a flow like the one set in the initial conditions. This problem besides being only aesthetic, is functional as it makes very complicated taking measures in order to check the simulation with analytical values. Water velocity values are used for comparing the reliability of the simulations and without water remaining is impossible to do it.

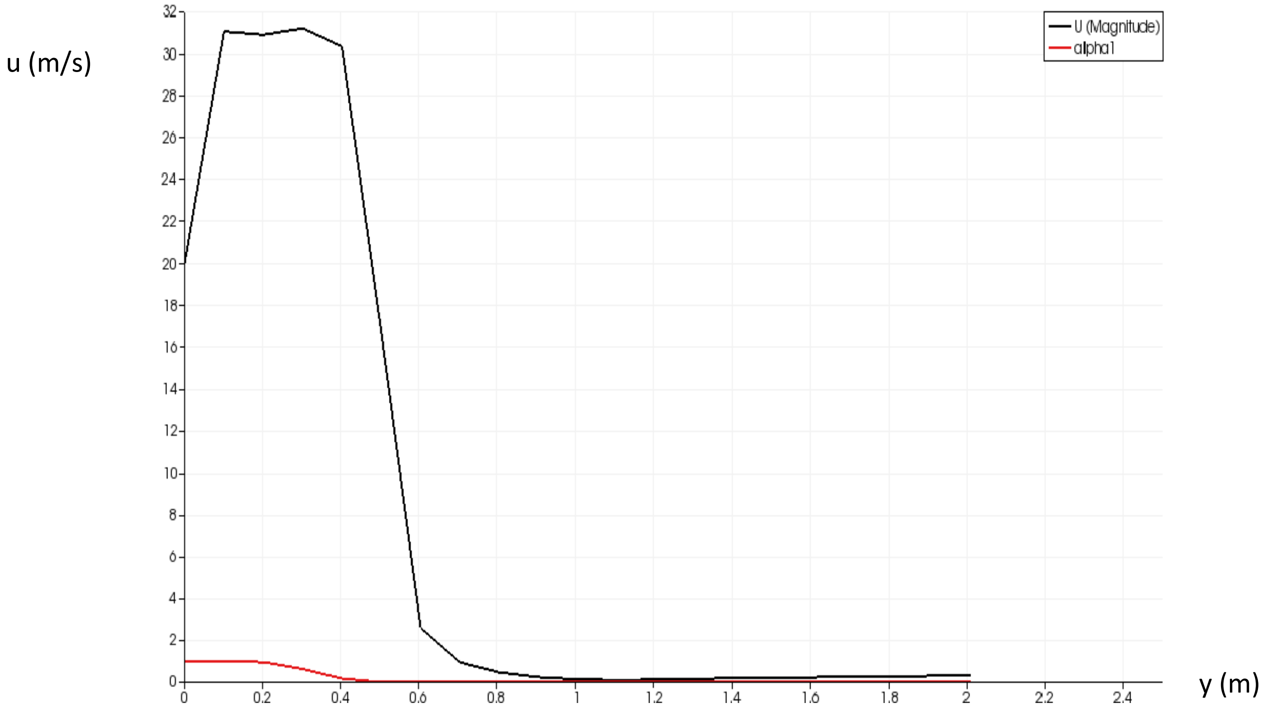


Figure4.8 CASE C45G2H Velocity values given by OpenFOAM at t=12s and x=25m

4.3 Closed channel flow

One of the objectives of the thesis is to generate the flow patterns that are characteristic features of multiphase flow. As explained in the literature study, in order to achieve these patterns the velocity of the phases has to be manipulated. Once the simulations are made the results are compared to experimental results. The **Taitel-Dukler** map defines the transition between different flow regimes. The results are shown in the form of a cross plot or map with the superficial gas velocity, U_{gs}, on the x-axis and the superficial liquid velocity, U_{ls}, on the y-axis. This maps are not general, but only valid for certain characteristics. Figure 4.9 is an example of a Taitel-Dukler map.

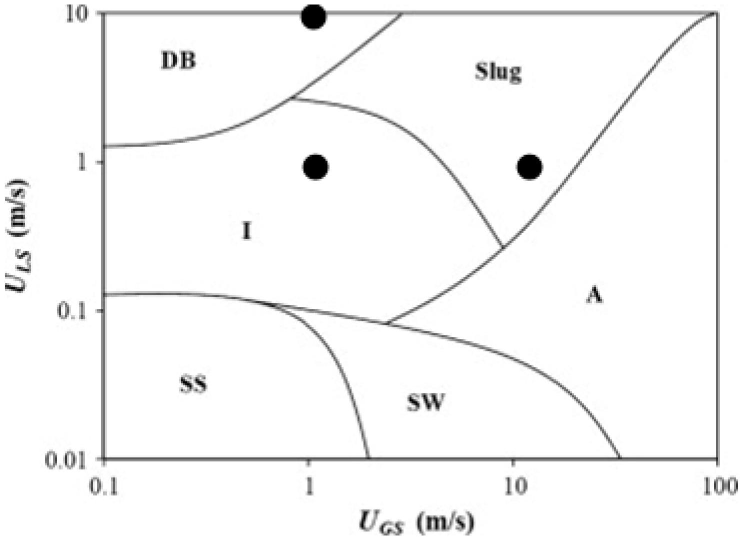


Figure 4.9. Taitel Dukler map and situation of the simulated cases in it. (DB): Dispersed bubbles. (Slug): Slug flow. (I): Intermittent flow. (A): Annular flow. (SW): Stratified wavy. (SS): Stratified Smooth

CASE PH_1

The first case simulated is the case where both phases have the same velocity in a horizontal 2D channel. The velocity then is set to 1 m/s. As it can be seen in the figure 4.10, the separation of the phases occurs and the flow just continues in this state throughout the computational domain. There is no mixing between the two phases and the interface is quite smooth. Of course, the air goes to the top and the water to the bottom, due to the density difference of the phases and the influence of gravity.

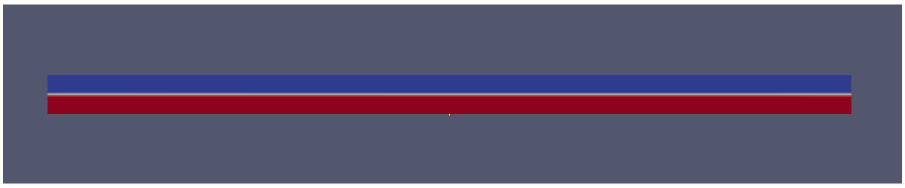


Figure 4.10. CASE PH_1 phase distribution at t=10s

Therefore, this case is concluded to be a stratified flow case, as it is expected since both phases have the same velocity.

The distribution of the pressure is normal as it can be seen in figure 4.11. It is a constant value for each phase and the abrupt change happens where the interface is due to the density change.

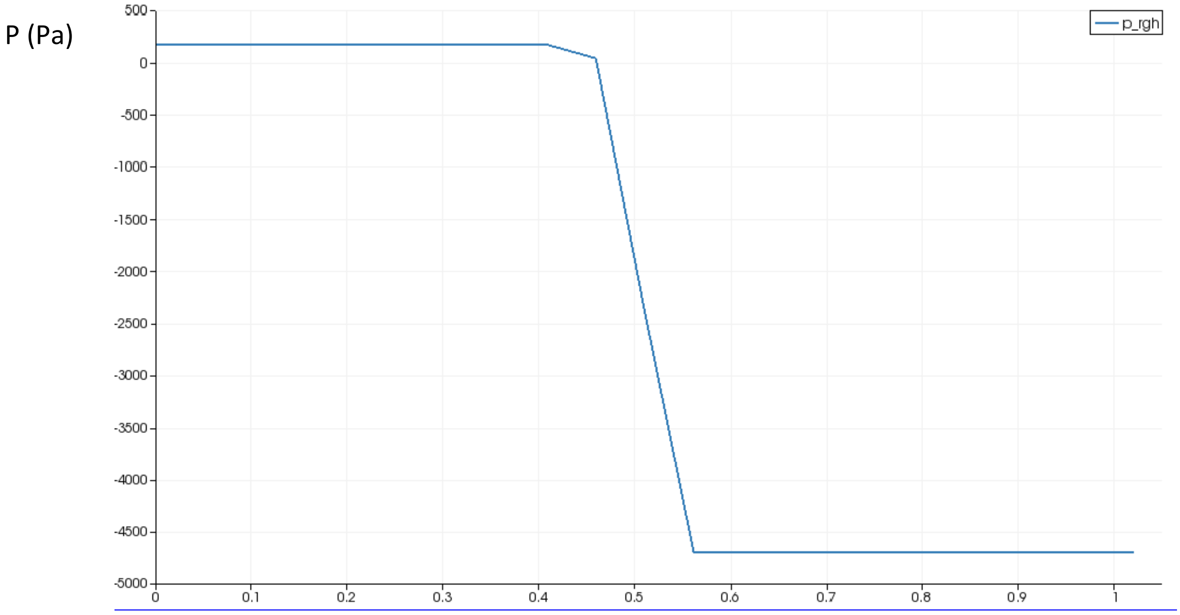


Figure 4.11 CASE PH_1 pressure distribution at t=10s and x=10m

PH10-1

In order to obtain the bubbly flow the velocity of the liquid phase has to be higher than the gaseous one. In this case the velocities for each of the phases were 10m/s for water and 1m/s for air. At low flow rates bubbles will gather at the top section of the closed channel due to buoyancy. But if the velocity keeps increasing the bubbles will uniformly distribute in the liquid. The size and the shape of the bubbles is small and spherical and larger ones too. If the bubbles get bigger they can deform and be wobbly.

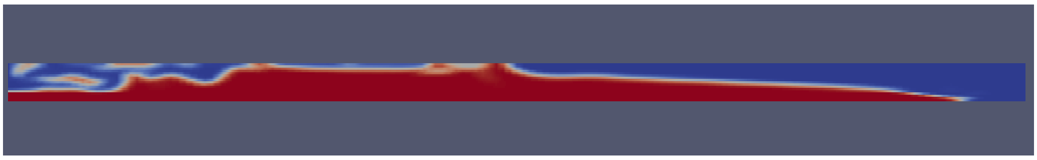


Figure 4.12 CASE PH10-1 phase distribution at t=3s

In figure 4.12/4.13 it can be seen how it is not a well-defined bubbly-flow but it is a transition to it, and some bubbles can be seen along the two parallel plates. They do not have a very spherical form but they are recognizable bubbly flow.

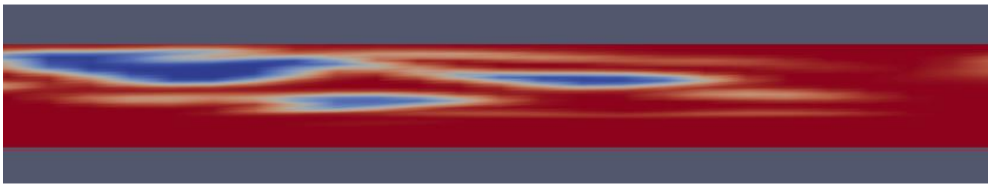


Figure 4.13 CASE PH10-1 phase distribution at t=5s and close to x=20m

PH15-1

The following case corresponds to a velocity of 15 m/s for the water and 1 for the air.

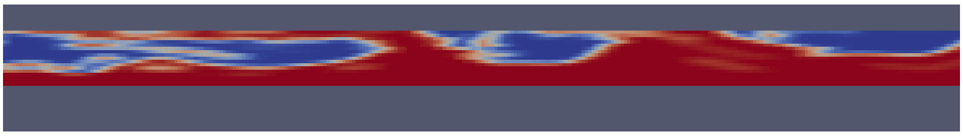


Figure 4.14 CASE PH15-1 phase distribution at t=10s

A bullet shape big bubble starts to develop, with a length of around 1-2 diameters and it goes to the top of the channel due to the gravity forces. The large bubbles are narrower than the ones that happen in the slug flow, and they sometimes lose their shape. Therefore, this can be considered as an intermittent flow.

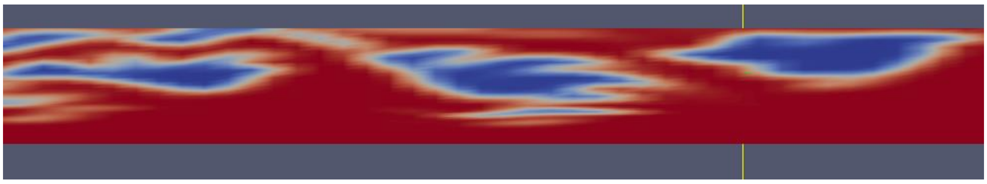


Figure 4.15 CASE PH15-1 phase distribution at t=11s and close to x=10m

PH1-10

When the velocity of the air is sufficiently high, the flow pattern obtained is the annular flow. In theory, the air flows in an inner-core and the water forms a film on the channel wall.

If the velocity of the air keeps increasing the water will shed of from the film layer and will entrain as droplets into the gas core. This regime is known as annular mist flow and there is a balance between the droplets that shed from the liquid layer and the bubbles coalescing at it.

In this case it has been used an horizontal 2D channel with inlet values of 1 m/s for the water phase and 10 m/s for the gas.

In this simulation it can be seen how the water tends to go to top and low walls creating a thin layer and leaves the middle part for the air.

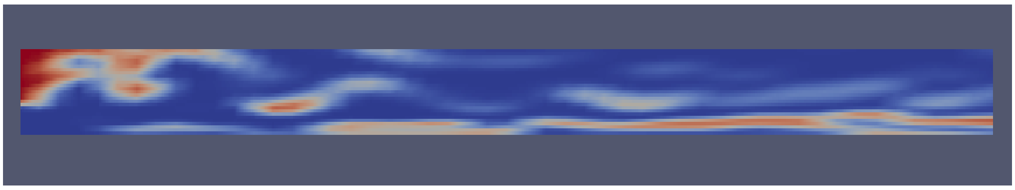


Figure 4.16 CASE PH1-10 phase distribution t=5s

Initially, the flow starts from a stratified position with all the air over the water layer but the turbulences created with the input velocities make the flow evolve to an unstable regime where both phases keep separated but in different positions.

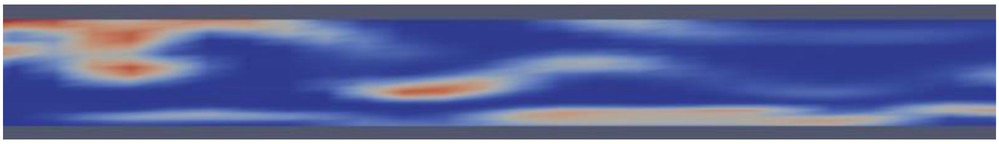


Figure 4.17 CASE PH1-10 phase distribution t=5.5s and close to x=3m

PV1-1

In this case vertical 2D channel of 100m long and 1m height with 1m/s inlet values is studied.

.

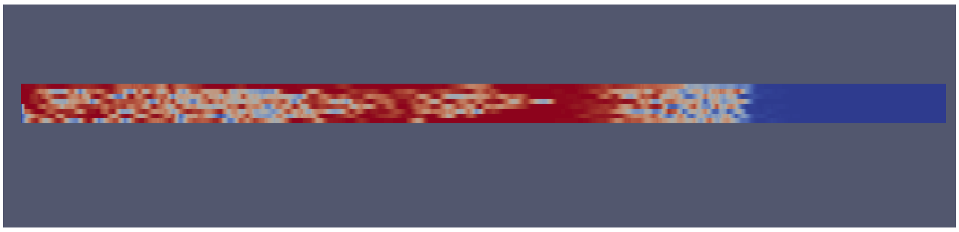


Figure 4.18 CASE PV1-1 phase distribution t=20s

The first thing that can be said about it is that, even if the velocities at the entries are the same, the behavior of the fluxes is completely different. As it can be seen previously in the PH1-1 case, the fluxes were stratified, but in this case they are not and they are in transition to an intermittent or even bubbly flow.

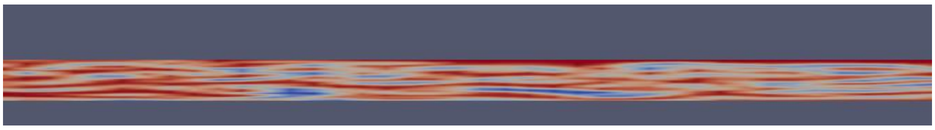


Figure 4.19 CASE PV1-1 phase distribution t=15s and close to x=20m

This difference is as a consequence of the effect of the gravity forces that are added to the pressure the inlet values make in the flow. This help both faces mix during the simulation as during the free fall the different density values each phase have are not that important.

PV10-1.5

In this simulation a 100m long and 1m height vertical 2D channel is run.

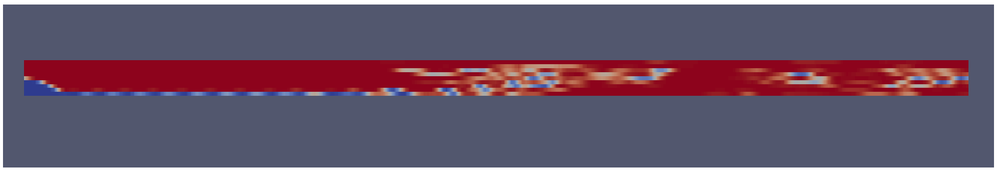


Figure 4.20 CASE PV10-1.5 phase distribution at t=9.5s

10 m/s and 1.5 m/s entry values have been used for water and liquid phases, respectively. Using these values, theoretically in a horizontal 2D channel a bubbly-flow should happen but this is a vertical 2D channel. The effect of the free fall plus the bigger water input makes the channel fill of water and the little air that remains makes different behaviors. It starts almost annular just being bubbly-flow at the end. It is clear that the flow is under transition and longer simulation time should be used to get a more descriptive results.

PV1-10

This simulation consists in a vertical 2D channel 100m x 1m x 0.2m with 1m/s and 10m/s initial inlet values for water and air.

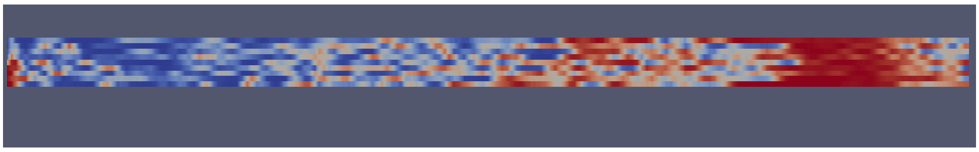


Figure 4.21 CASE PV1-10 phase distribution at t=9.5s

As it can be seen in the figure, the flow is not fully developed. The fluids change in different ways until an annular shape starts to be distinguished.

This form is the one that should be expected in a horizontal 2D channel but for a vertical one the behavior changes and it is less predictable.

PI3G

In this case it has been analyzed a 20m long 2D channel with an inclination of 3 degrees. The inlet velocity values are 1m/s for both phases.

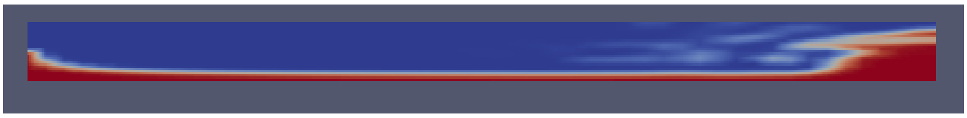


Figure 4.22 CASE PI3G phase distribution at t=13s

Both phases are initially separated and they stay this way until the end. The problem is that the liquid phase goes faster out of the channel and then it finishes only with a thin layer of water and a lot of air above.

PI5G

In this case and a 20m long inclined channel has been studied. The inclination is 5 degrees and the velocity inlet values are 1m/s.

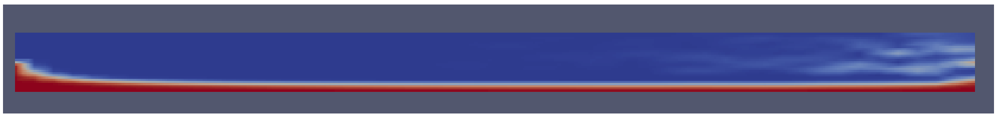


Figure 4.23 CASE PI5G phase distribution at t=20s

Similarly as happens in the previous case, the liquid phase moves faster than the air and that makes the channel finish full of air even if the inlet have the same area for both phases. Furthermore, there is some instability at the end of the closed channel and a ''cloud'' of water appears inside the air phase.

PI10G

Here it can be seen the simulation of a 10 degrees inclined 20 m long channel with 1m/s inlet values.

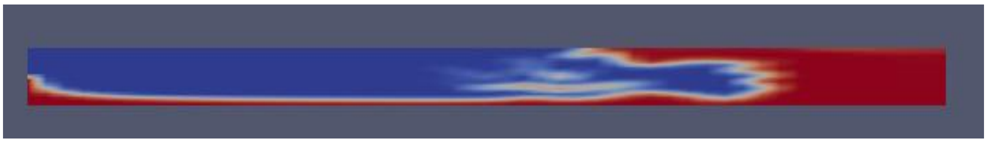


Figure 4.24 CASE PI10G phase distribution at t=4.6s

As consequence of the inclination the fluid moves fast and the water outlet is not big enough so, the water starts to collapse at the end of the channel and a wave in the opposite direction appears. This makes the flux instable but after few seconds the water at the end goes out of the channel and finally, most of the water leaves and almost nothing remains inside.

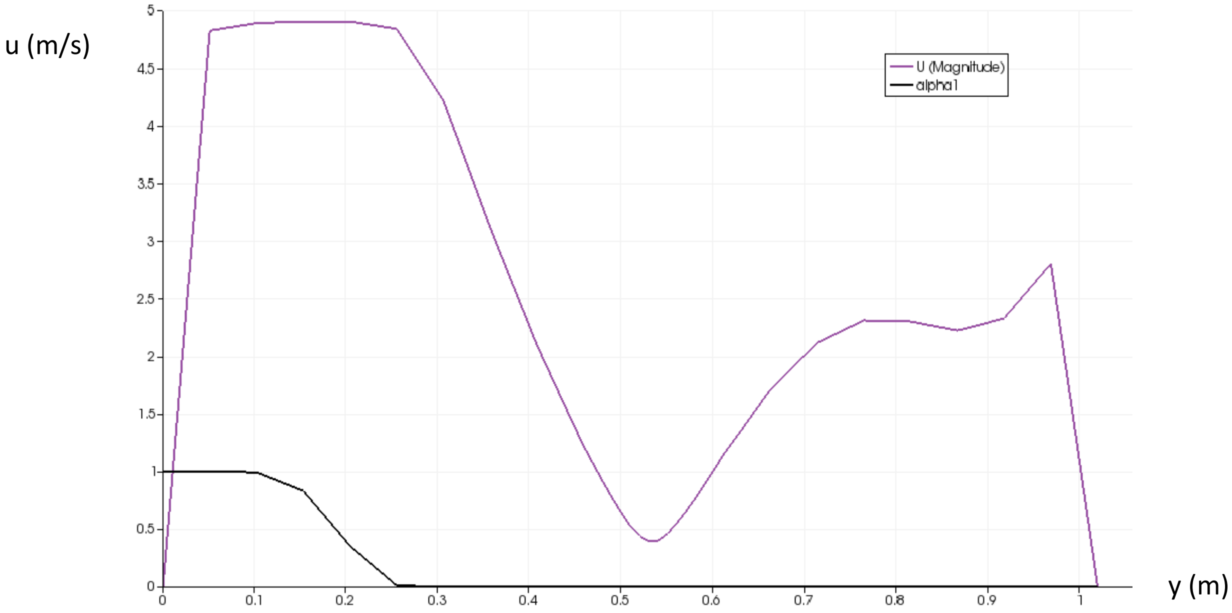


Figure 4.25 CASE PI10G velocity profile at t= 4s and x=10m

As it happens in the previous cases, the liquid phase is much faster than the air. As shown in figure 4.25.

5. CONCLUSIONS

The results obtained let us analyze the performance OpenFOAM has when working with simple multiphase flows.

For the open-channel flow, as the geometry is shallow, it is expected to find the maximum velocity close to the surface. As it can be seen in the figures describing the velocity profiles, the maximum velocity is always found close to the surface, and the velocity at the bottom is equal to zero fulfilling the no-slip condition. However, in this graphs it can be seen that the distributions are not parabolic, but turbulent. This may be due to the values of the entry length, L_e that the channels need in order to be a fully developed flow. The values for these entry lengths are way too big to be able to simulate without increasing the computational cost.

We are definitely analyzing an unsteady flow, because the depth varies both with time and space which is consistent with supercritical flows as the Froude number exceeds 1. This means that the flow velocity is higher than the wave velocity. The analogous situation in the gas motion would be the supersonic case. Here the information travels with the wave velocity. The values of the Froude numbers are not close to the critical value, which is often a dangerous area that designers should avoid.

The turbulent open-channel flows, should give some instabilities for Fr values bigger than one, showing some waves in the interface. However, the results given by OpenFOAM don´t show any kind of wave. This may be an effect of the low resolution, but again, more computational power would be needed to check it.

When it comes to the closed channel flow results, at the beginning the flow regimes were often not the expected ones because sometimes one of the phases disappeared from the channel or the distribution of the phases didn´t match any of the defined patterns. However, after several simulations and different velocity combinations, the results began to give more sense. These results are obviously not as good and precise as the ones that can be found on the web or in books, but due to the computational cost and the lack of time, more thorough studies could not be made.

The cases definitely have to be run using turbulence modeling because the velocities used here are big enough and the viscosity is low giving sufficiently high Reynolds numbers.

The Taitel Dukler maps are done by using pipes and the cases here are 2D closed channels so it is expected that the results don´t match perfectly, because the lack of this 3rd dimensions will most likely affect the results.

The results obtained for the horizontal closed channels quite match with the Taite-Dukler map. Here the superficial velocities for both phases are assumed to be the velocities given in the initial conditions to make it easy to set up and understand the phases. For the case PH_1 were the velocity for both phases is the same, a stratified pattern is obtained. The graph says instead that the flow should be stratified. However, if the theory part is read, it is clearly stated that if both phases have same low velocities the phases should separate. In this case then there is not match with the graph but yes with what the books say. In the case PH10-1 there is a correlation with the Taitel-Dukler map. It can be seen how when the velocity value for the gas is 1 and the water is 10 the graph takes us to a bubbly flow part. This case is not a crystal clear bubbly case but at least some bubbles can be observed. The next case, PH 15-1, gets out of the scope of the map, but it is definitely an intermittent flow. For the final horizontal closed channel case, PH 1-10, the point is not exactly in the annular flow region in the flow-pattern map but it is quite close to the annular region. This small change can be a consequence of some differences between the case set up in this study and the experiment case. It can be seen that the gas phase occupies the central area of the closed channel and that the water is situated around it. It is not spread in a uniform way but still the core of the closed channel is gas. The water is mainly concentrated in the lower part of the channel and this is due to the effect of the gravity.

The horizontal closed channels show some patterns that are difficult to identify but are definitely affected by gravity forces and act to stratify the heaviest phase to the bottom and the gaseous to the top.

The results obtained for the inclined closed channels don´t show any kind of useful results. What happened with the inclined closed channels and with some other cases is that the domain used to get empty from water with no apparent reason. Even if the entry of water was set up in the same way for all the cases, some of them used to have a continuous entry of water and some others used to get empty. It was not possible to find why this happened but luckily some of the cases worked well. However, it can be read in the theory that certain regimes only exist in certain inclination intervals, so maybe these combinations were not found when trying to create some patterns for inclined channels.

It has been quite complicated to check the results obtained, especially in open-channel cases.
The velocity profiles in this type of cases were not completely developed and it was impossible to check with enough precision using the data from the analytical solutions. It can easily be seen how the fluids behave as they should but the used simulation sizes limit the full evolution and it is hard to check a developing velocity profile with an analytical fully developed one. As result of this situation, OpenFOAM’s values are lower than expected; but probably with more computational time and weight, those values could be improved.

Even if the solutions have not been perfectly evaluated, the evolution of both fluids indicated a correct behavior and the given values are inside a logical interval. So, it could be concluded that open-channel simulations were properly done even if it is not possible to demonstrate it with 100% accuracy.

After some months using OpenFOAM as a CFD tool for solving engineering fluid mechanics problems it can be concluded that the software is a valid tool that offers the possibility of getting acquainted with the way CFD works. By using OpenFOAM the user can learn how to develop a fluid mechanic problem from the initial stage (creating the geometry, the mesh etc.) to the very end, and also choose the numerical models to be used as well as wide range of choices in other aspects of the simulation. However, it has to be said that the interface is not as easy-to-use as in other CFD packages. After working with other softwares like Star CCM+ , the way OpenFOAM works is a bit more difficult to use and the user has to have basic knowledge about programming.

Overall, OpenFOAM is a very useful tool both for the academic and professional background, as it has a very importance advantage, being an open-source software. It may not be as fancy looking or easy to use program compared to other CFD tools out there in the market (Star CCM+), but it is definitely a fact to be taken in mind whether it is more worth it paying tremendous amounts of money for the license or getting used to the software. In the current world where one of the biggest goals for the companies is reducing the cost of the operation while keeping the quality of the output untouched, OpenFOAM seems to be a very interesting alternative to the commonly used and very expensive CFD packages.

6. FUTURE WORK

If the study is to be continued there are some aspects that can be improved or more deeply studied.

Firstly, there is always the opportunity to refine the mesh so that the solution is more accurate, and in this case where there are different phases, the interface between them would be more sharply defined if the size of the cells is smaller. Of course, the refinement of the mesh implies an increase in the computational cost and this is one of the reasons why the grid size is not very small. Here, as the geometry of the case and the distribution of the phases are not very complicated this seems to be enough. Also, the space that the simulations occupied could not be bigger than 8GB more or less, so the cases had to be limited in a way. Running a case in parallel could also be advantageous if the mesh size increases sufficiently.

In relation to the refinement of the interface, this is never sharply defined in interFoam. It always occupies a volume instead of being a sharp interface. If the interface wants to be compressed it is possible to introduce a modified transport equation in interFoam. If this wants to be done, it is necessary to add a compression term in the transport equation, and then the interface should maintain clearer.

Another improvement that can be made is to change from 2-Dimensional to 3-Dimensional simulations. It is always more realistic to make a simulation in 3-D, because making some assumptions to simplify the case to a 2D can make the case lose accuracy in the results, especially when they are to be compared with experimental results. Also, the Taitel Dukler maps are done using actual pipes in three dimensions, so if the simulations are done in 3D, the results will match better map.

Here, like in most of the simulations, a turbulence model is adopted. However, if someone wants to directly solve the turbulence equations without any approximation the Direct Numerical Simulation can be used. In the Direct Numerical Simulation there is no approximation or averaging and they can solve three dimensional dynamic and time dependent Navies Stokes equations with the help of a supercomputer. However, the DNS is the most time consuming and computationally costly approach, so it has to be considered if this expense is worth the work or not.

Also cyclic boundary conditions could be used in a future work. There were used for this project but the obtained results were far from expected ones. Only simulations with very stable conditions could have been useful as the rest were crashing very soon or giving senseless results that did not respect the laws of physics.
Despite all these problems, cyclic boundary conditions could be used for stable and simple simulations with a more refined mesh and being careful the rest of boundaries. Even more, if it works with this type of boundaries the results would be much better and easily evaluable as velocity profiles would develop completely, and the comparison with analytical values could be done.

7. REFERENCES

[1] CHANSON, H. (2004). HYDRAULICS OF OPEN CHANNEL FLOW. AN INTRODUCTION (2ND EDITION). BUTTERWORTH-HEINEMANN.

[2] CROWE, C. T., ELGER D.F., WILLIAMS B.C. & ROBERSON J.A. (2010). ENGINEERING FLUID MECHANICS. (9TH EDITION). JOHN WILEY & SONS, INC.

[3] DORAO, C. (2012). MULTIPHASE FLOW. LECTURE NOTES FOR COURSE TEP4545.

[4] DOUGLAS, J.F., GASIOREK, J.M., SWAFFIELD, J. A. & JACK, L.B. (2005). FLUID MECHANICS, HARLOW, PERSON PRENTICE HALL.

[5] FERNANDINO. M. (2008). TEP23-APPLIED COMPUTATIONAL FLUID DYNAMICS. MULTIPHASE FLOW MODELING. LECTURE NOTES.

[6] FINNEMORE, E. J., FRANZINI, J.B.& DAUGHERTY, R.L. (2002). FLUID MECHANICS WITH ENGINEERING APPLICATION. BOSTON. MCGRAW-HILL.

[7] OPENFOAM FOUNDATION.(2011). FEATURES .ONLINE: HTTP://WWW.OPENFOAM.COM/FEATURES/

[8] OPENFOAM FOUNDATION.(2011). MULTIPHASE FLOW SOLVERS. ONLINE: HTTP://WWW.OPENFOAM.COM/FEATURES/STANDARD-SOLVERS.PHP#MULTIPHASEFLOWSOLVERS

[9] RUSCHE, H.(2002). COMPUTATIONAL FLUID DYNAMICS OF DISPERSED TWO-PHASE FLOWS AT HIGH PHASE FRACTIONS. INTERPHASE CAPTURING APPROACH: P. 147-178. ONLINE:

http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs/HenrikRuschePhD2002.pdf

[10] THOME, J.R.(2010).Engineering data book III. Chapter12:Two-Phase Flow Patterns. Pg. 3-4. Online: http://www.wlv.com/products/databook/db3/DataBookIII.pdf

[11] VERSTEEG, H.K. & MALALASEKERA W. (1995).An introduction to computational fluid dynamics. The finite volume method. Chapter 6.

[12] WHITE, F.M. (2003). Fluid Mechanics (5th edition). McGraw-Hill.

[13] OPENFOAM FOUNDATION. (2011). Users guide. Online: http://www.openfoam.org/docs/user/

[14]lucchini, t. OpenFOAM postprocessing and advanced running options. Department of Energy Politecnico di Milano. Online: http://web.student.chalmers.se/groups/ofw5/Basic_Training/ofPostProcessingLucchini.pdf

[15] OPENFOAM FOUNDATION.(2011). turbulence.Online: http://www.openfoam.com/features/turbulence.php

[16]OPENFOAM FOUNDATION. (2011). boundaries. online: http://www.openfoam.org/docs/user/boundaries.php

[17]CIVE 2400: Fluid mechanics section 2. Open channel hydraulics. School of Mechanical Engineering. University of Leeds Online: http://www.efm.leeds.ac.uk/CIVE/CIVE2400/OpenChannelHydraulics2.pdf

[18]CIVE 2400: Fluid mechanics section 1. Fluid flow in pipes. School of Mechanical Engineering. University of Leeds Online: http://www.efm.leeds.ac.uk/CIVE/CIVE2400/pipeflow2_2008a.pdf

Appendix A: blockMeshDict

```
/*------------------------------*- 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;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//

convertToMeters 1;

vertices
(
    (0 0 -0.1)
    (20 0 -0.1)
    (20 0.5 -0.1)
    (0 0.5 -0.1)
    (0 0 0.1)
    (20 0 0.1)
    (20 0.5 0.1)
    (0 0.5 0.1)
    (0 1 -0.1)
    (20 1 -0.1)
    (20 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (50 10 1) simpleGrading (1 1 1)
    hex (3 2 9 8 7 6 10 11) (50 10 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    inlet_air
    {
        type patch;
        faces
        (
            (3 7 11 8)
        );
    }
    inlet_water
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet_air
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    outlet_water
    {
        type patch;
        faces
        (
            (2 9 10 6)
        );
    }
    lowerWall
    {
        type wall;
        faces
        (
            (0 1 5 4)
        );
    }
    topWall
    {
        type wall;
        faces
        (
            (9 8 11 10)
        );
    }
);

mergePatchPairs
(
);

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

Appendix B: Boundary conditions

Closed channel cases

Alpha

```
/*--------------------------*- 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       volScalarField;
    object      alpha;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet_water
    {
         type           inletOutlet;
         inletValue     uniform 1;
         value          uniform 1;
    }
    inlet_air
    {
         type           inletOutlet;
         inletValue     uniform 0;
         value          uniform 0;
    }

    outlet_water
    {
        type            zeroGradient;
    }
    outlet_air
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }
    topWall
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

Epsilon

```
       /*--------------------------*- C++ -*--------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*-------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * ** * * * * * * * * * * * * * * //

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

internalField   uniform 0.00026775;

boundaryField
{
   inlet_water
    {
        type            fixedValue;
        value           uniform 0.000115641815387;
    }
   inlet_air
    {
        type            fixedValue;
        value           uniform 0.000310274047353;
    }

   outlet_water
    {
        type            zeroGradient;
    }
    outlet_air
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            epsilonWallFunction;
        value           uniform 0.00001498;
    }

    topWall
    {
        type            epsilonWallFunction;
        value           uniform 0.00001498;
    }
   defaultFaces
    {
        type            empty;
    }
}

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

K

```
/*--------------------------*- C++ -*-----------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0.00375;

boundaryField
{
    inlet_water
    {
          type            fixedValue;
        value           uniform 0.001343860036447;
    }
    inlet_air
    {
          type            fixedValue;
        value           uniform 0.002594823002503;
    }
    outlet_water
    {
        type            zeroGradient;
    }
    outlet_air
    {
        type            zeroGradient;
    }
    lowerWall
    {
        type            kqRWallFunction;
        value           uniform 0.002535;
    }
    topWall
    {
        type            kqRWallFunction;
        value           uniform 0.002535;
    }
    defaultFaces
    {
        type            empty;
    }
}

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

nut

```
/*--------------------------*- C++ -*----------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    inlet_water
    {
        type            calculated;
        value           uniform 0;
    }
    inlet_air
    {
        type            calculated;
        value           uniform 0;
    }

    outlet_water
    {
        type            calculated;
        value           uniform 0;
    }
    outlet_air
    {
        type            calculated;
        value           uniform 0;
    }
    lowerWall
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    topWall
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    defaultFaces
    {
        type            empty;
        }
    }

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

nuTilda

```
/*---------------------------*- C++ -*-------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*-------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * *  * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    inlet_water
    {
          type            fixedValue;
          value           uniform 0;
    }
    inlet_air
    {
          type            fixedValue;
          value           uniform 0;
    }

    outlet_water
    {
        type            zeroGradient;
    }
    outlet_air
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }

    topWall
    {
        type            zeroGradient;
    }



    defaultFaces
    {
        type            empty;
    }
}

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

p_rgh

```
/*----------------------------*- 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       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    inlet_water
    {
        type            zeroGradient;
    }
    inlet_air
    {
        type            zeroGradient;
    }

    outlet_water
    {
        type            zeroGradient;
    }

    outlet_air
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }
    topWall
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

U

```
/*--------------------------------*- 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       volVectorField;
    location    "0";
    object      U;

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

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

internalField   uniform (1 0 0);

boundaryField
{
    inlet_water
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    inlet_air
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    outlet_water
    {
        type            zeroGradient;
    }
    outlet_air
    {
        type            zeroGradient;
    }
    lowerWall
    {
        type            fixedValue;
        value           uniform (0.0 0 0);
    }
    topWall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    defaultFaces
    {
        type            empty;
    }
}


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

2. Open-channel

Alpha

```
/*--------------------------------*- 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       volScalarField;
    object      alpha;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    leftWall1
    {
      //  type            constantValue;
         type           inletOutlet;
         inletValue     uniform 1;
         value          uniform 1;
    }
    leftWall2
    {
    //    type            constantValue;
         type           inletOutlet;
         inletValue     uniform 0;
         value          uniform 0;
    }

    rightWall1
    {
        type            zeroGradient;
    }
    rightWall2
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }
    topWall
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

p_rgh

```
/*--------------------------------*- 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       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    leftWall1
    {
        type            zeroGradient;
    }
    leftWall2
    {
        type            zeroGradient;
    }

    rightWall1
    {
        type            zeroGradient;
    }

    rightWall2
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }
    topWall
    {
        type            totalPressure;
        p0              uniform 0;
        U               U;
        phi             phi;
        rho             rho;
        psi             none;
        gamma           1;
        value           uniform 0;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

U

```
/*--------------------------------*- 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       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    leftWall1
    {
        type            zeroGradient;
    }
    leftWall2
    {
        type            zeroGradient;
    }
    rightWall1
    {
        type            zeroGradient;
    }
    rightWall2
    {
        type            zeroGradient;
    }
    lowerWall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    topWall
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
    defaultFaces
    {
        type            empty;
    }
}


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

Appendix C: transportProperties

```
/*--------------------------------*- 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;
}
// * * * * * * * * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * //

phase1
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1e-06;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1000;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }

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

phase2
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1.48e-05;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }

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

sigma           sigma [ 1 0 -2 0 0 0 0 ] 0.07;


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

Appendix D: g

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    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 -9.8 0 );


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

Appendix E: turbulenceProperties and RASProperties

Turbulence properties

Open-channel cases

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;


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

Closed channel cases

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


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

RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    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;


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

Appendix F: controlDict

```
/*--------------------------------*- 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      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * //

application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         10;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   0.05;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.5;
maxAlphaCo      0.5;

maxDeltaT       1;


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

Appendix G: FvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    pcorr
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-10;
        relTol          0;
    }

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

    p_rghFinal
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0;
    }

    "(U|k|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon)Final"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor no;
    nCorrectors     3;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
    nAlphaCorr      1;
    nAlphaSubCycles 4;
    cAlpha          2;
}

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

Appendix H: FvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    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;
}

divSchemes
{
    div(rho*phi,U)  Gauss linear;
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss interfaceCompression;
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

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


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

APPENDIX I: SIMPLE

SIMPLE stands for Semi Implicit Method for Pressure-Linked Equations [11]. The algorithm was originally put forward by Patankar and Spalding (1972) and is essentially a guessed-and correct procedure for the calculation of pressure on the staggered grid. The staggered grid uses scalar variables such as pressure, density, temperature etc. at ordinary nodal points but the velocity components on staggered grids are centered on cell faces. This is because if the velocities are defined at the scalar grid nodes, the influence of pressure is not properly represented in the discretized momentum equations.
The PISO algorithm, which stands for Pressure Implicit with Splitting of Operators, of Issa (1986) is a pressure-velocity calculation procedure developed originally for non-iterative computation of unsteady compressible flows. However, it has been adapted successfully for the iterative solution of steady state problems.

The first step is to solve the discretized momentum equations with a guessed pressure field *p**, to yield the velocity components *u** and *v*.*
However, these velocities do not satisfy continuity. Instead, these velocities are defined,

$$u = u^* + u'$$
$$v = v^* + v'$$
$$p = p^* + p'$$

These equations are developed and introduced into the discretized continuity equation to yield the pressure correction equation. Once it is solved the field of *p´* is obtained, and when this is known the velocity fields of *u* and *v* can be obtained.
After, a new pressure is introduced where the guessed value is now the correct value of the pressure explained before.

$$p_c = p + p''$$

Where p'' is the second pressure correction.
The twice corrected velocity field equations may be obtained by solving the momentum equations once more. The substitution of these equations in the discretized continuity

equations yields a second pressure correction equation. With this equation the pressure correction is obtained and so, also the twice corrected pressure field and velocity values. However, under-relaxation is needed with the above procedure to stabilize the calculation process. If under relaxation is to be sued a new factor has to be introduces in the pressure correction equation

$$p = p^* + \alpha p'$$

Where α is the pressure under relaxation factor
Although this method implies a considerable increase in computational effort it has been found to be efficient and fast. If has a value of 1 the guessed pressure field p* is corrected by p´. However, the p´ is usually very large for stable computations. When α is equal to 0 would apply no correction at all. Then, if this value is between 0 and 1, a fraction of the correction field is added to the guessed pressure field, which is small enough to ensure stability and big enough to move the iterative improvement process forward.
The velocities are also under relaxed, but this is not covered here.
A correct choice of under relaxation value is essential for cost-effective simulations. A value too large could cause oscillatory or divergent solutions and a too small would cause extremely slow convergence. Unfortunately, the optimum values are not tabulated and vary from case to case.