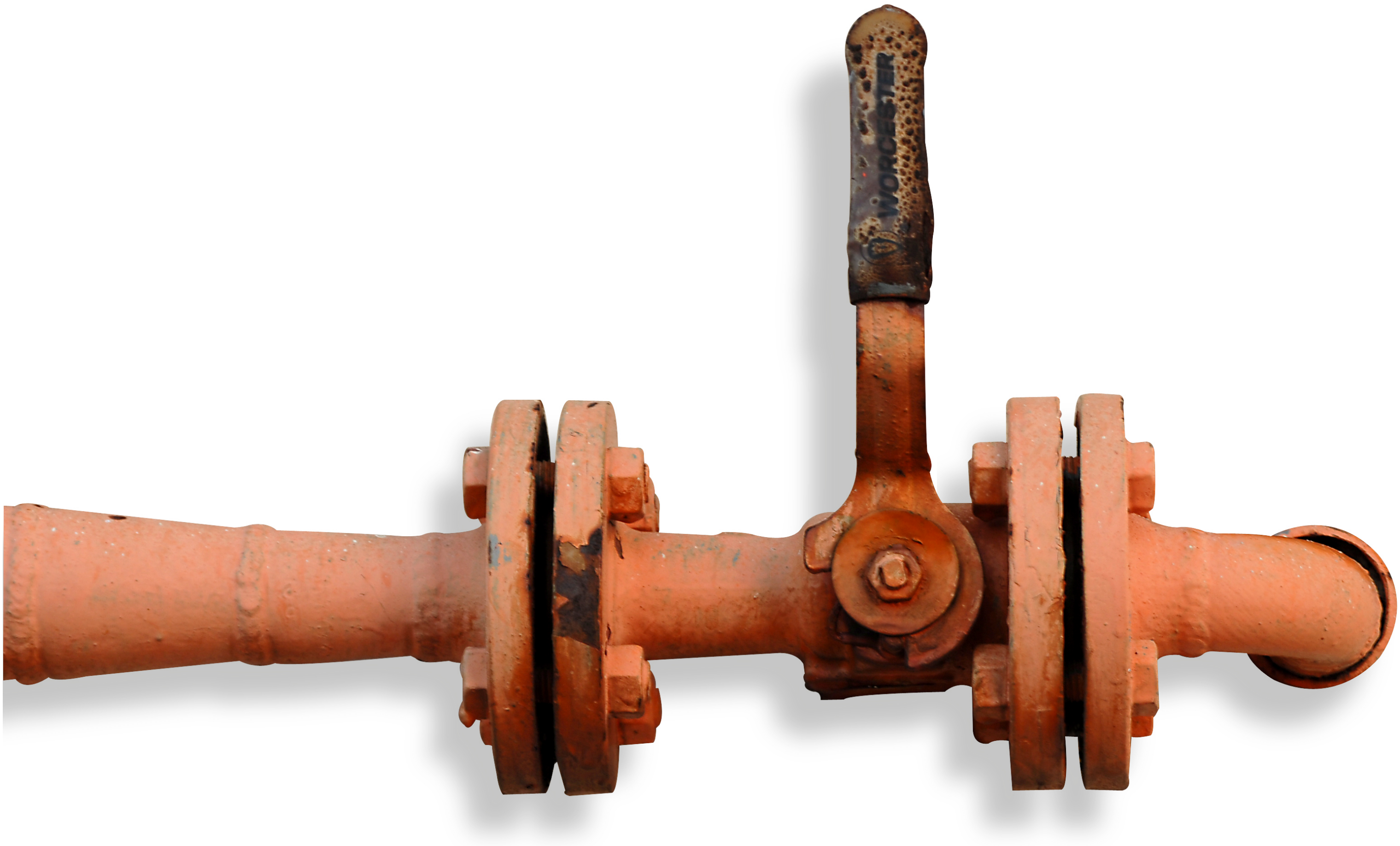
CFD of multiphase pipe flow: a comparison of solvers

Peeters, Pim Tomas

Track: Aerodynamics

December 20, 2016



CFD of multiphase pipe flow: a comparison of solvers

Peeters, Pim Tomas

Track: Aerodynamics

December 20, 2016

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on December 20, 2016

Student number:	4007956
Project duration:	March 2016 - December 2016
Thesis committee:	Dr. ir. B.W. van Oudheusden
	Dr. ir. A.H. van Zuijlen
	Prof. dr. ir. R.A.W.M. Henkes

SUMMARY

Pipelines with multiphase flow of gas, oil, and water are commonly used in the oil and gas industry. In the presence of offshore platforms or vessels, the pipeline ends with a vertical riser. A possible new concept is to use a single large diameter pipeline along the sea floor that ends into multiple smaller diameter risers. This concept might be of interest for future Floating Liquefied Natural Gas (FLNG) vessels. This report focusses on detailed numerical simulations for a small scale representation of this industrial flow splitting configuration: a two-phase flow of air and water though a 0.5 [m] long, 0.05 [m] diameter, horizontal pipe with a T-junction to a dual 2.5 [m] high, 0.05 [m] diameter riser system.

A comparison has been made between two multiphase flow solvers available in the well-known open-source code OpenFOAM. The `interFoam` solver utilises a mixture model formulation, while the `multiphaseEulerFoam` solver uses an Euler-Euler (two continua) formulation for both fluids. Both solvers use Volume of Fluid as interface sharpening method to solve the equations for incompressible flow. Air and water are flowing into the domain with a total volumetric flow rate of 64.2 [m^3 h^{-1}]. The volumetric flow rate of water in the simulations is taken as 1, 2 and 3 [m^3 h^{-1}]. Furthermore, a pressure difference between the outlets of the dual risers is applied. This is meant to analyse the difference between solutions of both solvers with respect to maldistribution between the two risers, pressure loss and liquid hold-up.

The Pressure Implicit with Splitting of Operator (PISO) algorithm with two corrector loops is used combined with a low Courant-Friedrichs-Lewy condition of 0.25 to obtain sufficient numerical stability. Second order discretisation schemes in space and a first order scheme in time are used. In order to speed up the calculations Geometric Algebraic Multigrid is used to solve the pressure field.

Meshing of the domains is done with `snappyHexMesh`. Three T-junction meshes are generated with increasing fineness of 23042, 47544 and 95292 cells. Two riser meshes are made with 67720 and 276784 cells. The Large Eddy Simulation turbulence model is used with the Smagorinksy Sub-Grid Scale model. The fixed flow rates at the inlet are coupled with a variable pressure. At the outlets the outflow velocity is variable with a fixed pressure. The turbulent viscosity at the wall is governed by wall functions. The calculations are done on the `hpc12` cluster at Delft University of Technology. The calculation times are in the order of one to three weeks for a typical simulation.

Overall, differences between the production at the outlets of `interFoam` and `multiphaseEulerFoam` are small. Solutions from both solvers indicate that the influence of a pressure difference over the outlets has more influence on the non-symmetric production of air than of water. The results from `multiphaseEulerFoam` look promising and due to its Euler-Euler momentum description it shows realistic flow behaviour in the junction. Improvements in simulating the system can be made by progressing the simulations longer in time, by increasing the entrance length of the horizontal pipe towards the junction and by choosing another Sub-Grid Scale model.

Acknowledgements

Hereby I present my master's thesis about CFD simulations of multiphase pipe flow. This is the result of seven months hard work in the topics of multiphase flow and OpenFOAM solvers, with which I was not familiar before. I am very proud of the report and the invaluable knowledge gained during this project.

Foremost, I want to express my sincere gratitude to my thesis supervisors dr. ir. Alexander van Zuijlen and prof. dr. ir. Ruud Henkes for their guidance during this research. Their support, ideas and feedback greatly helped me during this project. Also I want to express my gratitude to DRG for providing me with help and their facilities during this research.

Furthermore, I am grateful for my family for their indispensable support during my studies in Delft. Finally, I want to express my gratitude to all of my friends and especially Floor, for their and her encouragement during the writing of this thesis.

Pim Peeters
Delft, 6th December 2016

Contents

Acronyms

CAD	Computer-Aided Design
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
CLSVOF	Coupled Level Set Volume of Fluid
DNS	Direct Numerical Simulation
FLNG	Floating Liquefied Natural Gas
FOAM	Field Operations and Manipulation
GAMG	Geometric Algebraic Multigrid
GroovyBC	Groovy Boundary Condition
GUI	Graphical User Interface
LES	Large Eddy Simulation
LVIRA	Least Squares Volume of Fluid Interface Reconstruction Algorithm
MULES	Multidimensional Universal Limiter with Explicit Solution
PDE	Partial Differential Equation
PIMPLE	Merged PISO-SIMPLE
PISO	Pressure Implicit with Splitting of Operator
RANS	Reynolds-Averaged Navier-Stokes
SGS	Sub-Grid Scale
SHM	`snappyHexMesh`
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SLIC	Simple Line Interface Calculation
STL	STereoLithography
SWAK	SWiss Army Knife
TVD	Total Variation Diminishing
VOF	Volume of Fluid
WALE	Wall-Adapting Local Eddy

Nomenclature

$\partial\Omega$	[-]	Boundary of computational domain
Ω	[-]	Computational domain
α	[-]	Volume fraction of fluid
$\nabla\alpha$	[m^{-1}]	Gradient of phase fraction
ϵ	[$\mathrm{m}^2\ \mathrm{s}^{-3}$]	Dissipation rate of turbulent kinetic energy
η	[m]	Kolmogorov length scale
η_t	[s]	Ratio between wall clock time and simulation time
κ	[rad m^{-1}]	Turbulent eddy wave number
κ	[-]	von Kármán constant
λ	[kg $\mathrm{m}^{-1}\ \mathrm{s}^{-1}$]	Bulk viscosity
μ	[kg $\mathrm{m}^{-1}\ \mathrm{s}^{-1}$]	Dynamic viscosity
$\overline{\mu}$	[kg $\mathrm{m}^{-1}\ \mathrm{s}^{-1}$]	Averaged dynamic viscosity `interFoam`
ν	[$\mathrm{m}^2\ \mathrm{s}^{-1}$]	Kinematic viscosity
ν_t	[$\mathrm{m}^2\ \mathrm{s}^{-1}$]	Turbulent viscosity
$\tilde{\nu}$	[$\mathrm{m}^2\ \mathrm{s}^{-1}$]	Modified turbulent viscosity
ψ	[-]	Blending parameter for time discretisation schemes
ρ_0	[kg m^{-3}]	Density of incompressible fluid
ρ	[kg m^{-3}]	Density
$\overline{\rho}$	[kg m^{-3}]	Averaged density in `interFoam`
$\tilde{\rho}$	[kg m^{-3}]	Averaged density
$\boldsymbol{\sigma}$	[Pa]	Stress tensor
σ	[N m^{-1}]	Surface tension
$\boldsymbol{\tau}$	[Pa]	Deviatoric stress tensor
τ_η	[s]	Kolmogorov time scale

τ_{ij}	[Pa]	Turbulence Reynolds stresses
τ_w	[Pa]	Wall shear stress
φ	[-]	Arbitrary scalar
$\phi_{ll}(r)$	[-]	Limited linear limiter
$\phi_{vl}(r)$	[-]	Van Leer limiter
$\boldsymbol{A}$	[-]	Arbitrary matrix (in matrix solving example)
A	[m^2]	Pipe area
C^+	[-]	Integration constant in law of the wall
C_α	[-]	Interface compression coefficient
C_D	[-]	Drag force coefficient
C_L	[-]	Lift force coefficient
C_{vm}	[-]	Virtual mass force coefficient
C_f	[-]	Wall friction coefficient
Co	[-]	Courant number
E	[-]	Integration constant in law of the wall
$E(\kappa)$	[$\mathrm{m}^3\ \mathrm{s}^{-2}$]	Energy spectrum of turbulent flow
$\boldsymbol{F}$	[N]	Force vector
$\boldsymbol{F}_{b,k}$	[N]	Interphase force transfer vector of phase k
$\boldsymbol{F}_{d,k}$	[N]	Drag force transfer vector of phase k
$\boldsymbol{F}_{ext}$	[N]	External force
$\boldsymbol{F}_k$	[N]	Interfacial force between fluids
$\boldsymbol{F}_{l,k}$	[N]	Lift force transfer vector of phase k
$\boldsymbol{F}_{o,k}$	[N]	Other forces transfer vector of phase k
$\boldsymbol{F}_{s,k}$	[N]	Surface tension transfer vector of phase k
$\boldsymbol{F}_{visc}$	[N]	Viscous force vector
$\boldsymbol{F}_{vm,k}$	[N]	Virtual mass force transfer vector of phase k
$G(t,\boldsymbol{x})$	[-]	Filter kernel for Large Eddy Simulation
$H(x)$	[-]	Heaviside step function
$\boldsymbol{I}$	[-]	Unit tensor / identity matrix
$\mathcal{L}$	[m]	Characteristic length scale
L	[m]	Length of pipe

N	[-]	Nodal point of neighbouring cell
P	[-]	Nodal point of cell
$\dot{Q}$	[m^3 h^{-1}]	Volumetric flow rate
R_1	[m]	First characteristic radius of curvature
R_2	[m]	Second characteristic radius of curvature
Re	[-]	Reynolds number
$R_{k,p}$	[-]	Phase ratio of fluid k at patch k
$\boldsymbol{S}_{ij}$	[-]	Strain rate tensor
$\boldsymbol{S}_{ij}^d$	[-]	Alternative strain rate tensor
S	[-]	Boundary of computational domain
$\mathcal{U}$	[m s^{-1}]	Characteristic velocity scale
V	[m^3]	Volume
V_{cell}	[m^3]	Cell volume
Y_k	[-]	Normalised hold-up of fluid k
$\boldsymbol{b}$	[-]	Arbitrary vector (in matrix solving example)
c	[-]	Continuous phase (as subscript)
c_s	[-]	Smagorinsky constant
d	[-]	Dispersed phase (as subscript)
d	[m]	Diameter of pipe
d_{32}	[m]	Sauter mean diameter
d_k	[m]	Characteristic diameter of bubble or droplet of phase k
$\boldsymbol{f}$	[N m^{-3}]	Force vector per unit volume
f	[-]	Fanning friction factor
$\boldsymbol{g}$	[m s^{-2}]	Gravity vector
g	[m s^{-2}]	Gravitational acceleration
g	[-]	Gas phase (as subscript)
h_l	[m]	Liquid height
k	[m^2 s^{-2}]	Turbulent kinetic energy
k	[-]	Phase indicator (as subscript)
l	[-]	Liquid phase (as subscript)
m	[kg]	Mass

$\boldsymbol{n}$	[-]	Normal vector
p	[Pa]	Pressure
Δp	[Pa]	Pressure difference (commonly between the outlets)
Δp_{riser}	[Pa]	Pressure drop in the riser
r	[m]	Radius
t	[s]	Time
Δt	[s]	Time step
Δt_u	[s]	User defined maximum time step
t_{step}	[s]	Wall clock time per time step
$\boldsymbol{u}$	[m s^{-1}]	Velocity vector
$\overline{\boldsymbol{u}}$	[m s^{-1}]	Averaged velocity vector in `interFoam`
u^+	[-]	Wall velocity
u_τ	[-]	Wall friction velocity
u_1	[m s^{-1}]	Velocity in direction x_1
u_2	[m s^{-1}]	Velocity in direction x_2
u_3	[m s^{-1}]	Velocity in direction x_3
u_η	[m s^{-1}]	Kolmogorov velocity scale
$\boldsymbol{u}_\Omega$	[m s^{-1}]	Velocity vector of control volume
$\boldsymbol{u}_c$	[m s^{-1}]	Interface compression velocity vector
$\boldsymbol{x}$	[m]	Position vector
$\boldsymbol{x}$	[-]	Solution vector (in matrix solving example)
x	[m]	First normal direction in space, alias of x_1
x_1	[m]	First normal direction in space
x_2	[m]	Second normal direction in space
x_3	[m]	Third normal direction in space
Δx	[m]	Mesh spacing
$\widetilde{\Delta x}$	[m]	Typical length scale of cell
y	[m]	Second normal direction in space, alias of x_2
y^+	[-]	Wall distance
y_1	[m]	First nodal distance to the wall
z	[m]	Third normal direction in space, alias of x_3

List of Figures

List of Tables

Chapter 1

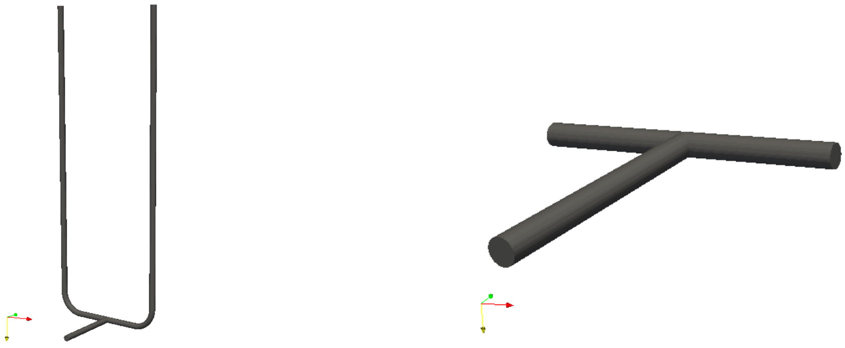
Introduction

Multiphase flows are present in numerous engineering applications, such as nuclear facilities, dredging and hydrocarbon pipe flow. For the latter, a mixture of oil, natural gas, water and solids can be present in the system at any time. The pressure in ageing oil or gas reservoirs is decreasing [19], which can lead to various unwanted flow effects in wells, pipelines, and risers, such as unstable flow with slugs [83]. Therefore it is of importance for the oil and gas industry to be able to accurately predict the multiphase flow through their systems, given the input conditions, e.g. arrival pressure and mass flow rate [7]. This chapter will introduce these challenges in more detail. Also, a summary of previous work is made. This chapter is concluded with a description of the new steps that are taken in this study towards simulating multiphase flow.

Multiphase flow covers all the flow of multiple immiscible phases, being gas, liquid and solid particles. Due to the numerous combinations of moving matter, the area of multiphase flow is rather large. This research will focus on the mixture of gas and liquid. Solids transport, such as slurry flow, is thus not treated. Furthermore, immiscible liquids, e.g. a combination of water and oil, are also outside the scope of this work. In this thesis a two-phase mixture of water and air is investigated. This choice is made due to the available validation data and the relative ease of repeating experiments with these fluids in a fluid flow lab.

While the applicability range of multiphase flow is rather large, this thesis will focus on the challenges in the field of hydrocarbon production and transport. In the introduction of the book of Oliemans [58] several of these challenges in the upstream area are given. The importance of flow regime prediction is emphasised due to the influence that the flow regime has on the pressure loss and the liquid accumulation. Oliemans stresses the fact that pipeline cost can increase by millions of dollars when the diameter is increased by 1 [cm]. Thus, an accurate prediction of the flow behaviour is needed in order to choose the right pipe configuration and operational procedures.

This report focuses on the geometries as presented in figure 1.0.1. This geometry, which is also investigated by Worthen [95], is one of the possible ways of connecting a sub-sea production line with a large diameter to a Floating Liquefied Natural Gas (FLNG) vessel. Due to the fact that the risers are smaller in diameter than the production pipeline, multiple risers are needed to transport the fluids upward. A main design requirement is to guarantee that there is an equal split of liquid and gas among the two risers. Therefore a riser system as presented in figure 1.0.1a is investigated. Furthermore, in order to become familiar with the numerical solvers on smaller domains, the T-junction is isolated and simulations are performed for this geometry as well. The T-junction investigated in this thesis is presented in figure 1.0.1b.



(a) Riser system investigated in this thesis.

(b) T-junction investigated in this thesis.

Figure 1.0.1: Two geometries investigated in this thesis

1.1 PREVIOUS WORK

In order to resolve the 3D Navier-Stokes equations a Computational Fluid Dynamics (CFD) model is needed, due to the fact that an explicit mathematical solution for this flow problem does not exist. Various tools have been developed to reduce the computational costs compared to Direct Numerical Simulation (DNS) for turbulent flows. Classical solvers, as available in CFD packages such as CFX, FLUENT and OpenFOAM, use for instance Large Eddy Simulation (LES) or Reynolds-Averaged Navier-Stokes (RANS) to represent turbulent flows. Both techniques sacrifice simulation precision in order to speed up the calculations. LES will simulate the larger dynamic and turbulent scales, whereas a subgrid model is used for the smaller scales. By using RANS, only averaged values of a given turbulent flow field are obtained.

In this report the governing equations for multiphase fluid flow are given. Non-dimensional numbers for multiphase pipe flow have been given by for instance Taitel and Dukler [84]. These numbers define multiphase flow regimes, e.g. bubble flow, slug flow and annular flow. To solve these governing equations CFD methods are introduced. The first CFD method is DNS, in which all scales up to the Kolmogorov scale η are resolved. Due to the major computational cost [62, p. 336], the method is restricted to small Reynolds numbers. Despite its cost, DNS has its use for validation cases [1] and investigation of turbulence [70]. The second method applies averaging to the Navier-Stokes equations to obtain the RANS equations. By averaging the governing equations, one obtains additional terms which require closure relations. A more recent development in CFD is the use of LES. By dividing the flow field into large and small scales through applying a filter, the computational costs can be reduced. The unresolved scales can be modelled by a Sub-Grid Scale (SGS). Various models are developed to model the small scales, e.g. the Smagorinsky model and the Wall-Adapting Local Eddy (WALE) model.

Compared to single phase flow models, numerical methods for multiphase flow have to overcome extra difficulties. One of the most important modelling considerations is the choice of the interface capturing method. When interface capturing is omitted, numerical diffusion will occur which results in non-physical simulations. In the simulations in this thesis the Volume of Fluid (VOF) method is used. This methods is mass conservative [80] and widely implemented.

To provide insight in multiphase flow, dedicated methods are analysed. A large number of studies on Lattice Boltzmann methods have been conducted by for instance McNamara and Zanetti [52]. An overview the applicability of the Lattice Boltzmann method for particle collision in multiphase flow is given in the work of Aidun et al. [4]. However, the huge number of particles in large systems makes the tracking and calculation of the occurrence of collisions almost impossible. The Euler-Lagrange method can be seen as a middle way between Lattice Boltzmann and Euler-Euler (i.e. full-continuum) modelling. By using this method the collisions of particles are simulated using a particle density function, which decreases the computational cost [30].

However, the full two-fluid (Euler-Euler) approach is used for most multiphase flow systems. In this method the phases are modelled as continua and the interfacial stress, amongst other boundary conditions and stresses, determines their behaviour [38]. RANS in combination with the k-ϵ [65] description for the modelling of turbulence is used regularly in cited papers. For instance in the work of Hernandez-Perez et al. [36] about the choice of mesh, in the work of Chen et al. [16] about T-junctions and in the work of Capecelatro et al. [14] for risers. It can often be a good choice to model turbulence by RANS and the k-ϵ model. When a more detailed description is wanted, one can use LES in combination with a SGS model. In these models the smallest scales are modelled by choosing an appropriate value of the eddy viscosity. Difficulties in the best choice for the SGS model and its parameters may exist, depending on the problem description and geometry [55].

Two main current problems in multiphase flow analysis are investigated in this thesis, namely the junction and the vertical riser. Junctions are used in many processes, such as in hydrocarbon production and in chemical plants. In this report, the focus lies on splitting junctions, contrary to mixing junctions. Splitting junctions in multiphase flow showed a maldistribution in the experiments by Yang and Azzopardi [97]. The work of Chen et al. [16] couples multiple vertical T-junctions in order to obtain a relatively good splitting of the phases. However, research opportunities are prevalent in modelling of multiphase flow through junctions with for instance a larger diameter, different orientation or other flow inlet conditions.

Multiphase flow through risers is investigated for multiple flow regimes and pipe diameters. The behaviour of multiphase flow through vertical pipes is important because churn flow and slug flow can be detrimental for production continuity and for structural integrity. A numerical model for slug flow under different conditions is available [14]. Slug flow is experimentally investigated [50] and is compared to CFD codes in the literature [3]. In the work of Hernandez-Perez et al. [36] different meshes are investigated. Overall, it can be concluded that various studies have been conducted for one-momentum models such as OpenFOAM's `interFoam`. Clear research opportunities exist in the area of modelling multiphase flow with an Euler-Euler model, coupled with interface sharpening. An OpenFOAM solver that utilises this coupling is `multiphaseEulerFoam`.

1.2 NEW STEPS TOWARDS SIMULATING MULTIPHASE PIPE FLOW

As mentioned in the previous section, research on simulating multiphase flow with a two Euler description while including interface sharpening is scarce. The relative ease of using an one-fluid VOF code, such as `interFoam`, has led to numerous recent papers and theses about this topic [22, 26, 43, 49]. The few papers utilising the `multiphaseEulerFoam` solver use simple hexadral meshes [87] or focus on mixing problems [93].

This thesis uses the OpenFOAM library to simulate multiphase flow. The open source code of OpenFOAM has advantages for simulations in the academic world such as the possibility of implementing additional boundary conditions or turbulent models. Disadvantages include the lack of a proper manual and the steep learning curve of using OpenFOAM. Therefore this thesis also includes the governing equations of both the `interFoam` and `multiphaseEulerFoam` solvers and the additional equations used in the source code of these solvers. Furthermore, in the appendices, the case files for both solvers are included for reproducibility and to serve as a reference for future work.

This report investigates the applicability and performance of `multiphaseEulerFoam`, as described by Wardle and Weller [93], in more complex geometries: the impacting T-junction and a 2.5 [m] riser system. The meshing is done with an inbuilt OpenFOAM utility, which does not necessarily generate hexahedral cells. By using a non-hexahedral mesh, problems arise for the orthogonality and skewness corrections. While `interFoam`, due to its mixture momentum formulations, proves to handle these corrections well, additional measures have to be taken for `multiphaseEulerFoam`. In order to couple both phases in the `multiphaseEulerFoam` solver, a momentum-transfer term is used, consisting of the drag force, lift force, virtual mass force and other forces. The magnitude of these forces is governed by coefficients which need to be selected by the user of the solver.

The different formulations of the momentum equations result in different numerical results. The differences between the solvers are investigated in this thesis. As mentioned, papers about using `multiphaseEulerFoam` for multiphase pipe flow are scarce. Therefore, this thesis has the following research objective.

Research objective of this thesis

Investigate the differences between the VOF mixture description of OpenFOAM's `interFoam` module and the Euler-Euler coupled VOF description of OpenFOAM's `multiphaseEulerFoam` module when predicting the inlet pressure, liquid hold-up and production at the outlets for an impacting T-junction coupled with two 2.5 [m] risers with an internal diameter of $d = 0.05$ [m].

It is expected that the presence in the Euler-Euler formulation in the `multiphaseEulerFoam` solver will give a more realistic flow in terms of the droplet and wave formation. Furthermore, due to the difference in the momentum calculation of the two solvers, differences in the liquid production at the outlets are expected.

1.3 STRUCTURE OF THIS REPORT

This section gives an overview of the chapters of this report. In chapter 2 the governing equations of multiphase flow are presented. First, the Navier-Stokes equations are given. In order to distinguish single phase flow from multiphase flow behaviour, surface tension and multiphase flow regimes are explained. Using the governing equations DNS, RANS and LES are described. Especially the latter is of interest, because this technique is used in this report as turbulence model. Interface capturing is treated after the introduction of the SGS models in LES. Interface capturing techniques are used to clearly describe the location of an interface between the phases and to prevent numerical diffusion of this interface.

After the investigation of the working principles of the OpenFOAM solvers and solutions algorithms, the simulation set-up is explained. First, the riser system with a height of 2.5 [m] is meshed. Furthermore, the lower part of the riser system is extracted, such that a T-junction remains. This T-junction is also meshed. An OpenFOAM program called `snappyHexMesh` (SHM) is used to mesh these geometries. The resulting quality of the mesh is very important to obtain an accurate solution. Therefore the mesh quality parameters are monitored. Layers are added to the walls of the mesh, in order to correctly apply the turbulent wall functions.

After the mesh is made, the boundary conditions are either taken from sources or derived. Correct boundary conditions are important in OpenFOAM, due to the fact that a wrong description of the problem can lead to meaningless results or to an unstable simulation. In this thesis LES is used, as it is available in `multiphaseEulerFoam`, whereas RANS is not available. A Smagorinksy SGS is used in LES to model the small scales of the flow. Despite its shortcomings its use is widespread in the CFD community.

In chapter 5 the results of the simulation are discussed. Mesh dependency is checked for both geometries. The results of `interFoam` and `multiphaseEulerFoam` are discussed and compared. First a quantitative analysis of the results is made. This is done in order to check whether the flow appears to be physical. Chapter 6 compares the results from both solvers in the T-junction mesh and in the large riser mesh. Differences between the results from the simulations will be indicated. The calculated inlet pressure, the production at the outlet and the hold-up in the domains are the key parameters analysed.

Recommendations for further research are stated in chapter 7. The effect of the entrance length needs further research. In the simulations an entrance length of $z_{entr} = 10D = 0.5$ [m] is used. A longer entrance length is recommended. One `multiphaseEulerFoam` simulation for a long ($z = 7.5$ [m]) pipe is made and these results show that the stratified flow undergoes a transition into wavy stratified flow after 0.5 to 1 [m].

A term used extensively throughout this report is the interface compression velocity. This term governs, in both interFoam and multiphaseEulerFoam, the magnitude of the interface sharpening. By adding an interface compression velocity to the governing equation of the transport for the phase fraction (i.e. how much of a cell is filled with a fluid), diffusivity is prevented. For the simulations presented in this report interface compression is turned on. A first investigation of the effects of turning the interface compression off is given in chapter 7. A more extensive investigation of the effect of interface compression is recommended.

In chapter 7 a brief investigation on an alternative SGS is conducted. Further research on alternative SGSs could be performed, for instance on WALE. While the stability of WALE is questionable, this SGS is supported by multiphaseEulerFoam. It seems that WALE prefers highly orthogonal meshes. Further research on this model should be performed in order to confirm this hypothesis and to deploy it successfully in CFD simulations.

A good understanding of the Euler-Euler solver multiphaseEulerFoam can be beneficial for a broad range of engineering applications. The Euler-Euler formulation of the flow provides (in theory) a better description of the real flow behaviour. The advantages of OpenFOAM include the fact that it is free of a licence fee. Another benefit is the full insight in the working of the solver, which is especially advantageous in the academic field. With the steps taken in this thesis further research can be set-up in order to validate the multiphaseEulerFoam solver against pipe flow data. With that validation, the solver can become an indispensable tool to simulate multiphase pipe flow.

Chapter 2

Governing equations

This chapter introduces the governing equations for fluid flow, the multiphase flow regimes and the theoretical methods of modelling fluid flow. First, the governing equations are given. Then non-dimensional numbers for multiphase fluid flow are given. These numbers are used to define the multiphase flow regimes. This chapter is concluded by an overview of the theoretical methods in CFD and a summary of interface capturing methods.

2.1 FUNDAMENTALS OF MULTIPHASE FLOW

In order to set-up the governing equations, a control volume Ω with volume V is defined. This three-dimensional entity holds a certain amount of fluid. The fluid is allowed to enter and leave the volume with velocity $\boldsymbol{u}$ over the boundary of the volume $S = \partial\Omega$. In figure 2.1.1 a schematic illustration of the control volume is given.

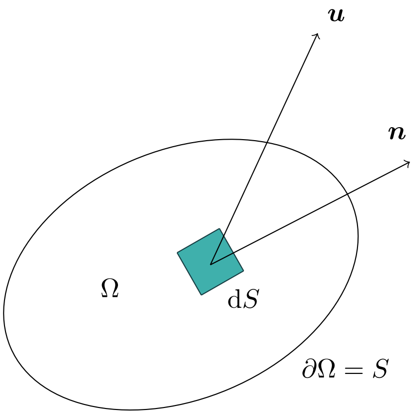


Figure 2.1.1: Control volume definition

By using the fact that mass is conserved in closed systems, it can be seen that the change in the sum of the all the masses $\rho\,\mathrm{d}V$ must be equal to the nett-outflow of mass through all surface parts $\rho\boldsymbol{n}\,\mathrm{d}S$ that are normal to the surface S. The integral mass balance reads:

$$\frac{\mathrm{d}}{\mathrm{d}t}\iiint_{\Omega}\rho\,\mathrm{d}V = -\iint_{\partial\Omega}\rho\boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}S \tag{2.1.1}$$

In the control volume momentum is conserved. One can apply Newton's second law $\boldsymbol{F} = m\boldsymbol{a}$ to the fluid flowing through the control volume to obtain the expression given in equation 2.1.2. The various terms represent from left to right: the time rate change of momentum, the nett-outflow of momentum, the pressure force, body forces, the viscous force $\boldsymbol{F}_{visc}$ and the external force $\boldsymbol{F}_{ext}$.

$$\frac{\mathrm{d}}{\mathrm{d}t}\iiint_{\Omega}\rho\boldsymbol{u}\,\mathrm{d}V = -\iint_{\partial\Omega}\rho\boldsymbol{u}\boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}S - \iint_{\partial\Omega} p\cdot\boldsymbol{n}\,\mathrm{d}S + \iiint_{\Omega}\rho\boldsymbol{f}_{body}\,\mathrm{d}V + \boldsymbol{F}_{visc} + \boldsymbol{F}_{ext} \tag{2.1.2}$$

Besides the mass and the momentum, the energy is also conserved in the control volume. In this report no heat transfer and temperature effects are taken into account. This means that the analysis of the energy equation is omitted in this thesis.

2.1.1 Differential form of the fluid flow equations

The differential form of the integral equations can be found by using the divergence theorem, which is also known under the name of Gauss's theorem.

Divergence theorem

Let Ω be a volume in $\mathbb{R}^3$ with a piecewise smooth boundary S. If a vector field $\boldsymbol{F}$ is continuously differentiable the following expression holds:

$$\iiint_{\Omega}(\nabla\cdot\boldsymbol{F})\,\mathrm{d}V = \iint_{\partial\Omega}\boldsymbol{F}\cdot\boldsymbol{n}\,\mathrm{d}S \tag{2.1.3}$$

The Reynolds transport equation is used to move a partial derivative with respect to time inside an integral. In this transport equation the term $\boldsymbol{u}_\Omega$ represents the velocity of the volume. A stationary control volume which is considered in this report implies $\boldsymbol{u}_\Omega = 0$.

Reynolds transport equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\iiint_{\Omega}\boldsymbol{F}\,\mathrm{d}V = \iiint_{\Omega}\frac{\partial\boldsymbol{F}}{\partial t}\,\mathrm{d}V + \cancel{\iint_{\partial\Omega}(\boldsymbol{u}_\Omega\cdot\boldsymbol{n})\boldsymbol{F}\,\mathrm{d}S} \tag{2.1.4}$$

By applying the divergence theorem and Reynolds transport equation to equations (2.1.1) and (2.1.2) the differential form of the continuity equation is obtained.

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}) = 0 \tag{2.1.5}$$

The momentum equation requires addition manipulation in order to present it in a simplified form. The Cauchy momentum equation is applied, which uses the stress tensor $\boldsymbol{\sigma}$ and the corresponding stresses on a fluid parcel as given in figure 2.1.2.

By dividing the stress tensor in normal and tangential components, the viscous and pressure forces can be expressed as follows:

$$\boldsymbol{\sigma} = \begin{bmatrix}\sigma_{11} & \sigma_{12} & \sigma_{13}\\ \sigma_{21} & \sigma_{22} & \sigma_{23}\\ \sigma_{31} & \sigma_{32} & \sigma_{33}\end{bmatrix} = \begin{bmatrix}\sigma_{11}+p & \sigma_{12} & \sigma_{13}\\ \sigma_{21} & \sigma_{22}+p & \sigma_{23}\\ \sigma_{31} & \sigma_{32} & \sigma_{33}+p\end{bmatrix} - \begin{bmatrix}p & 0 & 0\\ 0 & p & 0\\ 0 & 0 & p\end{bmatrix} = -p\boldsymbol{I} + \boldsymbol{\tau} \tag{2.1.6}$$

Here $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{\tau}$ is the deviatoric stress tensor. By applying the divergence theorem and the Reynolds transport equation to the integral form of the momentum equation in equation 2.1.2 and by combining the result with equation 2.1.6 the general differential form of the momentum equation for compressible flow is obtained.

$$\rho\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\nabla p + \nabla\cdot\boldsymbol{\tau} + \boldsymbol{f}_{body} + \boldsymbol{f}_{ext} \tag{2.1.7}$$

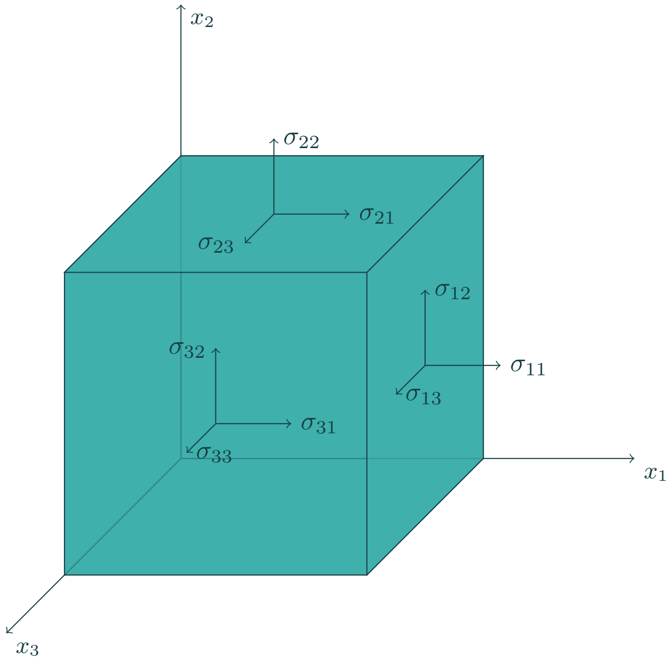


Figure 2.1.2: Visualisation of the stress tensor on a fluid particle $\mathrm{d}V = \mathrm{d}x_1\,\mathrm{d}x_2\,\mathrm{d}x_3$.

Equation 2.1.7 is valid for both Newtonian and non-Newtonian fluids. Generally, light oils and hydrocarbon gas can be considered Newtonian [32]. In Newtonian fluids the shear stress is linearly proportional to the local strain rate, i.e.:

$$\boldsymbol{\tau} \propto \nabla \boldsymbol{u} \tag{2.1.8}$$

With the additional assumptions that the the fluid is isotropic and that $\nabla \cdot \boldsymbol{\tau}$ equals zero for fluid at rest one arrives at:

$$\boldsymbol{\tau} = \mu \nabla \boldsymbol{u} \tag{2.1.9}$$

When this result is substituted into equation 2.1.7, the momentum equation for compressible, isotropic, Newtonian fluids is obtained:

$$\rho \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\nabla p + \nabla \cdot \left(\mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathrm{T}}\right)\right) + \nabla \left(\lambda \left(\nabla \cdot \boldsymbol{u}\right)\right) + \boldsymbol{f}_{body} + \boldsymbol{f}_{ext} \tag{2.1.10}$$

Where:

- μ Dynamic viscosity of the fluid [kg m^{-1} s^{-1}]
- λ Bulk viscosity of the fluid [kg m^{-1} s^{-1}]
- $\boldsymbol{f}_{body}$ Body forces per unit volume e.g. gravity [N m^{-3}]
- $\boldsymbol{f}_{ext}$ External forces per unit volume [N m^{-3}]

In equation 2.1.10 the material derivative (also known as substantial derivative) is used. This derivative describes the time rate change of a quantity moving through a velocity field $\boldsymbol{u}$:

$$\frac{\mathrm{D}(\cdot)}{\mathrm{D}t} = \frac{\partial(\cdot)}{\partial t} + \boldsymbol{u} \cdot \nabla(\cdot) \tag{2.1.11}$$

2.1.2 Surface tension

Surface tension is the natural behaviour of a fluid to minimise the surface area, such that the internal energy is minimised in the system. Over the interface between two fluids a pressure difference is present called the Young–Laplace pressure:

$$\Delta p_{g\to l} = \sigma \nabla_s \cdot \boldsymbol{n}_{l\to g} = \sigma \left(\frac{1}{R_1} + \frac{1}{R_2} \right) \tag{2.1.12}$$

Where:

- σ Surface tension coefficient [N m^{-1}]
- $\boldsymbol{n}_{l\to g}$ Unit normal pointing towards the gas phase [-]
- R_1 First characteristic radius of curvature [m]
- R_2 Second characteristic radius of curvature [m]

In figure 2.1.3 an illustration of the cohesive forces of two fluids is presented. Typical values of the surface tension are $\sigma = 72.88 \cdot 10^{-3}$ [N m^{-1}] for water–air at $T = 293$ [K] [24, p. 506] and $\sigma = 24.9 \cdot 10^{-3}$ to $34.0 \cdot 10^{-3}$ [N m^{-1}] at $T = 311$ [K] [2].

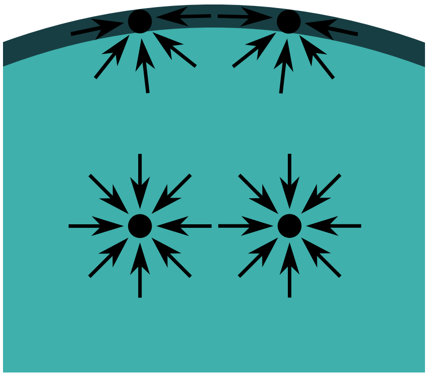


Figure 2.1.3: Surface tension in a fluid. Image part of the public domain.

2.1.3 Overview of flow regimes and their prediction

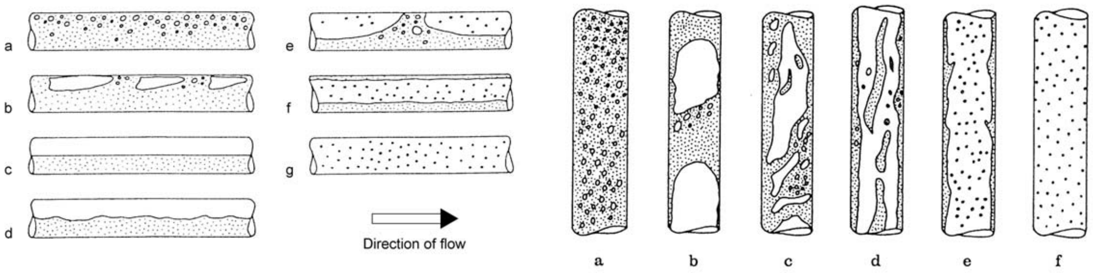
Multiphase flow in pipes behaves in different ways, depending on a number of properties such as the mass flow rate, the fractions of gas and liquid, the system pressure and the pipe geometry (e.g. pipe diameter and inclination angle). Experiments have been conducted and flow patterns have been analysed as illustrated in figures 2.1.4a and 2.1.4b. In the paper of Taitel and Dukler [84] a flow map has been constructed, which is an overview of possible flow patterns in a horizontal pipe, given the parameters from equation 2.1.13 [84]. In figure 2.1.5 the flow pattern map of Taitel and Dukler is shown.

Dimensionless groups used in Taitel and Dukler [84]

$$X = \sqrt{\frac{|(\mathrm{d}p/\mathrm{d}x)_l^s|}{|(\mathrm{d}p/\mathrm{d}x)_g^s|}}, \quad T = \sqrt{\frac{|(\mathrm{d}p/\mathrm{d}x)_l^s|}{\Delta\rho\, g \cos\alpha}},$$
$$Y = \frac{\Delta\rho\, g \sin\alpha}{|(\mathrm{d}p/\mathrm{d}x)_g^s|}, \quad F = \sqrt{\frac{\rho_g}{\Delta\rho}}\frac{u_g^s}{\sqrt{Dg\cos\alpha}}, \quad K = F\sqrt{\mathrm{Re}_l^s} \tag{2.1.13}$$

Where:

- X The Lockhart–Martinelli parameter
- T Ratio of turbulent forces and gravitational forces
- Y Relative forces acting on liquid from pressure losses and gravity



(a) Different horizontal flow patterns in a pipe. The shaded area represents the heavier fluid. **a)** bubble flow **b)** plug flow **c)** stratified flow **d)** wave flow **e)** slug flow **f)** annular flow **g)** spray flow.

(b) Different vertical flow patterns in a pipe. The shaded area represents the heavier fluid. **a)** bubble flow **b)** plug flow **c)** churn flow **d)** wispy-annular flow **e)** annular flow **f)** spray flow.

Figure 2.1.4: Flow patterns in horizontal and verticals pipes as illustrated in the book of Baehr and Stephan [10, p. 435].

- F Modified Froude number
- K Product of modified Froude number and the superficial Reynolds number

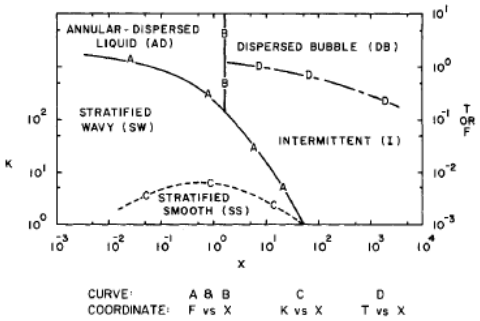


Figure 2.1.5: A flow pattern map for horizontal flow taken from the work of Taitel and Dukler [84].

In 1980 Taitel et al. [85] created a flow map for vertical flow for the bubble, slug, churn and annular flow regimes. Due to the complex nature of vertical flow through risers, a unique flow map has been constructed for different fluids. An example of this flow pattern map for vertical flow is given in figure 2.1.6.

2.1.4 Conclusion of governing equations

This section provides the basics of the governing equations of fluid flow. While this derivation is general, it can be transform with relative ease to the equations used in the CFD solvers in chapter 3. This section also provides an overview of multiphase flow behaviour in more detail by investigating surface tension and different multiphase flow regimes. Next section will investigate the theoretical methods of CFD and the different strategies to model turbulence.

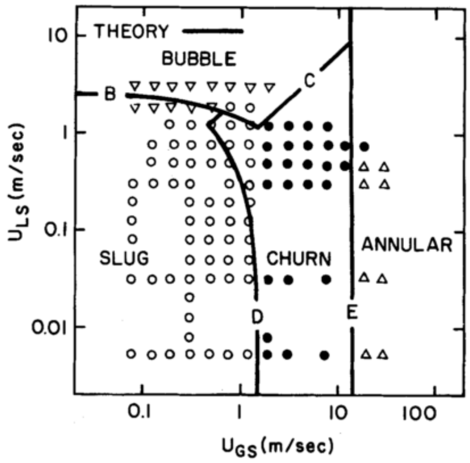


Figure 2.1.6: A flow pattern map for vertical water-air flow ($T = 298$ [K], $p = 1.0 \cdot 10^5$ [N m^{-2}], $D = 0.025$ [m], $L = 130D$ [m]) taken from the work of Taitel et al. [85]. Markers indicate experimental data for bubble flow (downward triangle), slug flow (circle), churn flow (dot) and annular flow (upward triangle).

2.2 Theoretical methods in computational fluid dynamics

For some simple geometries analytical solutions of the Navier-Stokes equations exist, but solutions in three dimensions with non-trivial geometries cannot be found. CFD is used to approximate the Partial Differential Equations (PDEs) in a numerical way. By dividing the computational domain in many control volumes, called cells, the flow can be simulated. Many numerical techniques to simulate turbulence exist, the most common methods will be treated in this section of this report.

CFD methods are becoming more optimised and specialised in different applications [92]. Numerical methods in pipe flow have been intensively used, especially for the laminar–turbulent transition and for fully developed turbulent structures [27]. While single phase flow solutions are developed, multiphase flow solvers lag behind. This is, among other factors, due to the larger complexity of this problem. This section provides a theoretical foundation of the developed CFD methods. First, the numerical techniques to simulate turbulence are introduced. Next, interface capturing methods, which are important in multiphase flows, are discussed.

2.2.1 Classical CFD solvers and their applicability on multiphase flow

Classically, three methods for numerically simulating the Navier-Stokes equations are available [55]. The first method to model turbulence is by using DNS. This method resolves all scales up to the scale of the smallest eddies (Kolmogorov scales) and can therefore be seen as exact. Unfortunately, DNS has a computational cost of $\mathcal{O}(\mathrm{Re}^3)$ [62, p. 336] and is therefore deemed to be not yet feasible for high Reynolds number flow [72]. One of the common methods for simulating turbulence is by using the RANS equations with the addition of a closure model for the Reynolds stress terms. The last common method of turbulence simulation is by using LES. CFD calculations with LES only resolve the large flow structures in the flow and use a SGS model to model the small scales.

Next sections will describe the CFD methods DNS, RANS and LES in that order. While DNS is not practical for high Reynolds number flow, it can be used as benchmark case for other simulation techniques. After that, RANS will be discussed briefly. This section is concluded by an analysis of LES and a short overview the SGS models.

2.2.1.1 DNS

As explained in the introduction of this section, DNS resolves all the structures of the flow up to the Kolmogorov scales [44]. The Kolmogorov scales are given in equation 2.2.1. DNS serves as a numerical validation tool. For example, in the paper of Santarelli and Fröhlich [70], DNS is used as a tool to investigate turbulent flow and the influence of small bubbles on turbulent structures.

Kolmogorov [44] scales

$$\eta \equiv \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}, \quad u_\eta \equiv (\epsilon\nu)^{\frac{1}{4}}, \quad \tau_\eta \equiv \left(\frac{\nu}{\epsilon}\right)^{\frac{1}{2}} \tag{2.2.1}$$

Where:

- η Kolmogorov length scale [m]
- u_η Kolmogorov velocity scale [m s^{-1}]
- τ_η Kolmogorov time scale [s]
- ν Kinematic viscosity [m^2 s^{-1}]
- ϵ Dissipation rate of turbulent kinetic energy [m^2 s^{-3}]

Recent developments in DNS have shown progress for two-phase flow at a moderate Reynolds number of Re $= 3111$ on a computational domain of $128 \times 513 \times 384$ cells [1]. The computational time was 46700 processor hours on 2048 cores (≈ 23 wall clock hours), which underlines the huge computational cost of DNS.

2.2.1.2 RANS

Turbulent flow is characterised by 3D motion and strong chaotic behaviour and is therefore computationally difficult to simulate. In his famous work Reynolds [65] has proposed the now-called RANS equations, stated in equation 2.2.2. The general Navier-Stokes equations are averaged under certain rules. The averaged values are indicated with a bar. These averaged quantities can be used in calculations. By averaging the Navier-Stokes equations an extra term appears. This Reynolds stress term $\nabla\left(\rho\overline{\boldsymbol{u}'\boldsymbol{u}'}\right)$ is the price one pays for averaging the equations, because an analytical representation of this term is unavailable and it must be modelled in order to use the equations.

Reynolds-Averaged Navier Stokes equations without external forces

$$\begin{aligned} \nabla \cdot \overline{\boldsymbol{u}} &= 0 \\ \rho\frac{\mathrm{D}\overline{\boldsymbol{u}}}{\mathrm{D}t} &= -\nabla\overline{p} + \mu\nabla^2\overline{\boldsymbol{u}} + \boldsymbol{g} - \nabla\left(\rho\overline{\boldsymbol{u}'\boldsymbol{u}'}\right) \end{aligned} \tag{2.2.2}$$

2.2.1.3 LES

LES is a filtering technique with a computational cost between RANS and DNS [12]. The technique has been introduced by Smagorinsky [75] for weather predictions. By using LES the intermediate solution for the solver applies a low-pass filter, such that only the large scales in the flow are resolved. In areas of interest, e.g. near walls or on fluid-gas interfaces, the solver resolves the smaller scales, such that a reliable solution is obtained. This simulation technique is used in various ways to model multiphase flows [30, 82].

In order to understand the idea of dividing the flow in large and small scales, the energy cascade is introduced. This concept was introduced by Richardson [66] in 1922. The energy cascade provides a relation between the turbulent kinetic energy spectrum $E(\kappa)$ and the wave numbers of the eddies κ (not to be confused with the von Kármán constant)

$$k = \int_0^\infty E(\kappa)\,\mathrm{d}\kappa \tag{2.2.3}$$

Where:

- k mean turbulence kinetic energy per unit mass [m^2 s^{-2}]
- $E(\kappa)$ energy spectrum of turbulent flow per unit mass [m^3 s^{-2}]
- κ wave number [rad m^{-1}]

The concept of the energy cascade is that turbulence is produced in the large scales of the flow, e.g. by vortex shedding. These large vortices degenerate into smaller eddies, which will fall apart until the smallest turbulent scale, the Kolmogorov scale, is reached. At the Kolmogorov scale the energy is dissipated as heat. By using LES a cut-off wave number $\tilde{\kappa}_c$ is applied, after which eddies in the flow are modelled instead of resolved. This technique is used to decrease computational time, while the large governing structures in the simulation are still present.

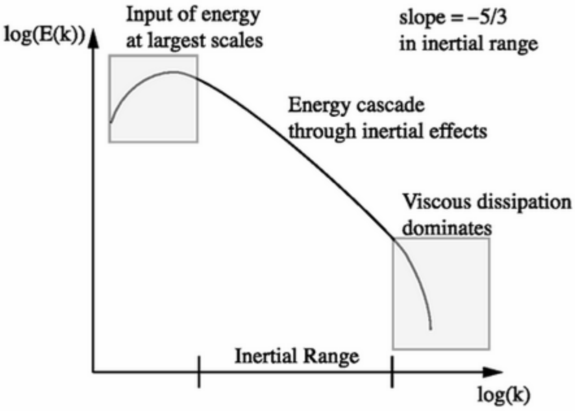

log(E(k))
Input of energy at largest scales
slope = −5/3 in inertial range
Energy cascade through inertial effects
Viscous dissipation dominates
Inertial Range
log(k)

Figure 2.2.1: Energy cascade graphically displayed by Berselli et al. [11].

The definition of the LES filter is defined by the spatial and temporal convolution integral over the velocity field as given in equation 2.2.4.

Definition of LES filtering by Leonard [46]

$$\overline{\boldsymbol{u}\,(t,\boldsymbol{x})} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \boldsymbol{u}\,(t,\boldsymbol{x})\,G\,(t-t',\boldsymbol{x}-\boldsymbol{r})\,\mathrm{d}t'd\boldsymbol{r} = G\star\boldsymbol{u} \tag{2.2.4}$$

After filtering the velocity field is divided in large and small scales by:

$$\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}' \tag{2.2.5}$$

The large scales in the flow are now given by $\overline{\boldsymbol{u}}$ and the small scales $\boldsymbol{u}'$ should be modelled in order to simulate turbulent flow. Note that the LES decomposition is not equal to the decomposition obtained in Reynolds decomposition. As mentioned, the filter G uses a spatial parameter and a temporal parameter, Δ and τ_c respectively. The filter function can be chosen under the constraint:

$$\int G(\boldsymbol{r},\boldsymbol{x})\mathrm{d}\boldsymbol{r} = 1 \tag{2.2.6}$$

The book of Pope provides common filter kernels G, which are listed in table 2.2.1. Furthermore, the filters are graphically represented in figure 2.2.2. In the table the Heaviside step function H is used which is defined as:

$$H(x) \equiv \frac{\mathrm{d}}{\mathrm{d}x} \max\{0, x\} \tag{2.2.7}$$

Table 2.2.1: Common choices for the filter kernel G in LES. Adapted from Pope [61, p. 563].

Name	Filter function G
Box	$\frac{1}{\Delta} H \left(\frac{1}{2}\Delta - \lvert r \rvert \right)$
Gaussian	$\sqrt{\frac{6}{\pi \Delta^2}} \exp \left(-\frac{6r}{\Delta^2} \right)$
Sharp spectral	$\frac{\sin\left(\frac{\pi r}{\Delta}\right)}{\pi r}$
Cauchy	$\left[24\,\Delta \left[\left(\frac{r}{\Delta} \right)^2 + \left(\frac{\pi}{24} \right)^2 \right] \right]^{-1}$

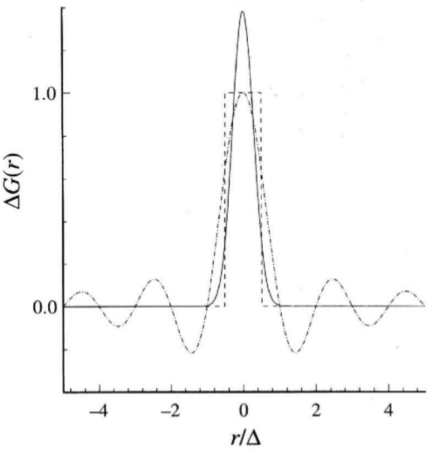


Figure 2.2.2: Graphic representation of filter functions G: box filter (dashed line), Gaussian filter (solid line) and spectral filter (dot-dashed line) [61, p. 564].

When the large scales $\bar{\boldsymbol{u}}$ are separated from the small scales $\boldsymbol{u}'$, a model for the latter term is needed. This SGS model describes turbulence in the scales which are smaller than the spatial filter length Δ. These turbulent stresses have to be determined in order to model the behaviour of the small scale structures of the flow. Most SGSs are based on Boussinesq's turbulent viscosity hypothesis [74] which relates the turbulence stress tensor to the strain rate tensor $\boldsymbol{S}_{ij}$. Compactly, this can be written as:

Boussinesq's turbulent viscosity hypothesis [74] for incompressible flow

$$\tau_{ij} + \frac{2}{3} k \boldsymbol{I} = 2\nu_t \boldsymbol{S}_{ij} \tag{2.2.8}$$

Where:

- τ_{ij} Turbulence stresses to be determined [Pa]

- k Turbulent kinetic energy [J]
- $\boldsymbol{I}$ Unit tensor [-]
- ν_t Turbulent viscosity in [m^2 s^{-1}]
- $\boldsymbol{S}_{ij}$ $= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$, the strain rate tensor [-]

The additional stress term is expressed in known terms by applying Boussinesq's hypothesis. The turbulent viscosity ν_t is yet unknown. This turbulent viscosity can be modelled in various ways, of which the Smagorinsky model [25], given in equation 2.2.9, is a common choice. Despite its shortcomings [55] such as the modelling of very fine droplets in a liquid [41], its use is prevalent in literature, e.g. in the paper of Tomaselli and Christensen [87] and the papers reviewed in the work of Tabor and Baba-Ahmadi [82]. Other models for SGS modelling include the Bardina model, filtered Bardina model and the mixed model [11, 55].

$$\nu_t = c_s \Delta^2 \sqrt{2\boldsymbol{S}_{ij}\boldsymbol{S}_{ij}} \tag{2.2.9}$$

Where:

- c_s Smagorinsky constant. Value depends on the problem, e.g. 0.13 for high-Reynolds-number turbulence and 0.17 in the inertial range [62].

Another interesting SGS model is WALE from the work of Nicoud and Ducros [56]. WALE has advantages over Smagorinsky in the vicinity of walls. The eddy viscosity of WALE is given in equation 2.2.10.

$$\nu_t = (C_m \Delta)^2 \frac{\left(\boldsymbol{S}^d_{ij}\boldsymbol{S}^d_{ij}\right)^{3/2}}{\left(\boldsymbol{S}_{ij}\boldsymbol{S}_{ij}\right)^{5/2} + \left(\boldsymbol{S}^d_{ij}\boldsymbol{S}^d_{ij}\right)^{5/4}} \tag{2.2.10}$$

Here $\boldsymbol{S}^d_{ij}$ is an alternative strain tensor formulation:

$$\boldsymbol{S}^d_{ij} = \frac{1}{2}\left(\overline{g}^2_{ij} + \overline{g}^2_{ji}\right) - \frac{1}{3}\delta_{ij}\overline{g}^2_{kk} \tag{2.2.11}$$

Where:

- $\overline{g}_{ij}$ $= \frac{\partial u_i}{\partial x_j}$

In this thesis the only SGS model used is the Smagorinsky model. In chapter 7 a case set-up with WALE is investigated for flow through a T-junction. Further research opportunities on alternative SGSs will be investigated in that chapter.

2.2.1.4 Conclusion on CFD methods

The numerical methods for simulating turbulence were analysed in this section. By using DNS all flow structures are resolved. Due to the large computational cost, using this technique is not feasible for large domains and for high Reynolds number flow. The RANS equations are obtained by applying Reynolds averaging on the governing equations. Additional closure relations are needed to model the Reynolds stress in RANS. LES methods use a spatial and temporal filter on the governing equations to filter out small scales in the flow. These small scales are modelled by a SGS. With the computational time being in between RANS and DNS it provides a good alternative when a detailed solution of high Reynolds number flow is required.

2.2.2 Interface capturing methods

Interface capturing methods are used to track multiphase flow interfaces and bubble formation. In a classical Euler-Euler simulations the interface is not followed, which results in numerical diffusion as

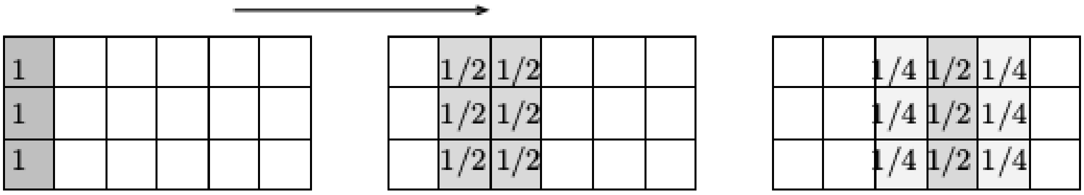


Figure 2.2.3: Schematic example of numerical diffusion of a two-phase flow [13].

shown in figure 2.2.3 [13]. Interface capturing methods prevent or limit numerical diffusion via various mechanisms, e.g. by using a "compression velocity" $\boldsymbol{u}_c$ [93] or by using the ghost fluid method [42].

Interface capturing methods aim to be as precise as possible, to reduce spurious currents [49, 72] and to minimise the computational loads. The problem description and scale of the problem strongly influence the best choice of interface capturing model in simulations [5, 91].

The following section explains the VOF method. VOF is based on the Simple Line Interface Calculation (SLIC) method of Noh and Woodward [57] and it forms the foundation of many interface tracking methods used in CFD solvers [26, 45, 93]. The OpenFOAM solvers `interFoam` and `multiphaseEulerFoam` use VOF. Other interface capturing methods, such as level-set [79] and Coupled Level Set Volume of Fluid (CLSVOF) [80] are not treated in this report.

2.2.2.1 VOF

VOF methods utilise a colour function to describe which cell is occupied by what fluid. Let $c(t, \boldsymbol{x})$ be the colour function and $V_{i,j,k}$ the cell volume then, for a two-phase fluid, the function is defined as [13]:

$$c(t, \boldsymbol{x}) \equiv \begin{cases} 1, & \text{if } \boldsymbol{x} \in \Omega_l \\ 0, & \text{else} \end{cases} \tag{2.2.12}$$

Where:

- Ω_l part of domain occupied by fluid l

The colour function in $\mathbb{R}^3$ is written in a discrete way as [15]:

$$c^n_{i,j,k} = \frac{1}{V_{i,j,k}} \iiint_{V_{i,j,k}} c\left(t, \boldsymbol{x}\right) \,\mathrm{d}V \tag{2.2.13}$$

In multi-fluid cells, i.e. cells with $0 < c(t, \boldsymbol{x}) < 1$ the orientation of the interface is given by neighbouring cells. A 2D example using Least Squares Volume of Fluid Interface Reconstruction Algorithm (LVIRA) [63] is given in figure 2.2.4 [15]. By minimising the function $G_{i,j}(\boldsymbol{n})$ the unit normal of the interface is found:

$$G_{i,j}(\boldsymbol{n}) = \sum_{m=-1}^{1} \sum_{n=-1}^{1} \left(c_{i+n,j+m} - c'_{i+n,j+m}(\boldsymbol{n})\right)^2 \tag{2.2.14}$$

The VOF method is a powerful and mass conservative [33] tool to mitigate numerical diffusion and satisfying results are obtained in vertical pipes [21], inertia-dominated flows [26], atomisation of droplets [26] and mixing problems [93].

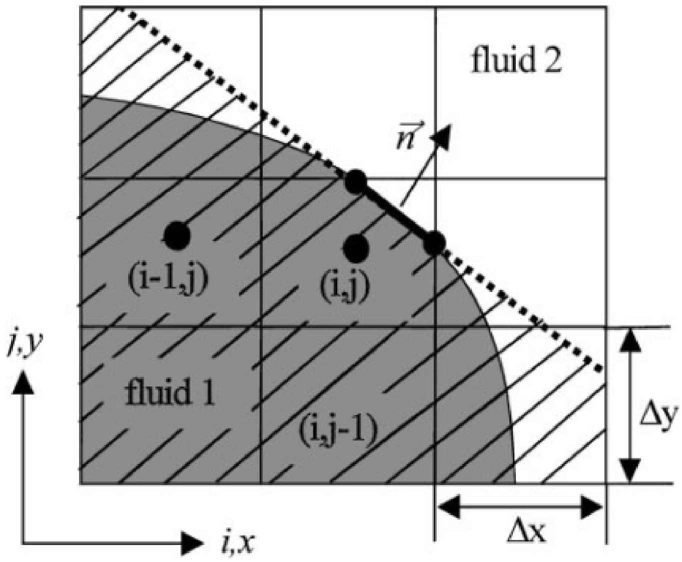


Figure 2.2.4: LVIRA example on $\mathbb{R}^2$ to determine the interface surface normal [15].

2.3 SUMMARY OF GOVERNING EQUATIONS

In this section the general governing equations for fluid flow were presented. Emphasis was put on the different flow regimes which are present in multiphase flow systems. Three methods to simulate turbulence were treated, of which LES will be used in this report. Furthermore, interface capturing methods were listed. The VOF method is used in both OpenFOAM solvers. The next chapter will explain the working principle of the solvers `interFoam` and `multiphaseEulerFoam`, as well as the differential schemes and matrix solvers used. Thereafter the simulations can be set-up and performed.

Chapter 3

OpenFOAM

This chapter introduces the basics of OpenFOAM, which is an open source software package for solving PDEs. In this thesis OpenFOAM **version 3.0.1**[1] is used. The structure of OpenFOAM is introduced in the first section. Secondly, the solvers utilised in this research, `interFoam` and `multiphaseEulerFoam`, are introduced. To solve the discretised equations, differential schemes are needed. These schemes are introduced in section 3.3. After that, a brief overview of matrix solvers is provided. This chapter is concluded by the two solution algorithms used in the simulations: Multidimensional Universal Limiter with Explicit Solution (MULES) and Pressure Implicit with Splitting of Operator (PISO).

3.1 Introduction to OpenFOAM

OpenFOAM is a `C++` library of source code for solvers and utilities. The acronym Field Operations and Manipulation (FOAM) states the general purpose of this library. Besides CFD calculations OpenFOAM can also be used for finite element methods and financial option calculations. A general structure of OpenFOAM can be found in figure 3.1.1.

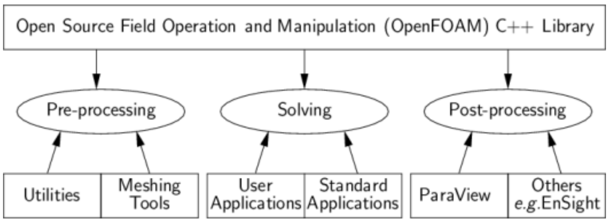


Figure 3.1.1: Structure of OpenFOAM. Source: `http://openfoam.com`.

OpenFOAM uses a case folder structure to set-up and save case data. An initial state of a case of OpenFOAM is shown in figure 3.1.2. Three directories are located in the root of the case folder: `0`, `constant` and `system`. The `0` directory contains the initial values of for instance the pressure and velocity. Values for these variables have to be initialised for the internal domain, as well as the boundaries, inlet patches and outlet patches. Examples of these files can be found in chapter A, as they are added for reproducibility.

As the name suggests, all constants are saved in the `constant` directory. In figure 3.1.2 one can see the `polyMesh` sub-folder where the mesh geometry is saved. Besides that folder, the files `g`, `transportProperties` and `turbulenceProperties` are located in `constant`. These files contain the orientation of the gravitational vector, fluid properties (e.g. viscosity) and turbulent settings (such as choice of SGS model), respectively.

[1] OpenFOAM build `3.0.1-d8a290b55d28` on the `hpc12` cluster of Delft University of Technology

The `system` folder contains information about meshing and how OpenFOAM should discretise and solve the Navier-Stokes equations. Meshing parameters are saved in `blockMeshDict` and `snappyHexMeshDict`. Information about differential schemes is located in `fvSchemes`. The `fvSolution` file is responsible for settings about matrix solvers and the used algorithms (e.g. PISO). To run the simulation on multiple processors, the `decomposePar` utility is used to divide the mesh over multiple processors. The settings for this utility are saved in `decomposeParDict`. Finally, one of the most important files for OpenFOAM is the `controlDict` file. This file contains settings about for example the solver, the magnitude of time step and the maximum Courant number.

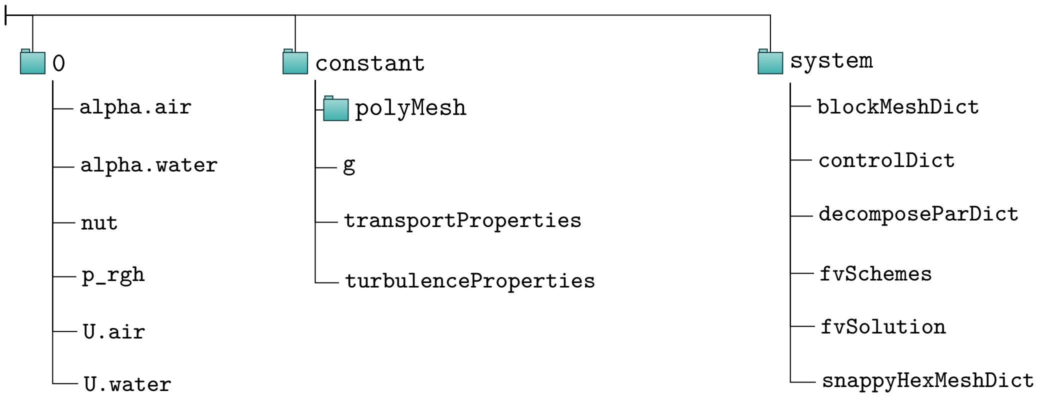


Figure 3.1.2: An example of the OpenFOAM case directory structure for `multiphaseEulerFoam`.

3.1.1 Advantages and disadvantages of using OpenFOAM

Using OpenFOAM instead of ANSYS FLUENT or CFX has a number of advantages and disadvantages. While the commercial programs run as a black box, OpenFOAM is an open-source toolbox, which means that the `C++`-code is readily available for inspection. The open-source nature is both the strength and the weakness of OpenFOAM. These pros and cons will be investigated in this subsection.

3.1.1.1 Advantages of OpenFOAM

Due to the availability of the source code, OpenFOAM offers a high degree of insight in the solver. With `C++` knowledge, the working principles of the solvers and utilities can be investigated. Especially for academic use it can be valuable to use the source code of the solver, for example to check whether the numerics are implemented correctly.

Another advantage of the availability of the source code is the quick bug fixing done by the community. When there are bugs in parts of OpenFOAM, users can search for the source of the errors and submit them to a designated forum for bugs in OpenFOAM. Furthermore, when a user has `C++` knowledge, the code can be modified to their demands. This means that the implementation of additional functions is relatively easy.

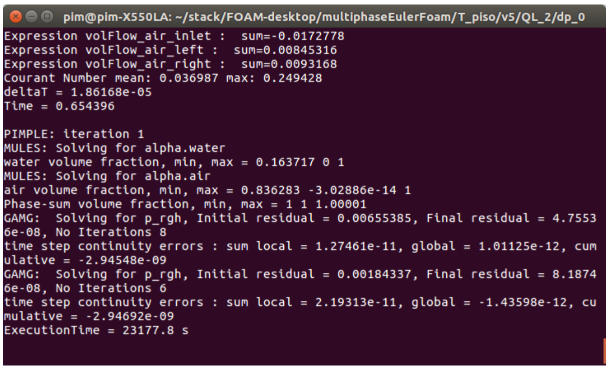
The availability of numerous solvers and boundary conditions in OpenFOAM is another advantage. In OpenFOAM version 3.0.1 more than 80 solvers and 115 boundary conditions are available. This means that a user has a large degree of flexibility in setting up cases and processing them.

The last advantage is the fact that OpenFOAM can be downloaded without license fee for all purposes. No license costs apply for the OpenFOAM core. Extra support is offered by companies and their product can be shipped with a Graphical User Interface (GUI). An example of such a support tool is Helix-OS.

3.1.1.2 DISADVANTAGES OF OPENFOAM

Historically, the OpenFOAM community has been divided several times into different routes. This means that they use different software packages, examples include OpenFOAM+ and OpenFOAM-ext. The cases are not necessarily interchangeable between the different packages.

The lack of a native GUI is another disadvantage. The OpenFOAM core works with a terminal interface only. Especially for new users, the lack of visual guidance can be overwhelming. Figure 3.1.3 provides an example of the output of running `multiphaseEulerFoam`.



pim@pim-X550LA: ~/stack/FOAM-desktop/multiphaseEulerFoam/T_piso/v5/QL_2/dp_0

```
Expression volFlow_air_inlet :  sum=-0.0172778
Expression volFlow_air_left :  sum=0.00845316
Expression volFlow_air_right :  sum=0.0093168
Courant Number mean: 0.036987 max: 0.249428
deltaT = 1.86168e-05
Time = 0.654396

PIMPLE: iteration 1
MULES: Solving for alpha.water
water volume fraction, min, max = 0.163717 0 1
MULES: Solving for alpha.air
air volume fraction, min, max = 0.836283 -3.02886e-14 1
Phase-sum volume fraction, min, max = 1 1 1.00001
GAMG:  Solving for p_rgh, Initial residual = 0.00655385, Final residual = 4.7553
6e-08, No Iterations 8
time step continuity errors : sum local = 1.27461e-11, global = 1.01125e-12, cum
ulative = -2.94548e-09
GAMG:  Solving for p_rgh, Initial residual = 0.00184337, Final residual = 8.1874
6e-08, No Iterations 6
time step continuity errors : sum local = 2.19313e-11, global = -1.43598e-12, cu
mulative = -2.94692e-09
ExecutionTime = 23177.8 s
```

Figure 3.1.3: A screenshot of a running instance of OpenFOAM's `multiphaseEulerFoam`.

The last major disadvantage is the lack of a proper manual for the solvers. The learning curve for a new user is very steep. To correctly set-up working cases can be difficult and it is often done by trial and error. The manual that is available on cfd.direct is succinct. Users often have look for help on CFD forums[2].

3.2 USED SOLVERS

In the previous chapter the governing equations for fluid flow are given. The resulting form of the Navier-Stokes equations is general and applicable for 3D compressible flows. In this report the solvers use a fluid description for incompressible flow. Therefore several simplifications can be made. The continuity equation is simplified to:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{3.2.1}$$

As introduced, solvers are used to numerically solve governing PDEs. In this report results from two solvers are presented: the results from the homogeneous mixture solver `interFoam` and the results from the two-fluid solver `multiphaseEulerFoam`. Both solvers use VOF for interface sharpening. In this section the governing equations of the `interFoam` solver are introduced. This section is concluded by the Euler-Euler approach utilised in `multiphaseEulerFoam`.

3.2.1 INTERFOAM

OpenFOAM's `interFoam` is an incompressible, two-phase solver with VOF interface sharpening. This solver uses the homogeneous mixture approach, in which the properties of the two phases, e.g. velocity and density, are averaged. One momentum equation for the mixture is solved, which means that it behaves as a single fluid solver. The `interFoam` solver uses the averaged continuity equation for incompressible flow [37]:

$$\nabla \cdot \overline{\boldsymbol{u}} = 0 \tag{3.2.2}$$

[2] e.g. http://www.cfd-online.com/Forum/

Where:

- $\overline{\boldsymbol{u}}$ The averaged velocity of the phases with (in this thesis): $\overline{\boldsymbol{u}} = \alpha_l \boldsymbol{u}_l + \alpha_g \boldsymbol{u}_g = \alpha_l \boldsymbol{u}_l + (1-\alpha_l)\boldsymbol{u}_g$

The homogeneous mixture momentum equation for incompressible flow is given by [37]:

$$\frac{\mathrm{D}\overline{\boldsymbol{u}}}{\mathrm{D}t} = -\frac{1}{\overline{\rho}}\nabla p + \frac{\overline{\mu}}{\overline{\rho}}\nabla^2 \cdot \overline{\boldsymbol{u}} + \boldsymbol{g} + \frac{\boldsymbol{F}_s}{\overline{\rho}} \tag{3.2.3}$$

Where:

- $\overline{\rho}$ Averaged density: $\overline{\rho} = \alpha_l \rho_l + (1-\alpha_l)\rho_g$
- $\overline{\mu}$ Averaged viscosity: $\overline{\mu} = \alpha_l \mu_l + (1-\alpha_l)\mu_g$
- $\boldsymbol{F}_s$ Surface tension force
- $\boldsymbol{g}$ Gravity vector

The following advection equation for the liquid phase fraction α_l is used [94]:

$$\frac{\mathrm{D}\alpha_l}{\mathrm{D}t} + \nabla \cdot (\boldsymbol{u}_c \alpha_l (1-\alpha_l)) = 0 \tag{3.2.4}$$

The addition of the term $\nabla \cdot (\boldsymbol{u}_c \alpha_l (1-\alpha_l))$ ensures that interface sharpening is only active in regions with intermediate α_l, i.e. on the boundary between phases. The compression velocity is an artificial velocity added close to the interface to provide interface sharpening [93]:

$$\boldsymbol{u}_c = \min (C_\alpha |\boldsymbol{u}|, \max (|\boldsymbol{u}|)) \frac{\nabla \alpha_l}{|\nabla \alpha_l|} \tag{3.2.5}$$

Where:

- $\boldsymbol{u}_c$ Interface compression velocity
- $\nabla \alpha_l$ Gradient of the void fraction
- C_α Compression indicator

When the compression indicator C_α is limited to $C_\alpha \leq 1$ equation 3.2.5 becomes:

$$\boldsymbol{u}_c = C_\alpha |\boldsymbol{u}| \frac{\nabla \alpha_l}{|\nabla \alpha_l|} \tag{3.2.6}$$

And the binary indicator for switching interface compression on or off:

$$C_\alpha = \begin{cases} 1, & \text{interface sharpening active} \\ 0, & \text{interface sharpening } \textbf{inactive} \end{cases} \tag{3.2.7}$$

The surface tension force follows from the paper of Albadawi et al. [5] and is defined as:

$$\boldsymbol{F}_s = \sigma \nabla \cdot \left(\frac{\nabla \alpha_l}{|\nabla \alpha_l|} \right) \nabla \alpha_l \tag{3.2.8}$$

3.2.2 MultiphaseEulerFoam

The `multiphaseEulerFoam` solver is a multiphase flow implementation of OpenFOAM which combines an Euler-Euler approach with VOF interface sharpening. The Euler-Euler approach means that a separate momentum equation is solved for each phase. The resulting fields are coupled by a momentum transfer tensor $\boldsymbol{F}_{b,k}$ and the surface tension $\boldsymbol{F}_{s,k}$. The momentum equation for each phase k in `multiphaseEulerFoam` is:

$$\frac{\mathrm{D}(\alpha_k \boldsymbol{u}_k)}{\mathrm{D}t} = -\frac{\alpha_k}{\rho_k}\nabla p + \nu_k \nabla \cdot (\alpha_k \nabla \boldsymbol{u}_k) + \alpha_k \boldsymbol{g} + \frac{\boldsymbol{F}_{b,k}}{\rho_k} + \frac{\boldsymbol{F}_{s,k}}{\rho_k} \tag{3.2.9}$$

With the advection equation for each phase fraction α_k [94]:

$$\frac{\mathrm{D}\alpha_k}{\mathrm{D}t} + \nabla \cdot (\boldsymbol{u}_c \alpha_k (1 - \alpha_k)) = 0 \tag{3.2.10}$$

In equation 3.2.9 the force $\boldsymbol{F}_{b,k}$ is the sum of the forces on bubbles and drops, given by:

$$\boldsymbol{F}_{b,k} = \boldsymbol{F}_{d,k} + \boldsymbol{F}_{l,k} + \boldsymbol{F}_{vm,k} + \boldsymbol{F}_{o,k} \quad \text{with} \quad \sum \boldsymbol{F}_{b,k} = 0 \tag{3.2.11}$$

Here the subscripts d, l, vm and o are the drag force, lift force, virtual mass force and other forces, respectively. Other forces (i.e., $\boldsymbol{F}_{o,k}$), including the Basset force, are often neglected [77]. The total momentum transfer between phases equals zero, due to global conservation of momentum.

The `multiphaseEulerFoam` solver requires an explicit formulation for which phase is continuous and which is dispersed. The user has to define the continuous and the dispersed phase, given by subscript c and d, respectively. In this thesis water is chosen to be the dispersed phase. The influence of this choice can be seen in the drag and lift force between phases. The drag force for phase k is defined as [93]:

$$\boldsymbol{F}_{d,k} = \frac{3}{4}\rho_c \alpha_c \alpha_d C_D \frac{|\boldsymbol{u}_d - \boldsymbol{u}_c| (\boldsymbol{u}_d - \boldsymbol{u}_c)}{d_d} \tag{3.2.12}$$

Here C_D is the drag coefficient of a bubble or a droplet. Throughout this study the drag model of Schiller and Naumann [73] is used which is defined as:

$$C_D = \begin{cases} \dfrac{24\left(1 + 0.15\,\mathrm{Re}_d^{0.683}\right)}{\mathrm{Re}_d} & \text{if } \mathrm{Re}_d \leq 1000 \\ 0.44 & \text{if } \mathrm{Re}_d > 1000 \end{cases} \tag{3.2.13}$$

Here Re_d is the droplet Reynolds number:

$$\mathrm{Re}_d = \frac{|\boldsymbol{u}_d - \boldsymbol{u}_c| d_d}{\nu_c} \tag{3.2.14}$$

The lift force $\boldsymbol{F}_{l,k}$ is defined as:

$$\boldsymbol{F}_{l,k} = -C_L \rho_c \alpha_k \boldsymbol{u}_k \times (\nabla \times \boldsymbol{u}_c) \tag{3.2.15}$$

Where:

- C_L Lift coefficient with a value of $C_L = 0.5$ [8] [68, p. 263]

The virtual mass forces is a consequence of the fact that when an immersed bubble is accelerated, also a part the surrounding fluid is accelerated. When applying Newton's second law to this combined movement, the virtual mass force $\boldsymbol{F}_{vm,k}$ is found [68, p. 312]:

$$\boldsymbol{F}_{vm,k} = C_{vm} \rho_c \alpha_k \left(\frac{\mathrm{D}\boldsymbol{u}_c}{\mathrm{D}t} - \frac{\mathrm{D}\boldsymbol{u}_k}{\mathrm{D}t} \right) \tag{3.2.16}$$

Where:

- C_{vm} Virtual mass coefficient for phase k, with a value of $C_{vm} = 0.5$ [31]

3.3 CHOICE OF DIFFERENTIAL SCHEME

Differential schemes are numerical approximations of the terms in the governing equations, which are provided to OpenFOAM in the `system/fvSchemes` file. Most terms in this research are approximated linearly. For instance, the Euler time scheme for a scalar φ is linear and defined as:

$$\frac{\mathrm{d}\varphi(t)}{\mathrm{d}t} \approx \frac{\varphi(t + \Delta t) - \varphi(t)}{\Delta t} \tag{3.3.1}$$

In order to transfer values from cell faces to the cell centre, interpolation is used, see figure 3.3.1. This project uses Gaussian interpolation, which is defined for scalar φ as:

$$\iiint_{V_P} \nabla\varphi \,\mathrm{d}V = \iint_{\partial V_P} \mathrm{d}S\,\varphi = \sum_f \boldsymbol{s}_f \varphi_f \tag{3.3.2}$$

Where:

$$\varphi_f = f_x \varphi_P + (1 - f_x)\,\varphi_N \tag{3.3.3}$$

With $f_x = \overline{fN}/\overline{PN}$, in which P and N are nodal points and f is their shared face, $\overline{fN}$ the distance between face f and cell centre N and $\overline{PN}$ is the distance between cell centres.

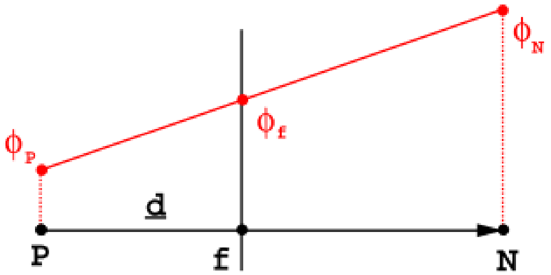


Figure 3.3.1: Graphical representation of interpolation by Jasak [40, p. 81].

3.3.1 Flux limiters

The simulations performed in the next chapters employ flux limiters of the Total Variation Diminishing (TVD) type. Fluxes are quantities flowing between cell faces and are therefore of major interest in finite volume calculations. The TVD property has been introduced by Harten [35] and provides a definition for limiters which dampen oscillations in CFD simulations. All limiters which fall in the region indicated in figure 3.3.2 are TVD [81]. In the simulation the van Leer limiter [89] is used for derivatives of the fluxes in the phase fraction:

$$\phi_{vl}(r) = \frac{r + |r|}{1 + |r|}, \qquad \text{with } \lim_{r\to\infty} \phi_{vl}(r) = 2 \tag{3.3.4}$$

This is a valid choice considering the research in this area by for instance Liu and Hinrichsen [48]. Another limiter investigated by Liu and Hinrichsen is the limitedLinearV scheme:

$$\phi_{ll}(r) = \max\left[\min\left(\frac{2r}{k}, 1\right), 0\right], \qquad \text{with } 0 < k \leq 1 \tag{3.3.5}$$

In the simulations a value $k = 1$ is used. This limiter is used for the velocity flux.

3.3.2 Differential schemes used in simulations

The simulations performed in this thesis use linear differential schemes. As mentioned in the introduction of this section, the time derivatives are represented by the linear scheme from equation 3.3.1. Gradient schemes, for example $\nabla\alpha$, are approximated by linear schemes with Gauss interpolation. The same linear scheme is used for divergence terms that include the flux ϕ, e.g. $\nabla \cdot (\phi \boldsymbol{u})$, but additionally the `limitedLinearV` flux limiter from equation 3.3.5 is used. Divergence schemes including α use the `vanLeer` limiter given in equation 3.3.4. Laplacian schemes, e.g. $\nabla^2 \cdot \boldsymbol{u}$, are approximated with a linear scheme, Gauss interpolation and a non-orthogonality correction. In the `system/fvSchemes` file this is indicated by

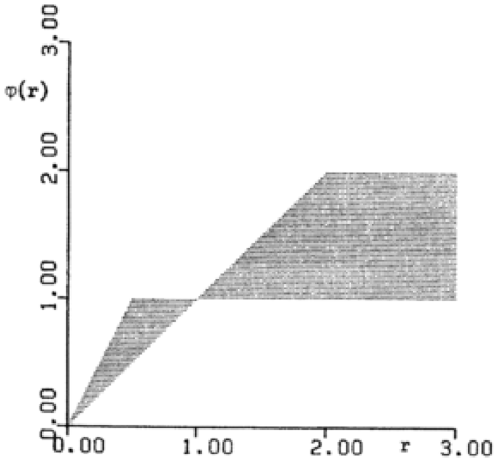


Figure 3.3.2: TVD region by Sweby [81].

the entry `Gauss linear corrected`. The default interpolation schemes are linear. The `snGradSchemes` indicator governs the gradient normal to the face of a cell. Non-orthogonality in these gradients is handled by the keyword `corrected`.

All schemes, except for the Euler scheme are of second order. A second order Crank-Nicolson time scheme [20] is available in OpenFOAM, but the Euler-scheme is chosen due to its prevalence in the tutorial files of OpenFOAM and the possible unstable behaviour of Crank-Nicolson [9]. All differential schemes used throughout this report are specified in the appendices for future reference. Now that when the differential schemes are known, the equations can be set-up and linear systems can be constructed by OpenFOAM. The next section will briefly touch upon the solution algorithms to solve those systems of equations.

3.4 MATRIX SOLVERS

Matrix solvers are used to solve linear systems present in (intermediate) solutions of the simulation and are prescribed in the `system/fvSolution` file.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{3.4.1}$$

Here $\boldsymbol{A}$ is a sparse matrix. This system is solved iteratively with Geometric Algebraic Multigrid (GAMG). By using GAMG, the solver will compute a pressure solution on a coarse grid first. By using this solution as an initial guess, faster convergence in the pressure momentum coupling can be found [78]. The `nCellsInCoarsestLevel` parameter in the `system/fvSolution` file shows how many cells are used for the coarsest mesh. Different smoothing methods exist, including diagonal incomplete-Cholesky, faster diagonal incomplete-Cholesky and Diagonal incomplete-LU [69, 88].

3.5 SOLUTION ALGORITHM

Several algorithms are used in the simulations to get the equations to convergence within a given tolerance. MULES is used to solve the phase fraction in an iterative way and it is explained in the following subsection. After the explanation of MULES, the pressure-momentum equation coupling algorithm PISO is investigated.

3.5.1 MULES

The MULES algorithm is designed to serve as an extra correction for the phase fraction α_k. By sub-cycling in a time step, the calculation for the phase fraction can be made more stable [22]. When the Courant-Friedrichs-Lewy (CFL) condition is relaxed, i.e. made higher than one, sub-cycling can be necessary for a stable solution. The `nAlphaSubCycles` parameter in the `fvSolution` file governs how many sub-cycles are calculated. Figure 3.5.1 provides a graphical representation of the MULES algorithm as described in the work of Damián [22]. Throughout this thesis no sub-cycling for α is used, thus `nAlphaSubCycles` = 1.

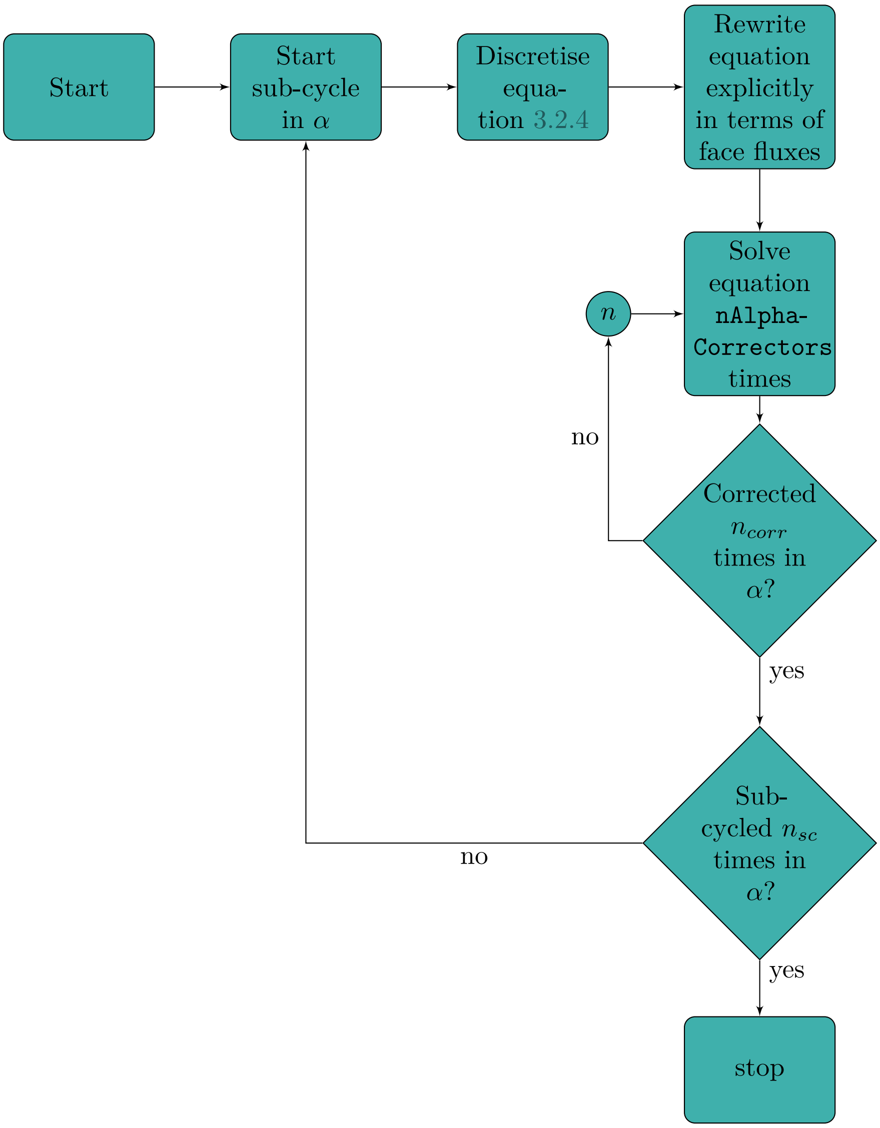

Start
Start sub-cycle in α
Discretise equation 3.2.4
Rewrite equation explicitly in terms of face fluxes
Solve equation nAlpha-Correctors times
n
no
Corrected n_{corr} times in α?
yes
Sub-cycled n_{sc} times in α?
no
yes
stop

Figure 3.5.1: The MULES algorithm as described in the work of Damián [22].

3.5.2 PISO, SIMPLE, PIMPLE

Semi-Implicit Method for Pressure-Linked Equations (SIMPLE), PISO and Merged PISO-SIMPLE (PIMPLE) are algorithms to solve the discretised coupled pressure momentum equations in OpenFOAM. Additional explanation on SIMPLE is omitted in this thesis, because the algorithm is designed for steady state problems. Additional information on the topic can be found in the studies of Jasak [40, p. 148] and Patankar [59].

In 1986 Issa [39] published a method on the non-iterative solving of the coupling between pressure and momentum, called PISO. The algorithm uses a series of predictor and corrector steps to calculate the coupling between the discretised pressure and momentum equations. In OpenFOAM the behaviour of PISO is governed in the `PISO` sub-library in the `fvSolution` file. By setting `nCorrectors` to a value above one, the algorithm performs more corrector steps, improving stability though at an additional computational cost. In this thesis a value for `nCorrectors` of 2 is used. In figure 3.5.2 a diagram with the principles of PISO is presented.

While PIMPLE is not used in this research, it can be noted that the algorithm is a combination of SIMPLE and PISO in which the relaxation of equations is combined with the corrector steps of PISO. By using PIMPLE, one can speed up the calculations, because stability can be maintained for a CFL condition of Co > 1. In this thesis PIMPLE is not used due to fact that LES uses explicit formulations which require Co < 1.

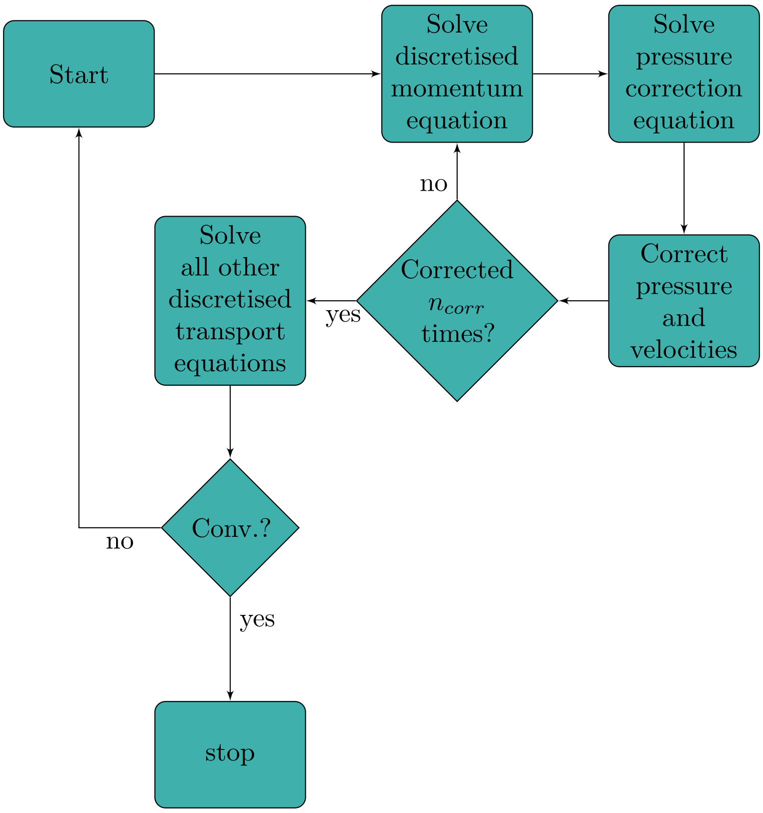

Start
Solve discretised momentum equation
Solve pressure correction equation
Correct pressure and velocities
Corrected n_{corr} times?
no
yes
Solve all other discretised transport equations
Conv.?
no
yes
stop

Figure 3.5.2: The PISO algorithm [39].

3.6 SUMMARY OF METHODS OF OPENFOAM

In this chapter the basics of OpenFOAM were explained. Cases in OpenFOAM are set-up by using the `0`, `constant` and `system` folders and associated files in those folders. While OpenFOAM has multiple, mostly academic advantages, the lack of proper documentation and existence of multiple versions are serious downsides.

In this research the `interFoam` and `multiphaseEulerFoam` solvers are used to simulate water and air in a riser system. The `interFoam` solver uses a averaged velocity field. This averaged velocity field is used to solve a single momentum equation. Another approach for the simulation of two-phase flow is using the `multiphaseEulerFoam` solver. This solver uses an Euler-Euler description of the flow, which means that two separate momentum equations are solved, one for air and one for water. These equations are coupled with the momentum transfer vector.

To discretise the governing equations of both solvers, differential schemes are used. The choice for the differential scheme per term in the equations must be prescribed in the `system/fvSchemes` file. In this research linear schemes with Gauss interpolation are used. In order to stabilise the discretisation, flux limiters such as `vanLeer` and `limitedLinearV` are used.

When the equations are discretised, a linear system is solved. The user has to specify the choices of matrix solvers by specifying them in the `system/fvSolution` file. In this thesis GAMG is used for the pressure equation. To finish calculating a time step, OpenFOAM has to couple the pressure equation to the momentum equation. This coupling is obtained by the PISO algorithm, made for transient simulations. By utilising correctors, a stable simulation can be obtained. Another algorithm used is MULES. This algorithm calculates and corrects the phase fraction in the mesh.

This chapter provided a basic overview of the working principles of the of OpenFOAM solvers. The next chapter will specify the meshing of the domain and the boundary conditions applied to the inlet, outlets and walls. With the settings provided in this chapter and the boundary conditions that will be provided in next chapter, the case description is complete.

Chapter 4

Simulation set-up

This chapter describes the simulation set-up. Whether a CFD simulation is successful depends on multiple factors, such as the mesh quality and the specified boundary conditions. This chapter will describe the domain geometry first. After that, the meshing procedure is explained. On the walls, inlet and both outlets boundary conditions must be specified, which are explained in section 4.3.

4.1 DOMAIN GEOMETRY

An open-source Computer-Aided Design (CAD) program called FreeCAD is used to specify the domain geometry. This program outputs STereoLithography (STL) files which can be read by various other CAD programs. In this research the inner diameter of the pipe is $d = 0.05$ [m]. In figure 4.1.1 a schematic drawing of the domain is given, which will be indicated as *riser system* in this report. This riser system can be found in figure 4.1.2a.

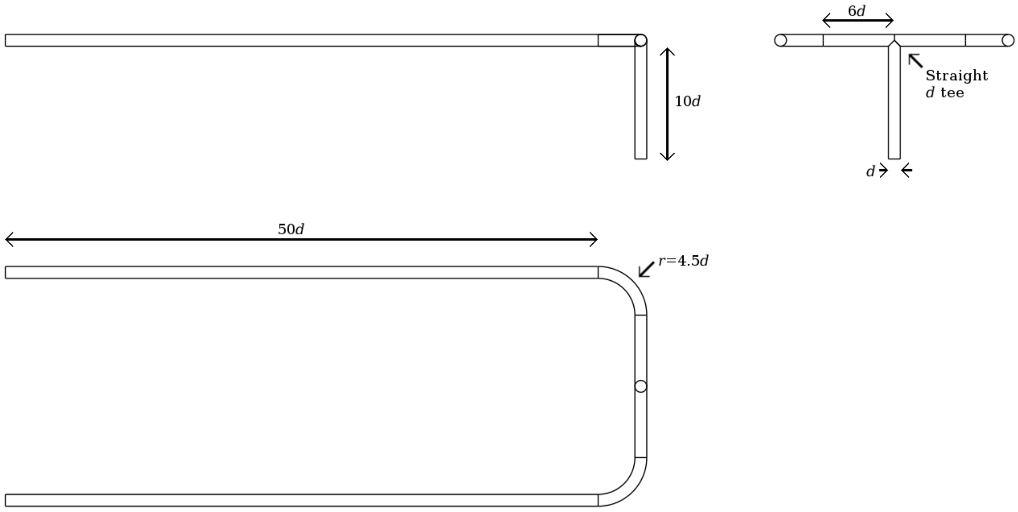
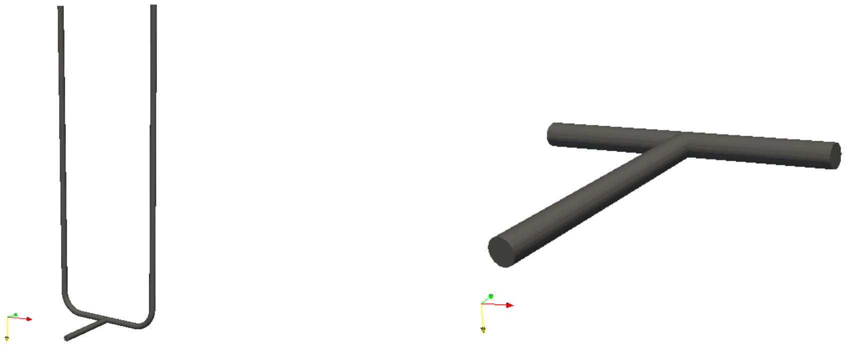


Figure 4.1.1: Schematics of the pipe system. In this report $d = 0.05$ [m].

Besides the complete riser system, a smaller geometry will be considered. This geometry, from here named the *T-junction* can be found in figure 4.1.2b. Simulations for this junction are performed to check whether this smaller geometry can be used to predict the flow behaviour in the complete riser system.



(a) Riser system investigated in this thesis.

(b) T-junction investigated in this thesis.

Figure 4.1.2: Two geometries investigated in this thesis

4.2 MESHING OF THE DOMAIN

Meshing is the process of transforming the computational domain into discrete cells. For simple geometries meshing is easy, because the domain is for instance fully rectangular and can be composed of fully hexahedronical, i.e. six-sided, cells. For more complex geometries, considered in this report, meshing is not straightforward. The meshing procedure used for the geometry described is explained in the next section. The general mesh type is the butterfly mesh [36].

4.2.1 Meshing procedure

OpenFOAM includes a tool to mesh arbitrary geometries called SHM. In the `snappyHexMeshDict` file one can specify various options which govern the mesh generation of SHM. First a background mesh is made which, together with the STL files, is used as input for SHM. The background mesh is usually made with the `blockMesh` utility and consists of rectangular blocks which together span a volume larger than the geometry. The SHM procedure consists of three stages: creation of the castellated mesh, snapping and layer addition.

4.2.1.1 Generation of castellated mesh and mesh snapping

A castellated mesh is made by refining the rectangular cells made with `blockMesh`. By splitting the cells the input geometry is approximated. The user can specify how many refinements must be made to approximate the surface. More refinement levels provide a better representation of the geometry, but it increases the number of cells. When the castellated mesh is ready, SHM will move the cells around to fit the specified geometry. This procedure is called snapping.

4.2.1.2 Adding layers

In the simulations the viscous layers are not resolved, but modelled by wall functions. These functions are used as a estimation to model flow near the wall without having to fully resolve the viscous region of the boundary layer. Fully resolving the viscous region requires a fine near-wall mesh spacing, which results in small cells and increased computational time [86].

The `nutkWallfunction` patch type uses the law of the wall (given in equation 4.2.1 and shown in figure 4.2.1) to calculate the velocity profile close to the wall.

$$u^+ = \frac{u}{u_\tau} = \frac{1}{\kappa} \ln\left(Ey^+\right) = \frac{1}{\kappa} \ln\left(y^+\right) + C^+ \tag{4.2.1}$$

Where:

u^+	The dimensionless velocity close to the wall
u_τ	Wall friction velocity
κ	The von Kármán constant with value $\kappa \approx 0.40$
E	An integration constant with value 9.8 for smooth walls
C^+	An integration constant with value 5.5 for smooth walls

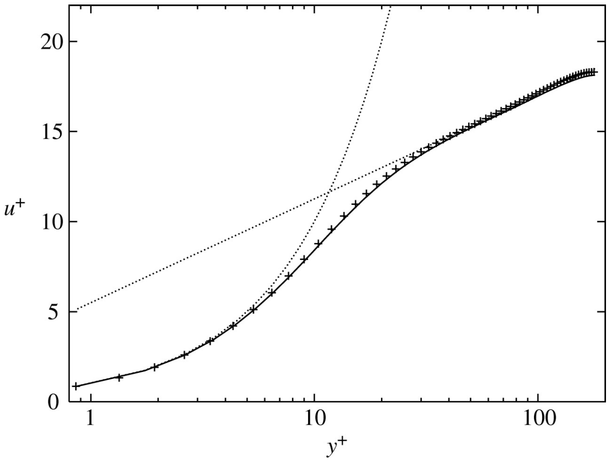


Figure 4.2.1: Graphical representation of u^+ from the work of Alfonsi and Primavera [6]. Solid line: DNS data from Moser et al. [54]. Curved dotted line: $u^+ = y^+$. Straight dotted line: $u^+ = 2.5 \ln(y^+) + 5.5$

A requirement to use wall functions in OpenFOAM is that the first cell has to be located in the buffer layer ($5 < y^+ < 30$) to fully model the velocity profile [86]. In order to find the layer thickness the wall friction coefficient C_f is required. Many wall friction correlations exist and one of them is the Blasius correlation which is used by Taitel and Dukler [84]:

$$C_f = 0.046\,\mathrm{Re}^{-0.2} = 0.046 \left(\frac{\mathcal{U}D}{\nu}\right)^{-0.2} \tag{4.2.2}$$

From the wall friction the wall shear stress can be calculated as:

$$\tau_w = \frac{1}{2} C_f \rho \mathcal{U}^2 \tag{4.2.3}$$

Which is transformed into the wall distance y^+ as:

$$y^+ = \frac{u_\tau y}{\nu}, \quad \text{with} \quad u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{4.2.4}$$

Combining equations 4.2.2 to 4.2.4 gives for the first cell wall distance y_1:

$$y_1 = \frac{\nu}{\sqrt{\frac{1}{2} \cdot 0.046 \left(\frac{\mathcal{U}D}{\nu}\right)^{-0.2} \mathcal{U}^2}} \cdot y^+ = 6.5938\, d^{0.1} \left(\frac{\nu}{\mathcal{U}}\right)^{0.9} \cdot y^+ \tag{4.2.5}$$

4.2.2 Quality control of snappyHexMesh

In the three stages of SHM there will be checks for the intermediate and final mesh against the mesh quality control parameters. It is of importance to govern these parameters strictly, in order to generate a mesh of high enough quality. In the work of Fabritius and Tabor [28] the mesh quality is evaluated by three parameters: skewness, non-orthogonality and minimum cell size. Another important measure for the mesh quality is the aspect ratio. These four mesh quality controls are discussed in this subsection.

Orthogonality is mathematically defined by two vectors having a scalar product of zero. For a hexahedronical mesh orthogonality would mean that the corners of each face are right angles. Non-orthogonality appears when the angles between faces are not 90 [deg]. Figure 4.2.2 shows the non-orthogonality between cell-centres P and N. Orthogonality is preferable because the diffusive terms of the Navier-Stokes equations use the face normal vector in order to calculate the flux φ between cells [28]. SHM does not produce structured meshes, thus some non-orthogonality is likely to appear in the generated mesh. The keyword `maxNonOrtho` in the `meshQualityControls` dictionary limits the non-orthogonality in the various stages of the meshing procedure.

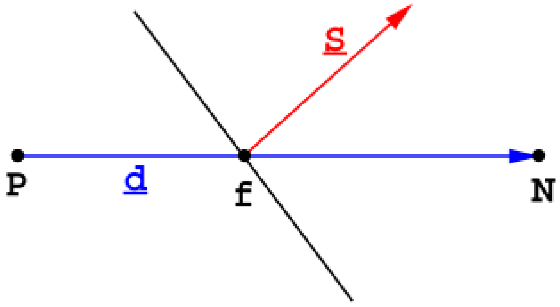


Figure 4.2.2: Non-orthogonality between two cells P and N. Image from the work of Jasak [40].

Besides non-orthogonality, skewness also introduces a numerical error of the diffusion-type [40]. Figure 4.2.3 graphically displays the effect of skewness between cells. To obtain a successful interpolation, the vector $\boldsymbol{S}$ should be calculated at the middle of the face, i.e. on point f. Due to skewness this point between P and N lies on f_i, which is not necessarily on the centre of the face. Vector $\boldsymbol{m}$ indicates this skewness error. Jasak [40] states that in order for a mesh to be of reasonable quality the following inequality should hold:

$$|\boldsymbol{m}| < |\boldsymbol{d}| \tag{4.2.6}$$

Where

- $\boldsymbol{d}$ The vector from node P to N

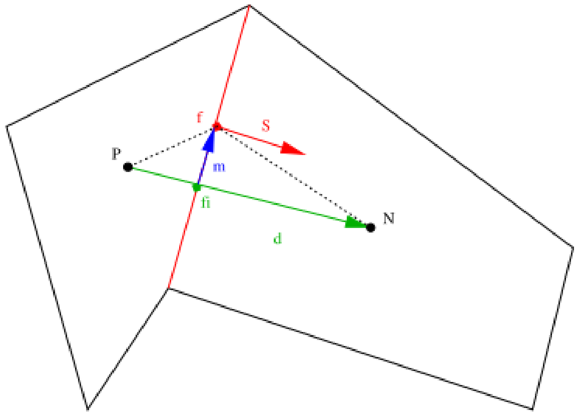


Figure 4.2.3: Skewness between two cells. Image from the work of Fabritius and Tabor [28].

The aspect ratio of a cell is the ratio between the largest and smallest face of a cell. Ideally, the aspect ratio of a cell should be one. High aspect ratios can lead to interpolation errors [28]. Minimum cell size is important in maintaining a sufficiently large time step. As explained in section 4.7 (equation 4.7.1), the time step is dependent on the cell dimensions. When velocity in a cell is sufficiently high and the cell dimension in that direction is small, the time step will be small due to the fact that the Courant number is limited to 1 or lower. The `minVol` entry in the `meshQualityControls` dictionary governs the minimal cell size. From the volume a typical length scale $\widetilde{\Delta x}$ for each cell can be derived by:

$$\widetilde{\Delta x} = \sqrt[3]{V_{cell}} \tag{4.2.7}$$

Where:

- V_{cell} is the cell volume.

In order to set-up the meshing of SHM, a mesh quality dictionary is used inside `snappyHexMeshDict`. The used mesh quality control are stated in table 4.2.1.

Table 4.2.1: Mesh quality settings used by SHM in this thesis.

Quality parameter	Value	Unit
maxNonOrtho	65	[deg]
maxBoundarySkewness	20	[-]
maxInternalSkewness	4	[-]
maxConcave	80	[-]
minFlatness	0.5	[-]
minVol	10^{-13}	[m^3]
minTetQuality	10^{-9}	[-]
minTwist	0.02	[-]
minDeterminant	0.001	[-]
minFaceWeight	0.02	[-]
minVolRatio	0.01	[-]

4.3 BOUNDARY CONDITIONS

In order to solve PDEs in CFD, boundary conditions have to be prescribed. This section will explain which boundary conditions are used in the simulations. First, the velocity boundary condition at the wall is given. Next, the pressure boundary condition at the wall will be derived, followed by a description of the phase fraction at the wall. This section will be concluded with the boundary condition for the turbulent viscosity at the wall.

4.3.1 VELOCITY AT THE WALL

The no-slip condition at the wall is explained in the paper of Day [23].

$$\boldsymbol{u}\big|_w = \boldsymbol{0} \tag{4.3.1}$$

This Dirichlet boundary condition describes that fluid near the boundary “sticks” to the wall, preventing fluid close to the wall from moving.

4.3.2 Wall boundary condition for the pressure

From the no-slip condition at the wall, the boundary condition for pressure at the walls can be derived. Consider the steady, incompressible Navier-Stokes equations with body force vector $\boldsymbol{f}_b = [f_{b,1}, f_{b,2}, f_{b,3}]^{\mathrm{T}}$ at the wall:

$$\cancel{u_1\frac{\partial u_1}{\partial x_1}} + \cancel{u_2\frac{\partial u_1}{\partial x_2}} + \cancel{u_3\frac{\partial u_1}{\partial x_3}} = -\frac{1}{\rho}\frac{\partial p}{\partial x_1} + \nu\left(\cancel{\frac{\partial^2 u_1}{\partial x_1^2}} + \cancel{\frac{\partial^2 u_1}{\partial x_2^2}} + \cancel{\frac{\partial^2 u_1}{\partial x_3^2}}\right) + g_1 + f_{b,1} \tag{4.3.2}$$

$$\cancel{u_1\frac{\partial u_2}{\partial x_1}} + \cancel{u_2\frac{\partial u_2}{\partial x_2}} + \cancel{u_3\frac{\partial u_2}{\partial x_3}} = -\frac{1}{\rho}\frac{\partial p}{\partial x_2} + \nu\left(\cancel{\frac{\partial^2 u_2}{\partial x_1^2}} + \cancel{\frac{\partial^2 u_2}{\partial x_2^2}} + \cancel{\frac{\partial^2 u_2}{\partial x_3^2}}\right) + g_2 + f_{b,2} \tag{4.3.3}$$

$$\cancel{u_1\frac{\partial u_3}{\partial x_1}} + \cancel{u_2\frac{\partial u_3}{\partial x_2}} + \cancel{u_3\frac{\partial u_3}{\partial x_3}} = -\frac{1}{\rho}\frac{\partial p}{\partial x_3} + \nu\left(\cancel{\frac{\partial^2 u_3}{\partial x_1^2}} + \cancel{\frac{\partial^2 u_3}{\partial x_2^2}} + \cancel{\frac{\partial^2 u_3}{\partial x_3^2}}\right) + g_3 + f_{b,3} \tag{4.3.4}$$

Which results in:

$$\left.\frac{\partial p}{\partial x_n}\right|_w = \rho g_n + \rho f_{b,n} \tag{4.3.5}$$

This result corresponds to OpenFOAM's `fixedFluxPressure`. Contrary to the `zeroGradient` boundary condition this method takes hydrostatic pressure and other body forces into account.

4.3.3 Phase fraction at the wall

Due to the expression of convection of the liquid phase fraction in the two analysed solvers, the `zeroGradient` boundary condition is imposed at the wall. In this case surface tension effects between the wall and the fluid are ignored. An other option for the boundary condition of the phase fraction include the contact angle boundary condition, which is not further investigated in this thesis.

4.3.4 Turbulent viscosity at the wall

As already explained in section 4.2.1.2 a wall function for the turbulent viscosity ν_t will be used. This function models the inner layer of the flow close to the pipe wall. The `nutkWallFunction` method uses an approximation based on turbulent kinetic energy k as follows [67].

$$\nu_t = \nu\left(\frac{ky^+}{\ln\left(Ey^+\right)} - 1\right), \quad \text{with} \quad y^+ = C_\mu^{1/4}\frac{y\sqrt{k}}{\nu} \tag{4.3.6}$$

Where:

- C_μ a constant with value 0.09

4.4 INITIAL, INLET AND OUTLET CONDITIONS

The CFD simulations performed in this research use one inlet and two outlets. The initial values on the inlet patch determine the flow behaviour in the domain. First the initial state of the system will be treated. Then, the inlet and outlet conditions are explained.

4.4.1 Initial conditions

The initial conditions in the system must be specified in order for OpenFOAM to start the simulations. At time $t = 0$, the pipe is filled with only air ($\alpha_l = 0$). The air in the domain is at rest, i.e. $\boldsymbol{u} = \mathbf{0}$ [m s^{-1}]. Furthermore, the pressure is taken equal to $p = 0$ [Pa]. Note that this is possible due to the use of incompressible solvers. The initial condition of fluid at rest results in a turbulent viscosity of zero.

4.4.2 Initial condition of the hold-up

A stratified flow regime is prescribed at the inlet. The method of predicting the initial phase fraction α_l (or low liquid hold-up or fraction) uses the paper of Spedding and Chen [76], rewritten by Hart et al. [34]:

$$\alpha_l = \left[1 + 0.45\left(\frac{\dot{Q}_g}{\dot{Q}_l}\right)^{0.65}\right]^{-1} \tag{4.4.1}$$

The liquid height of the pipe can be calculated with the expression from equation 4.4.2. In figure 4.4.1 a schematic overview of the liquid height in a pipe is given. Note that when the initial liquid hold-up α_l is known, the expression has to be solved numerically for the liquid height h_l. For example, when $\alpha_l = 0.2$ and $d = 0.05$ [m], the liquid height is solved to be $h_l = 0.0127$ [m].

$$\alpha_l = \frac{4}{\pi d^2}\left[\left(\frac{d}{2}\right)^2 \cos^{-1}\left(\frac{d-2h_l}{d}\right) - \left(\frac{d}{2} - h_l\right)\sqrt{dh_l - h_l^2}\right] \tag{4.4.2}$$

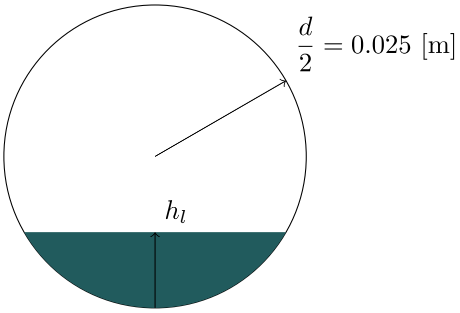
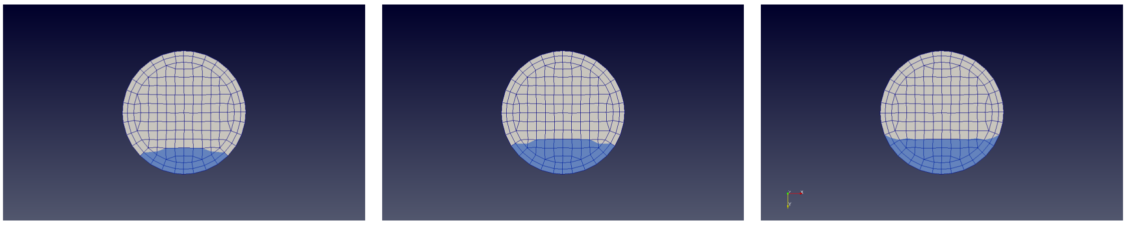


Figure 4.4.1: Schematic of liquid height h_l in a circular pipe with diameter $d = 0.05$ [m].

Whether a cell on the inlet patch is filled with water is determined by the location of the cell centre. The Groovy Boundary Condition (GroovyBC) from SWiss Army Knife (SWAK) for FOAM is used to specify the initial hold-up at the inlet patch. The initial values for the liquid hold-up from table 4.4.1 result in the hold-up profiles from figure 4.4.2.



(a) $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}] (b) $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] (c) $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}]

Figure 4.4.2: Hold-up on the inlet patch for the simulated flow rates.

4.4.3 Velocities

A mixture of air and water flows into the domain at the inlet patch. The total volumetric flow rate is taken as:

$$\dot{Q}_{total} = 64.2\ [\text{m}^3\ \text{h}^{-1}] = 1.78 \cdot 10^{-2}\ [\text{m}^3\ \text{s}^{-1}] \tag{4.4.3}$$

The volumetric flow rate of the liquid is simulated with values $\dot{Q}_l = \{1, 2, 3\}$ [m^3 h^{-1}]. Combined with the diameter of the pipe and the liquid hold-up α (found in section 4.4.2) at the inlet patch, the inlet velocities can be calculated. The velocities are given in table 4.4.1 and are applied to the inlet patch with `fixedValue`.

$$\boldsymbol{u}_l = \left[0, 0, \frac{\dot{Q}_l}{\alpha_l A}\right]^{\mathrm{T}}, \quad \text{with} \quad A = \frac{\pi d^2}{4} \tag{4.4.4}$$

$$\boldsymbol{u}_g = \left[0, 0, \frac{\dot{Q}_g}{(1-\alpha_l) A}\right]^{\mathrm{T}} \tag{4.4.5}$$

Table 4.4.1: Overview of the inlet velocities given volumetric flow rate.

$\dot{Q}_l$ [m^3 h^{-1}]	$\dot{Q}_g$ [m^3 h^{-1}]	α_l [-]	h_l [mm]	$u_{z,l}$ [m s^{-1}]	$u_{z,g}$ [m s^{-1}]
1.0	63.2	0.120	8.86	1.18	10.2
2.0	62.2	0.204	12.9	1.37	11.1
3.0	61.2	0.271	15.8	1.57	11.9

The velocities for the liquid and the gas are not known at the outlet patch. A hybrid `zeroGradient/fixedValue` boundary condition is applied. This patch type is available in OpenFOAM 3.0.x under the name `pressureInletOutletVelocity`. From the pressure field, the solver decides whether there is inflow or outflow on the patch. When there is flow out of the domain, the `zeroGradient` boundary condition is applied. When there is inflow, a `fixedValue` is used.

4.4.4 Pressure

At the inlet patch the pressure is not yet known, thus a `fixedFluxPressure` boundary condition is applied. At the outlet the pressure is fixed with `prghPressure`. An arbitrary value can be chosen, due to the use of incompressible solvers in this project. The reference pressure on the outlet patches is taken as $p_{ref} = 0$ [Pa].

4.5 PROPERTIES OF FLUIDS

In this thesis fluid flows of water and air are analysed. The behaviour of multiphase flow is partly determined by the properties of these fluids. Therefore it is of importance to state the fluid properties, which is done in this section. The flow will be a combination of air and water with properties as stated in table 4.5.1.

Table 4.5.1: Properties of fluids simulated.

Fluid	ρ [kg m^{-3}]	μ [kg s^{-1} m^{-1}]	ν [m^2 s^{-1}]
Water	999	$1.30 \cdot 10^{-3}$	$1.30 \cdot 10^{-6}$
Air	1.25	$1.78 \cdot 10^{-5}$	$1.42 \cdot 10^{-5}$

The mean drop diameter of water is found as $\mathcal{O}(3 \cdot 10^{-4})$ [m] with a maximum droplet diameter of $\mathcal{O}(2 \cdot 10^{-3})$ [m] [90]. In the simulations a value of $d_l = 3 \cdot 10^{-4}$ [m] is used. The `multiphaseEulerFoam` solver requires a typical diameter for both the dispersed and continuous phase, therefore a typical air bubble diameter has to be found. Pohorecki et al. [60] estimate the bubble diameter by the following formula:

$$d_{32} = 0.289\rho_l^{-0.552}\mu_l^{-0.048}\sigma^{0.442}\left(\frac{\dot{Q}_g}{A}\right)^{-0.124} = \mathcal{O}(2\cdot 10^{-3})\ [\mathrm{m}] \tag{4.5.1}$$

Here d_{32} is the Sauter mean diameter [71, p. 45].

4.6 TURBULENCE DESCRIPTION

The user has to describe the turbulence settings in the `constant/turbulenceProperties` file. It is of importance to notice that in this report LES is used due to the fact that `multiphaseEulerFoam` has no native support for RANS. In section 2.2.1.3 the theoretical foundations of LES are explained. In all simulations the Smagorinsky SGS model is used. As mentioned before, this model has several shortcomings, but is chosen due to its stability. In chapter 7 the choice of SGS is mentioned again and the recommendation of testing the `multiphaseEulerFoam` solver with WALE is made.

A filter width Δ has to be defined in order to enable LES in the simulations. A box filter is used, which is given by the `cubeRootVol` word for `delta` in the `constant/turbulenceProperties` file. From section 2.2.1.3 the box filter kernel is recalled as:

$$G(r) = \frac{1}{\Delta}H\left(\frac{1}{2}\Delta - |r|\right) \tag{4.6.1}$$

The Smargorinsky turbulent viscosity is then recalled as:

$$\nu_t = c_s\Delta^2\sqrt{2\boldsymbol{S}_{ij}\boldsymbol{S}_{ij}} \tag{4.6.2}$$

This completes the system of equations. Next section concludes this chapter with an overview of the time stepping and the CFL condition. In the appendices the case files for both the `interFoam` and `multiphaseEulerFoam` solver are given. With these files the simulations performed can be reproduced.

4.7 TIME STEPPING AND THE COURANT-FRIEDRICHS-LEWY CONDITION

In order to correctly simulate flows a maximum time step Δt_{max} has to be imposed, as described in the paper of Courant et al. [18] and given in equation 4.7.1.

CFL condition for problems in $\mathbb{R}^3$ [18]

$$\mathrm{Co} = \Delta t\sum_{i=1}^{3}\frac{u_i}{\Delta x_i} \leq \mathrm{Co}_{max} \tag{4.7.1}$$

Where:

- Co Courant number [-]

In explicit solvers, such as time marching methods, e.g. Euler discretisation schemes, a hard limit on the time step is given by the CFL condition:

$$\mathrm{Co}_{max} = 1 \tag{4.7.2}$$

Essentially this means that a fluid parcel with speed $\boldsymbol{u}(t, \boldsymbol{x})$ can only travel through one cell per time step. Note that the CFL stability criterion is a necessary one, not sufficient for stability. Various simulations require stricter limits on Co_{max}. For instance, at interfaces in vertical risers the maximum Courant number is often taken as 0.25 [17, 21, 83, 93] in order to obtain numerical stability. This thesis follows that last recommendation and uses a Co_{max} of 0.25.

Both solvers can use runtime adjustable time steps in order to cope with fluctuating velocities in the computational domain. By letting the user set the maximum Courant number Co_{max} the variable maximum time step is obtained as follows:

$$\Delta t_{max} \leq \min \left\{ \Delta t_u, \frac{\mathrm{Co}_{max}}{\sum_{i=1}^{3} \frac{u_i}{\Delta x_i}} \right\} \tag{4.7.3}$$

Where:

- Δt_u the user described maximum time step, which acts as a fail safe if Co → ∞.

4.8 CONCLUDING REMARKS ON SIMULATION SET-UP

In this chapter the simulation set-up were presented. With the governing equations for the solvers of OpenFOAM from the previous chapter and the meshed geometry and the boundary conditions from this chapter a complete set-up for the simulation case has been made which is used to perform the simulations.

SHM is used to mesh the geometry of the riser system and the T-junction. As indicated, mesh quality controls are very important in order to obtain converged solutions. Quality indicators such as maximum non-orthogonality and maximum skewness are explained. Wall layers are added to the mesh in order to set the y^+ value such that wall functions can be used.

Boundary conditions for α_k, p, ν_t and $\boldsymbol{u}_k$ are derived in this chapter. Correct boundary conditions are of importance, because a faulty description of the case leads to wrong results or non-converging solutions. The no-slip boundary condition at the wall is applied for $\boldsymbol{u}_{wall,k}$. From there, a constant value for the derivative of the pressure at the wall is derived. The behaviour of ν_t at the wall follow from the `nutkWallFunction` patch. The derivative of the phase fraction at the wall is zero.

In order to have a stable solution, an inlet velocity $\boldsymbol{u}_{inlet,k}$ is coupled with a constant derivative of p at the inlet. At the outlet the pressure is set to $p = 0$, this is allowed because both solvers use incompressible calculations. With the `pressureInletOutletVelocity` patch type the derivative at the outlet for $\boldsymbol{u}_{outlet,k}$ is set to zero. Table 4.8.1 provides an overview of the used boundary conditions in OpenFOAM.

Table 4.8.1: Boundary conditions used in the simulations.

Variable	Inlet	Outlet	Walls
α	`groovyBC*`	`inletOutlet`	`zeroGradient`
ν_t	`calculated`	`calculated`	`nutkWallFunction`
p	`fixedFluxPressure`	`prghPressure`	`fixedFluxPressure`
U	`flowRateInletVelocity`	`pressureInlet-` `OutletVelocity`	`fixedValue`

The properties of the fluids air and water are provided in section 4.5. Together with the density ρ_k for each phase, the viscosity ν_k is provided. Furthermore, a characteristic droplet diameter d_k is given for each phase. This droplet diameter is used in the constant/transportProperties file to calculate the drag between phases.

The last important set of parameters is given in the constant/turbulenceProperties file. These turbulent settings initialise LES and prescribe the filtering used. The filter kernel $G(x, r)$ is recalled, together with the turbulent eddy viscosity used in the simulations.

Now that the governing equations, geometry and boundary conditions are known, the actual simulations can commence. The next chapter will provide the results of the simulations. Also mesh convergence checks will be made. After that, the interFoam and multiphaseEulerFoam solvers will be validated against ANSYS FLUENT results for the same geometry.

Chapter 5

Analysis of results

This chapter presents the simulation results as obtained with the OpenFOAM solvers `interFoam` and `multiphaseEulerFoam`. Visual representations of the investigated systems will be alternated with plots from the post processing done in Matlab. As explained earlier, the integral of the outflow of the two phases equals the volumetric flow rate. A package called SWAK for FOAM is used to calculate the flow rates on the inlet and outlet patches. The focus is on the meshing results, the mesh dependency and the qualitative differences between the solvers. The goal of this chapter is to analyse the developed flow patterns, pressure at the inlet and production at the outlets, such that there is enough confidence to proceed with a comparison between the results of `interFoam` and `multiphaseEulerFoam` in the next chapter.

5.1 RESULT OF MESHING OPERATIONS

The meshing procedure set-up is explained in section 4.2. This section provides an analysis of the meshes built with SHM. Five meshes are produced, which are tested by multiple solvers, using various initial conditions. Three of these five meshes are **T1**, **T2** and **T3**, cover the T-junction. The other two meshes are made from the total riser geometry (figure 4.1.1) and are from now on called **R1** and **R2**. Additional remarks on the non-symmetrical meshing of SHM are made at the end of this section.

5.1.1 Meshing results of T-junction

A mesh dependency test is performed to verify whether the results are influenced by the fineness of the mesh. The T-section is meshed through three variants as shown in table 5.1.1.

Table 5.1.1: Properties of cells in the three T-meshes.

Mesh	Cells (% of T2)		Avg. cell volume [m^3]
T1	23042	(48%)	$9.50 \cdot 10^{-8}$
T2	47544	(100%)	$4.60 \cdot 10^{-8}$
T3	95292	(200%)	$2.30 \cdot 10^{-8}$

The meshed T-junction is displayed in figure 5.1.1. As can be seen from the graphs, the coarse mesh **T1** has only one layer of wall cells. This is due to the fact that the wall thickness is doubled for this mesh to better match the dimensions of the inner cells.

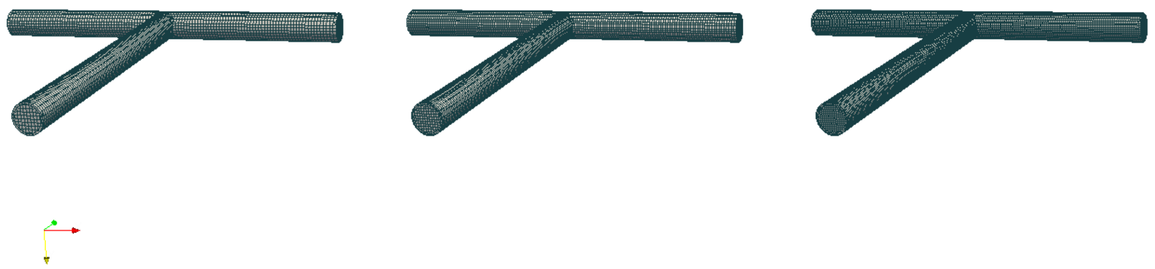


Figure 5.1.1: Comparison of the three T-junction meshes. From left to right T-mesh with 23042, 47544 and 95292 cells.

All mesh quality parameters explained in previous chapter are the same for the three meshes. With the built-in OpenFOAM utility `checkMesh` the mesh quality parameters can be found. The leading parameters are given in table 5.1.2.

Table 5.1.2: Mesh quality of the three T-meshes. (*AR = Aspect Ratio)

Mesh	Avg. non-orthogonality	Max. non-orthogonality	Max. skewness	Max. AR*
T1	6.9	59.1	2.4	8.9
T2	6.3	55.7	2.0	4.9
T3	5.4	55.8	1.2	6.2

5.1.2 Meshing results of riser section

Equal meshing settings are used in SHM in order to mesh the riser system, although the bounding box is extended in the negative y-direction. Adding layers to the walls of this geometry proved to be difficult, mainly due to the elbows. In these elbows the walls are not normal to the directions from the base mesh of `blockMesh`, therefore careful meshing was needed. Two meshes are produced, **R1** and **R2**, both with 100% of the layers added to the walls. The properties of the riser system meshes are given in table 5.1.3.

Table 5.1.3: Properties of cells in the two riser meshes.

Mesh	Cells (% of T2)	Avg. cell volume $[\mathrm{m}^3]$
R1	67720 (24%)	$18.9 \cdot 10^{-8}$
R2	276784 (100%)	$4.60 \cdot 10^{-8}$

Running the `checkMesh` utility on both riser system meshes provides the non-orthogonality and skewness of the meshes, these parameters are listed in table 5.1.4. The non-orthogonality is higher, which is due to the addition of the elbow parts.

Table 5.1.4: Mesh quality of the two riser system meshes. (*AR = Aspect Ratio)

Mesh	Avg. non-orthogonality	Max. non-orthogonality	Max. skewness	Max. AR*
R1	10.6	51.4	1.4	6.4
R2	6.8	60.5	2.3	9.7

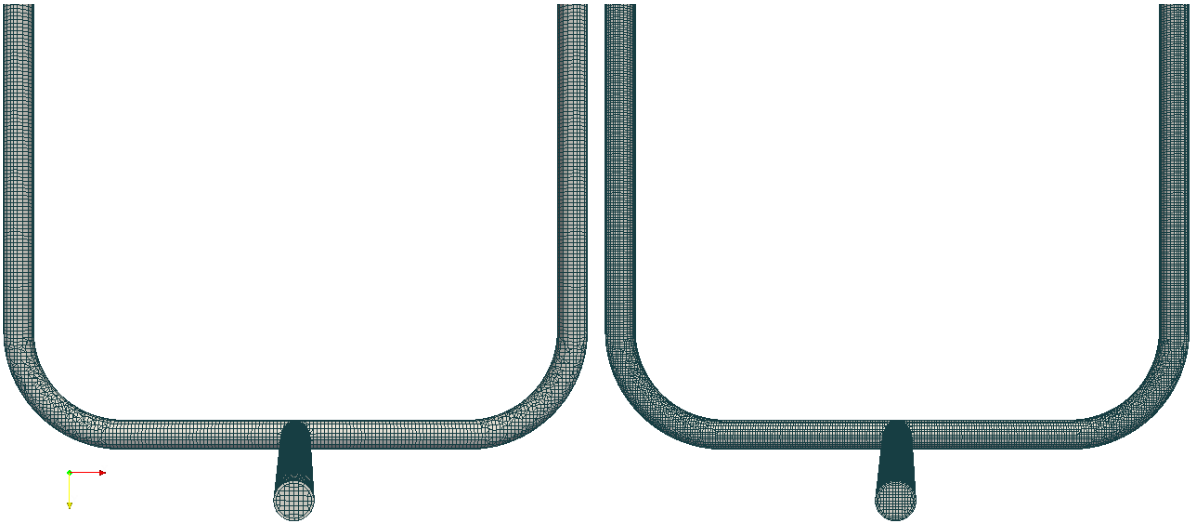


Figure 5.1.2: Comparison of (part of) the coarse mesh **R1** (left) and medium mesh **R2** (right).

5.1.3 Additional remarks with respect to the meshing results

Due to the nature of the SHM code, it is more likely for the program to output non-symmetrical meshes. Therefore, in situations where symmetry would be expected, discrepancies can appear. However, the differences between outlet parameters are small. By using the crinkle cut tool in ParaView, the non-symmetry is visualised. In figure 5.1.3 a cut is made through the yz-plane, with midpoint $(0, 0, 0)$.

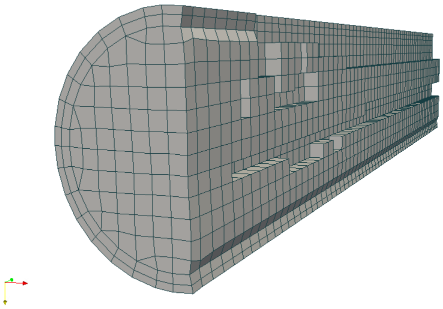


Figure 5.1.3: Crinkle cut to visualise non-symmetry in the SHM mesh generation procedure.

5.2 Mesh dependency analysis

The previous section introduced the five meshes used in this mesh dependency test. Three T-junction meshes are made: **T1**, **T2** and **T3**. Two riser system meshes are generated by SHM: **R1** and **R2**. In this section first an analysis of the T-junction mesh is given. This section is concluded with an analysis of the two riser meshes. Similar settings in terms of base mesh are used for mesh **T2** and **R2**, therefore **R2** can be expected to perform well when **T2** performs well.

5.2.1 Mesh dependency in T-junction

In order to test the effect of the fineness of the mesh on the solution, a mesh dependency test in `multiphaseEulerFoam` for the T-junction is done. Such a test is valuable, because it will tell which minimum number of mesh cells still gives a good accuracy. Three key output parameters are investigated in this section: the calculated inlet pressure, the liquid production at the left outlet and the production of air at the left outlet.

5.2.1.1 Mesh dependency of the inlet pressure

As described in the previous chapter, the boundary condition of the inlet is set as `fixedFluxPressure`. By using this boundary condition the derivative of the pressure is set equal to the body forces. Since the derivative is fixed, the actual value is calculated by OpenFOAM.

The pressure is sampled after the simulation is completed and this is done by averaging the pressure field `p` over the `inlet` patch. A small chain of `bash` commands is written to automatise the output of the pressure field to a file called `pressures.txt`. This chain of commands will also output a `times.txt` file in which the available time steps are given in vector form.

```
patchAverage p inlet | grep -oP ’(?<=\[0\] = )[\- 0-9.]+’ > pressures_x.txt;
ls -d */ | grep ’[0-9]’ | sort -n | tail -n +2  | sed ’s/.$//’ > times.txt;
paste times.txt pressures_x.txt > pressures.txt; rm -rf pressures_x.txt
```

To smooth out pressure spikes in the calculations a moving average is used. Matlab provides the inbuilt function `movmean` for this purpose. A window of 1 [s] is taken to show the general development of the pressure, while not losing too many details. In figure 5.2.1 an example of the effect of the `movmean` function is given.

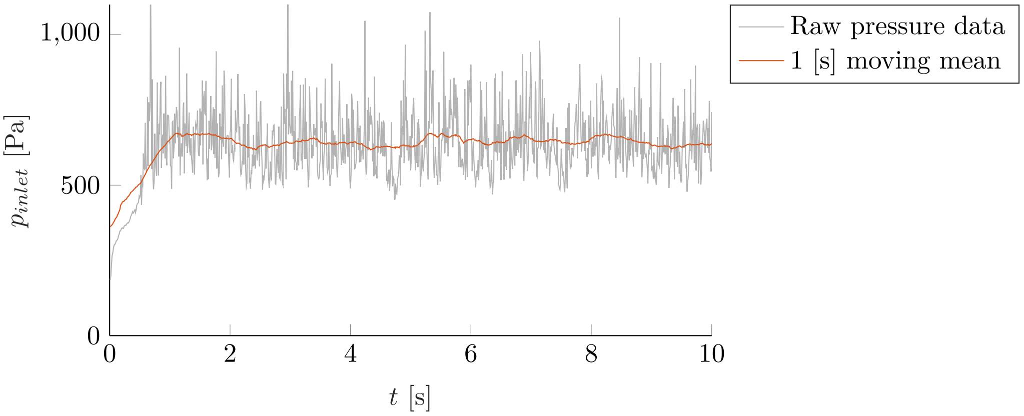


Figure 5.2.1: Example of the effect of the `movmean` function on the raw pressure data. A `multiphaseEulerFoam` simulation on mesh **T2** with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa] between the outlets.

The average pressure at the inlet, sampled between $t = 2$ [s] and $t_{end} = 10$ [s], are summarised in table 5.2.1. A surprising finding is that in more than half of the cases the pressure is lower at the coarsest mesh **T1** and higher in the medium mesh **T2** as compared to the prediction at the finest mesh **T3**. Furthermore, the largest oscillations are found in the results of mesh **T2**.

Table 5.2.1: Mean and standard deviation of p_{inlet} for three T-junction meshes. Samples taken from $t = 2$ [s] to $t_{end} = 10$ [s].

		Mesh T1		**Mesh T2**		**Mesh T3**	
$\dot{Q}_{l,in}$ [m^3 h^{-1}]	Δp [Pa]	$\mu_{p_{inlet}}$	($\sigma_{p_{inlet}}$) [Pa]	$\mu_{p_{inlet}}$	($\sigma_{p_{inlet}}$)[Pa]	$\mu_{p_{inlet}}$	($\sigma_{p_{inlet}}$) [Pa]
1	0	232	(35)	221	(22)	187	(25)
1	250	375	(17)	393	(26)	363	(27)
2	0	406	(55)	491	(104)	438	(89)
2	250	546	(49)	643	(106)	598	(93)
3	0	647	(140)	871	(236)	770	(214)
3	250	780	(138)	992	(252)	907	(204)

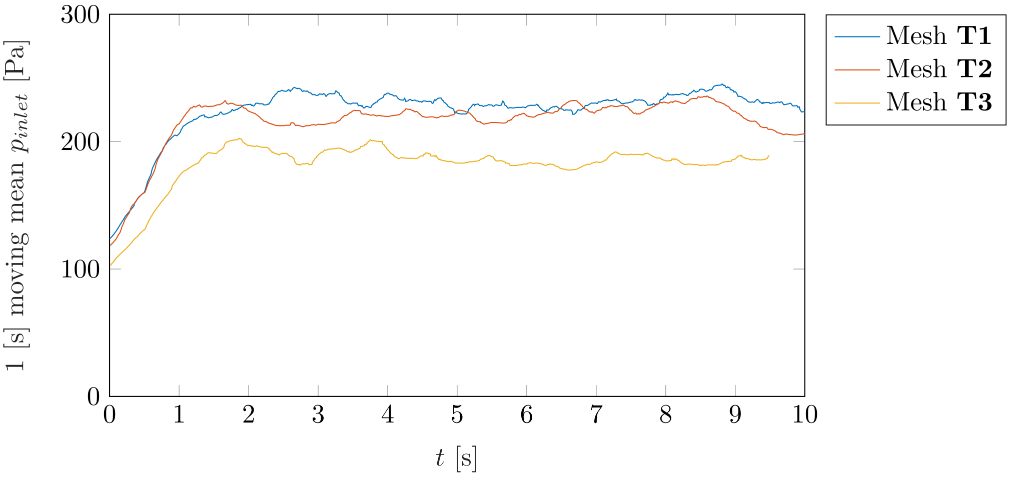


Figure 5.2.2: The inlet pressure for three meshes with $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

When the volumetric flow rate of water at the inlet is increased, the results for the inlet pressure on the finest mesh are in between values obtained on the coarse and medium meshes. Overall, the finest mesh **T3** has a quite stable profile, as can also been seen from table 5.2.1. In figures 5.2.2 to 5.2.4 the results for various configurations are displayed.

The main conclusion of this mesh dependency test with these three meshes is that the coarse (**T1**) and medium (**T2**) mesh do not smoothly approach the pressures from the fine (**T3**) mesh, a ±10% discrepancy is found. This indicates that the applied meshes are not fine enough yet to give results in the asymptotic convergence range.

5.2.1.2 Mesh dependency of the left outlet liquid production

Post-processing of the volumetric flow rates is done by using a program called SWAK for FOAM. The run-time post-processing of the flow rates is obtained by using the following expression on the patches:

$$\dot{Q}_{k,p} = \sum_{i=1}^{\#Cells} \alpha_{i,k} \left(\boldsymbol{u}_{i,k} \cdot \boldsymbol{n} \right) \tag{5.2.1}$$

Where:

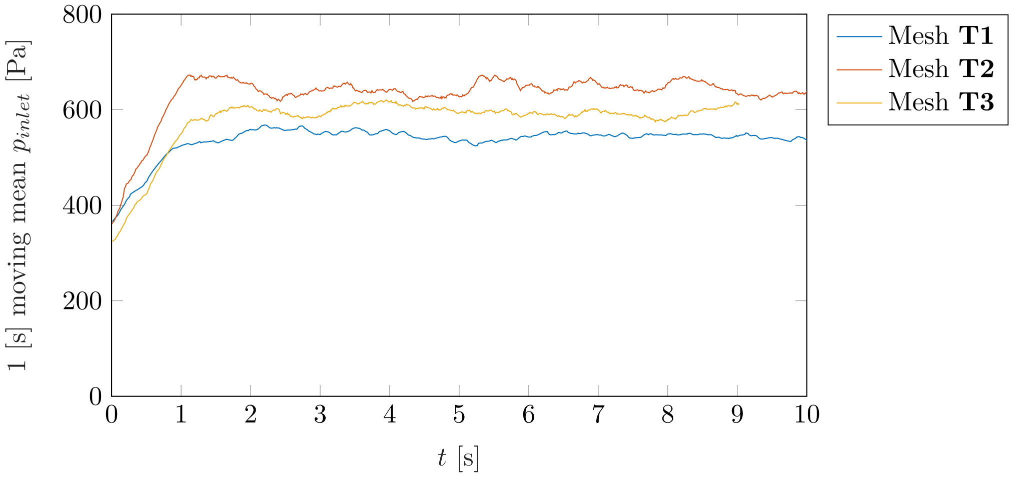


Figure 5.2.3: The inlet pressure for three meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

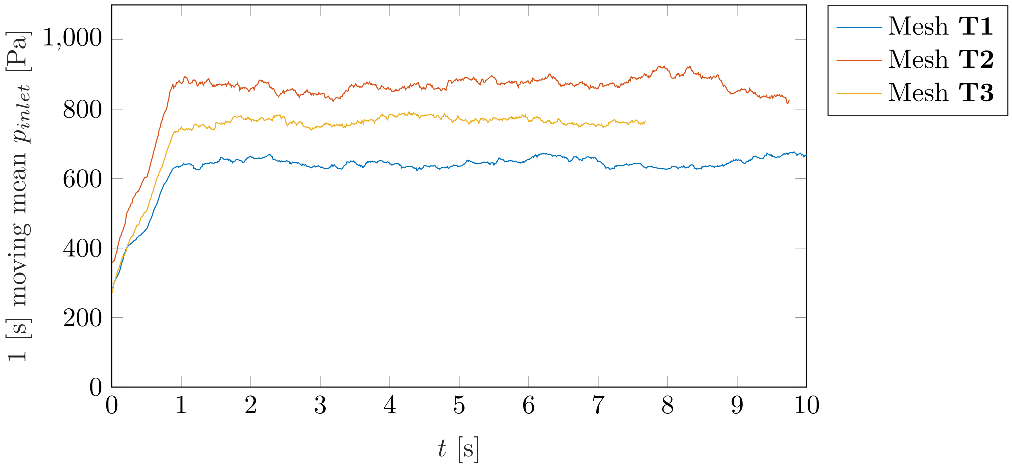


Figure 5.2.4: The inlet pressure for three meshes with $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

- i Iterative variable over all cell faces on the patch p
- $\alpha_{i,k}$ Phase fraction of phase k in cell i
- $\boldsymbol{u}_{i,k}$ Velocity of phase k in cell i
- $\boldsymbol{n}$ Normal vector w.r.t. the orientation of patch p

An expression for $\dot{Q}_{l,left}$ and $\dot{Q}_{l,right}$ is found. In order to analyse the production of the six cases, a non dimensional production ratio is defined in equation 5.2.2. This is done to compare cases with different phase-specific volumetric flow rates.

Definition of the phase ratio of fluid k at outlet and inlet

$$R_{k,p} \equiv \frac{\dot{Q}_{k,p}}{\dot{Q}_{k,inlet}} \tag{5.2.2}$$

In table 5.2.2 the results for the liquid production at the left outlet are summarised. Contrary to the pressure calculations, the production values at the medium mesh **T2** are much closer to the results of the fine mesh. The differences in the production between the medium and fine meshes are between 0.3% and 2%. The differences between the coarse mesh and the fine mesh are between 0.4% and 12.3%. Especially when a pressure difference exists between the junction outlets, the discrepancy between the results of those meshes is large.

Table 5.2.2: Mean and standard deviation of $\dot{Q}_{l,left}$ for three T-junction meshes. Samples taken from $t = 2$ [s] to $t_{end} = 10$ [s].

		Mesh T1		Mesh T2		Mesh T3	
$\dot{Q}_{l,in}$ [m^3 h^{-1}]	Δp [Pa]	$\mu_{R_{l,left}}$	$(\sigma_{R_{l,left}})$ [-]	$\mu_{R_{l,left}}$	$(\sigma_{R_{l,left}})$ [-]	$\mu_{R_{l,left}}$	$(\sigma_{R_{l,left}})$ [-]
1	0	0.50	(0.17)	0.47	(0.16)	0.50	(0.12)
1	250	0.70	(0.24)	0.63	(0.23)	0.63	(0.20)
2	0	0.51	(0.13)	0.50	(0.12)	0.50	(0.12)
2	250	0.57	(0.19)	0.55	(0.16)	0.55	(0.16)
3	0	0.50	(0.13)	0.50	(0.17)	0.49	(0.14)
3	250	0.55	(0.15)	0.52	(0.18)	0.53	(0.17)

In figures 5.2.5 to 5.2.7 the normalised production values at the left outlet are displayed between $t = 2$ [s] and $t_{end} = 10$ [s]. The liquid production at the coarse mesh, **T1**, exhibits a more oscillating behaviour than at the other meshes. Especially in figure 5.2.5 the large discrepancy between the prediction of the solutions can be seen.

Overall it can be concluded that the medium mesh can accurately predict the liquid output. Further mesh refinement to investigate the production of water at the outlets is thus not needed. The coarse mesh shows errors up to 12% for the prediction of the liquid output and care should be taken when analysing the results obtained with this mesh.

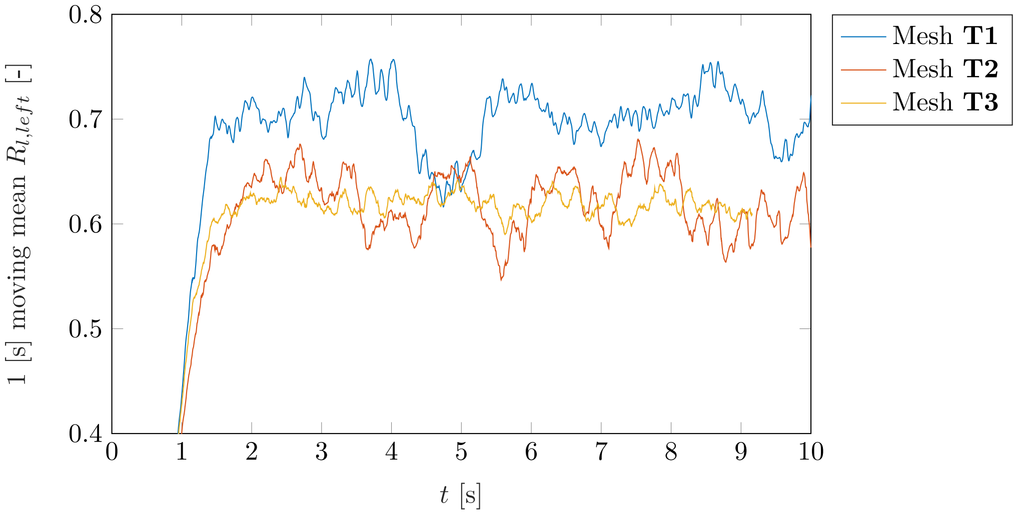


Figure 5.2.5: Production of water at the left outlet for three meshes with $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

5.2.1.3 Mesh dependency of the gas production

To conclude this mesh dependency test, the output of air through the left outlet is analysed for the three T-junction meshes. Interestingly enough, in most cases even the production at the coarsest mesh is close to the value of of the finest mesh, as can also be seen in table 5.2.3. The only case where the deviation between the coarse mesh and the other meshes is large is for the case with $\dot{Q}_{l,inlet} = 1$ [m^3

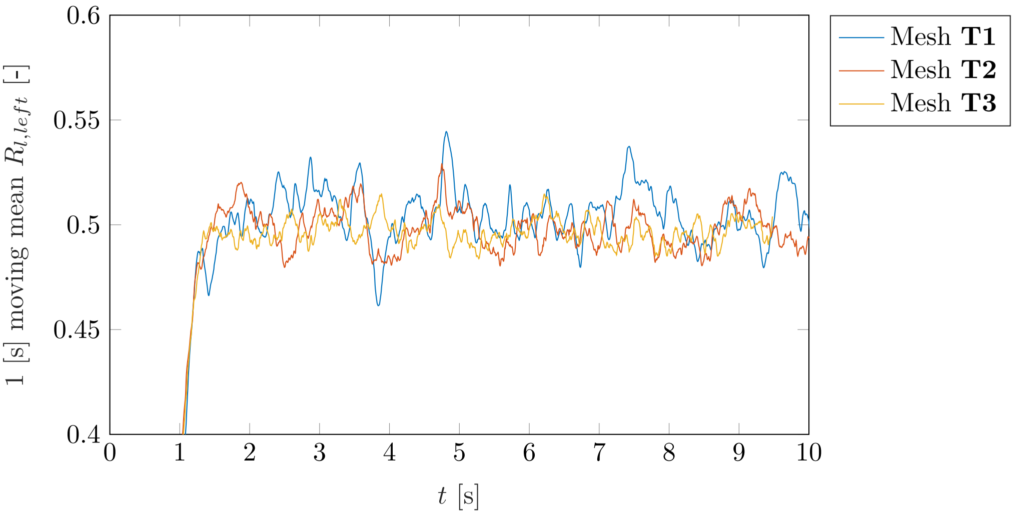


Figure 5.2.6: Production of water at the left outlet for three meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

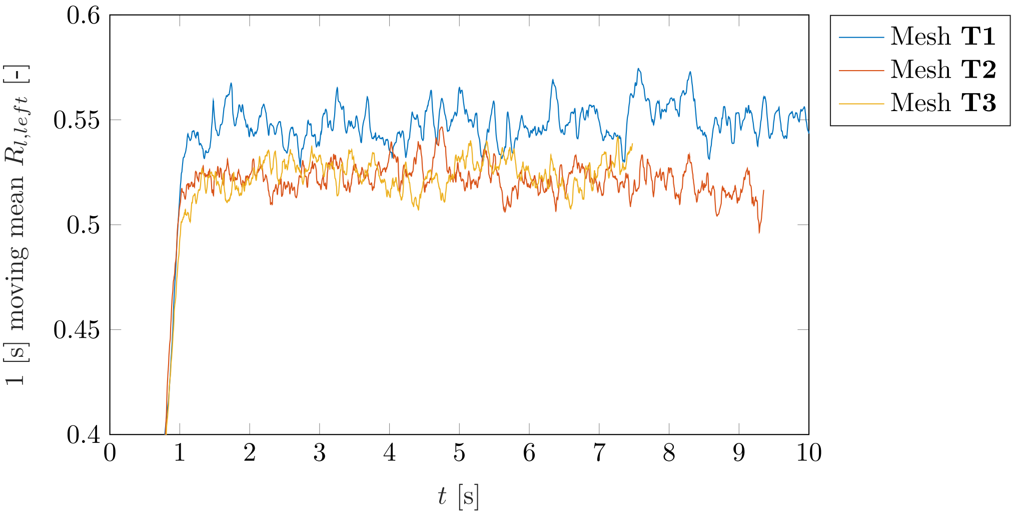


Figure 5.2.7: Production of water at the left outlet for three meshes with $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

h^{-1}], $\Delta p = 250$ [Pa] case. In figure 5.2.8 a clear difference between the production of air at the outlet can be seen. The normalised gas production plots is shown in figures 5.2.8 to 5.2.10.

5.2.1.4 Mesh dependency of the liquid hold-up

The liquid hold-up in the T-junction is related to the inlet pressure. As can be seen from figures 5.2.11 to 5.2.13. When no pressure difference between the outlets is applied the liquid hold-up in the domains is similar. However, when a pressure difference of $\Delta p = 250$ [Pa] between the outlets is applied, there is a small difference between the results from the meshed. The finest mesh, gain, predicts a liquid hold-up in between the values of the other meshes. The flow regime is very important for both the results of the inlet pressure and the liquid hold-up. Additional simulations with even finer meshed have to be performed in order to draw firm conclusions whether the **T3** predicts the right flow regime.

Table 5.2.3: Mean and standard deviation of $\dot{Q}_{g,left}$ for three T-junction meshes. Samples taken from $t = 2$ [s] to $t_{end} = 10$ [s].

		Mesh T1		**Mesh T2**		**Mesh T3**	
$\dot{Q}_{l,in}$ [m^3 h^{-1}]	Δp [Pa]	$\mu_{R_{g,left}}$	($\sigma_{R_{g,left}}$) [-]	$\mu_{R_{g,left}}$	($\sigma_{R_{g,left}}$) [-]	$\mu_{R_{g,left}}$	($\sigma_{R_{g,left}}$) [-]
1	0	0.50	(0.015)	0.50	(0.014)	0.50	(0.023)
1	250	0.86	(0.041)	0.79	(0.041)	0.79	(0.031)
2	0	0.50	(0.023)	0.50	(0.021)	0.50	(0.017)
2	250	0.66	(0.044)	0.63	(0.032)	0.66	(0.026)
3	0	0.50	(0.029)	0.50	(0.030)	0.50	(0.023)
3	250	0.60	(0.033)	0.62	(0.033)	0.61	(0.028)

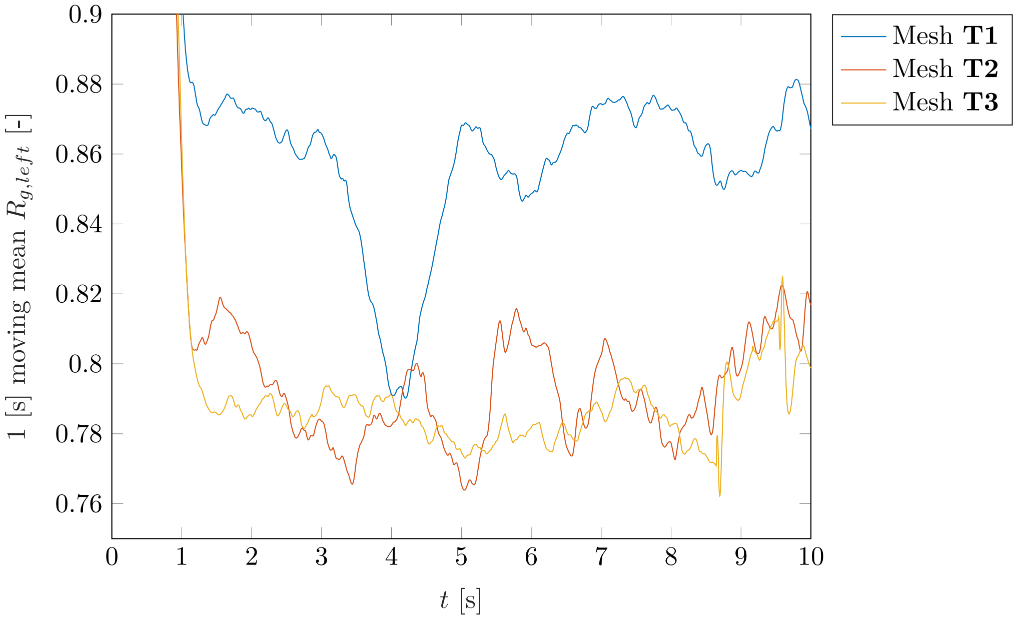


Figure 5.2.8: Production of air at the left outlet for three meshes with $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

5.2.2 Mesh dependency for the riser system

Due to the small number of simulations in the riser system and the long time needed to run these simulations, this mesh dependency section is kept concise. This section will be expanded when the simulations are finished. One can see that the calculated inlet pressure for the **R2** riser is a bit lower than the values at the **R1** mesh, but overall the development and shape are similar. In terms of production it is hard to draw strong conclusions due to the lack of data. Longer simulation times must prove whether simulations on the **R1** mesh are similar to the simulations on mesh **R2**.

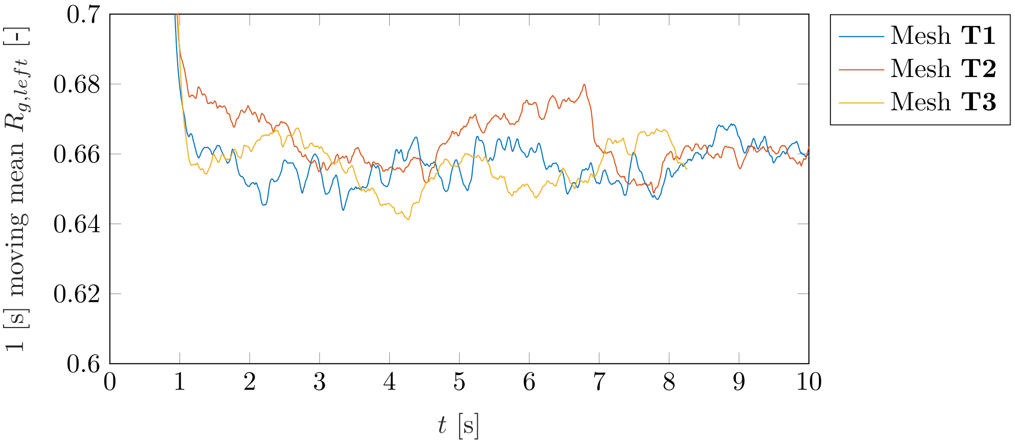


Figure 5.2.9: Production of air at the left outlet for three meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

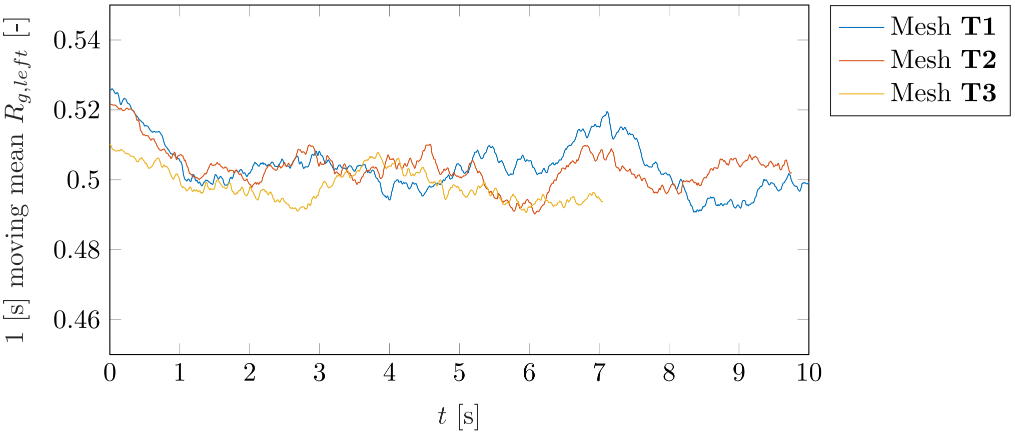


Figure 5.2.10: Production of air at the left outlet for three meshes with $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

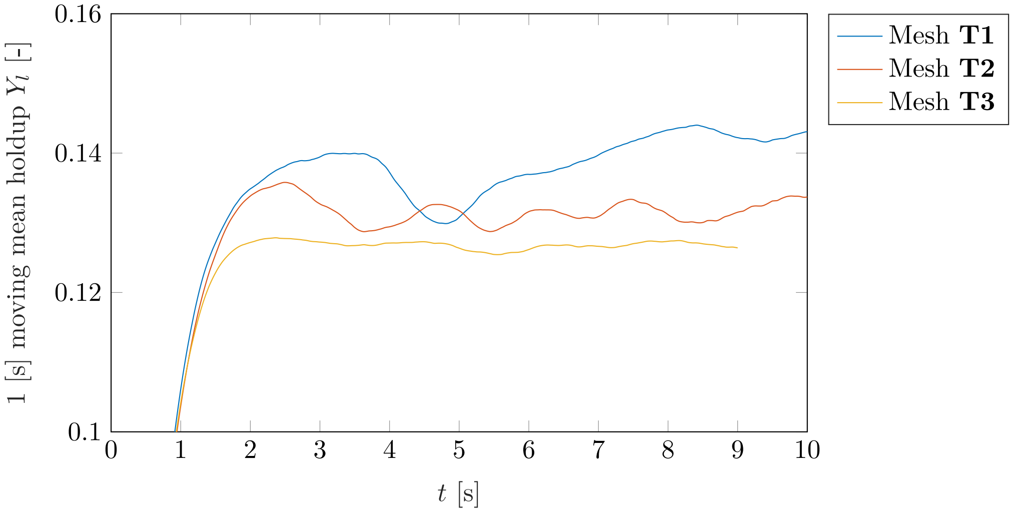


Figure 5.2.11: Liquid hold-up in the junction for three meshes with $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

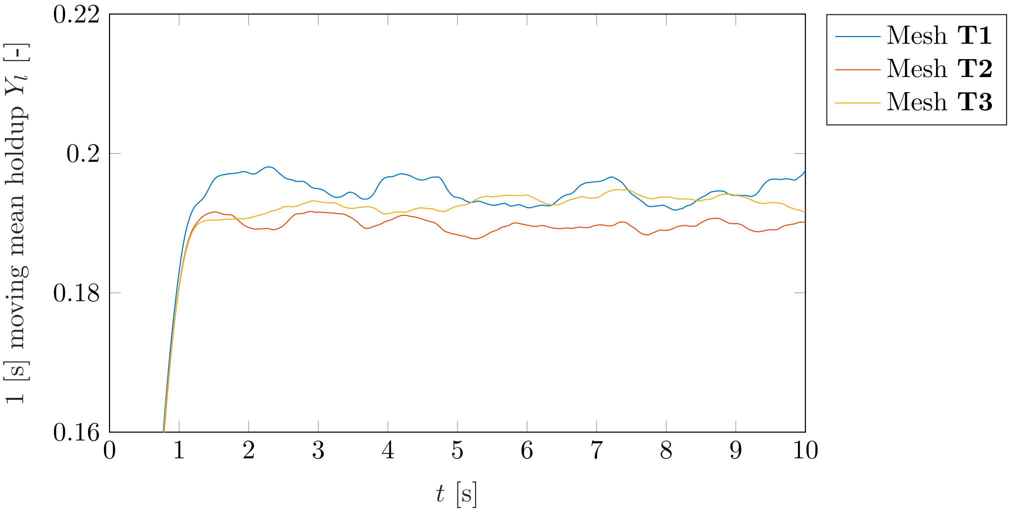


Figure 5.2.12: Liquid hold-up in the junction for three meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

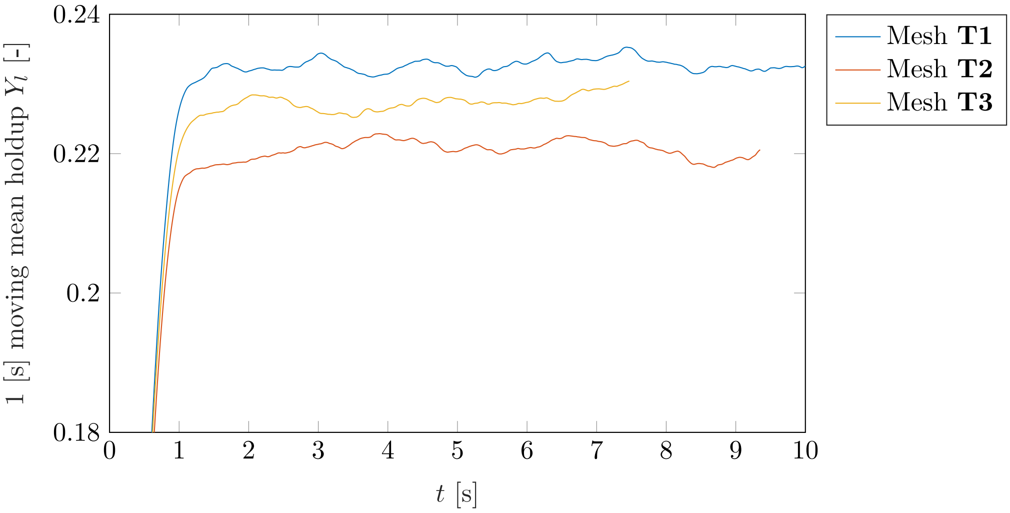


Figure 5.2.13: Liquid hold-up in the junction for three meshes with $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

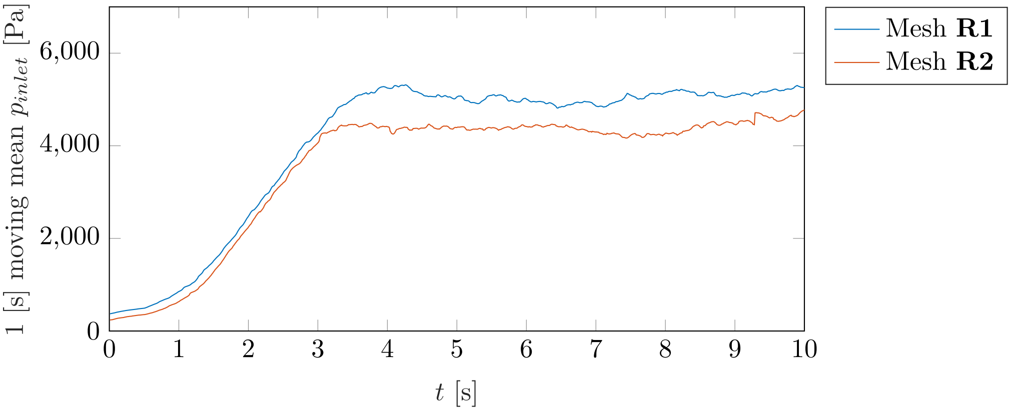


Figure 5.2.14: The inlet pressure for two meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

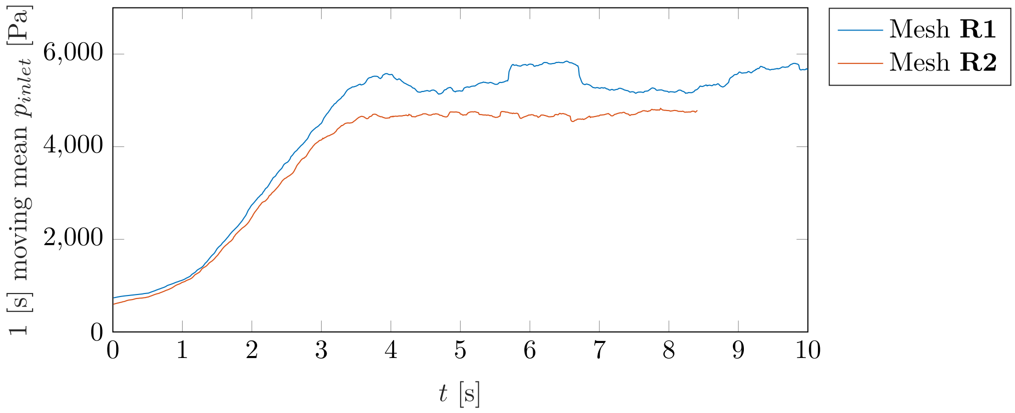


Figure 5.2.15: The inlet pressure for two meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 500$ [Pa]

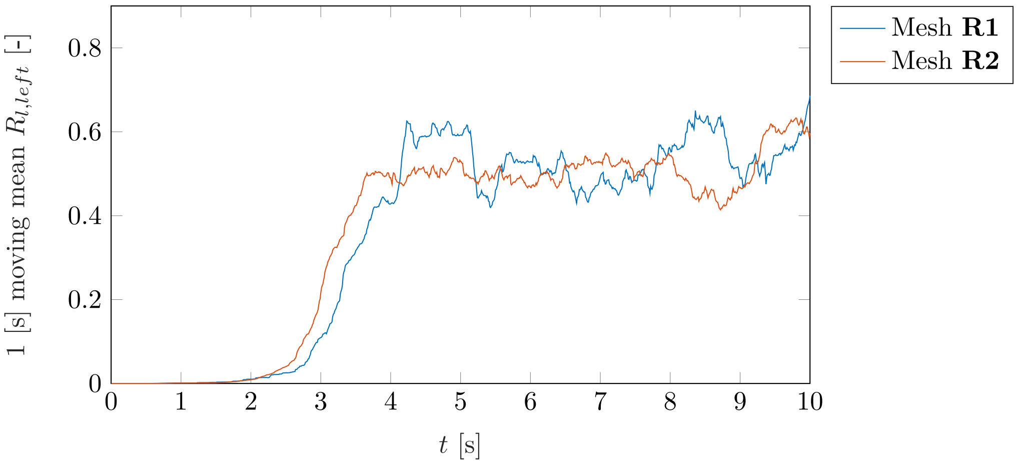


Figure 5.2.16: Liquid production at the left outlet for two meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

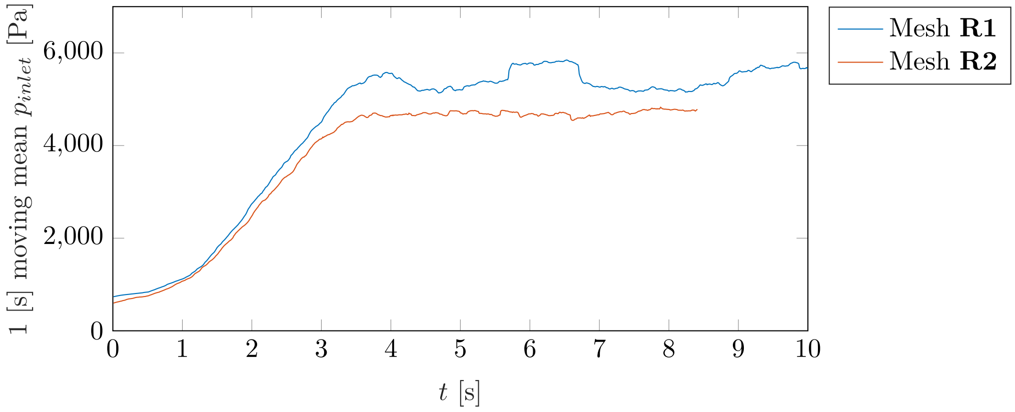


Figure 5.2.17: Liquid production at the left outlet for two meshes with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 500$ [Pa]

5.3 Analysis of interFoam results

By using `interFoam`, results for the T-junction and for the large riser geometry are obtained. In this section the results are analysed in both a quantitative and qualitative way.

5.3.1 T-junction

The T-junction geometry is simulated at mesh **T2** in order to compare them later in this report to the results from the `multiphaseEulerFoam` solver. In figure 5.3.1 an example of a result from `interFoam` is shown. This T-junction, with mesh **T2** is shown at $t = 1$ [s]. As can be seen from the figure, steady outflow is already established at this point in time. It is interesting that flow pattern in the inlet section is stratified. No clear waves are generated in this part of the domain. Right before the junction some waves are developed. Although the flow pattern is predicted to be slug flow or stratified wavy flow [95], the solver doesgive these patterns. In the work of Worthen [95], ANSYS FLUENT also does not does not give these flow patterns. The main reason is possibly the rather short length of the horizontal pipe, which is insufficient to allow the stratified flow to develop waves or even slugs. A separate study can be performed to study the development of the flow pattern in an inlet pipe, see also chapter 7. The analysis of the inlet length is omitted in this report due to time restrictions. Furthermore figure 5.3.1 shows non-physical droplet formation after the junction. The droplets stick to the wall in an elongated way.

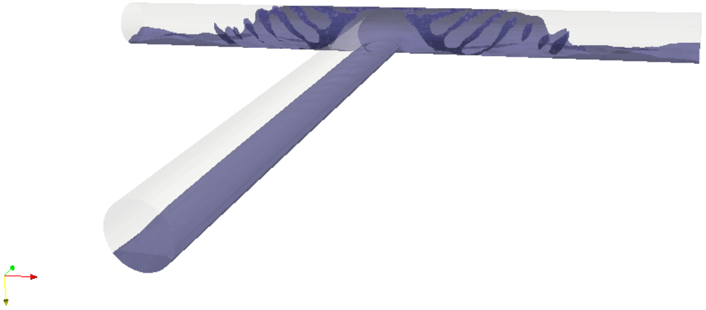


Figure 5.3.1: Snapshot of T-junction with mesh **T2** in `interFoam`, $\dot{Q}_{l,input} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa] and $t = 1$ [s]

One can see that two cases are missing in table 5.3.2: the outlet pressure difference $\Delta p = 375$ [Pa] and $\Delta p = 500$ [Pa] for $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}]. Upon analysing these results it was noted that inflow occurred through the right outlet, while this was prohibited by the outflow boundary condition for the velocity. A `fixedValue` boundary condition is applied on this field when the velocity vector was pointed towards the inside the domain. Unfortunately, the `interFoam` solver uses a linear discretisation at the patches. To satisfy the mass conservation equation, the solver calculates nett inflow of air at the outlet patch for high pressure differences.

In figure 5.3.3 one can see a snapshot of the T-junction with a pressure difference applied over the outlets. A difference in water flow through the two outlet branches can be seen. About 58.7% of the water flows out through the left outlet branch. The air in the domain is even more affected by the pressure difference, roughly two-thirds of the air leaves through the left outlet.

As can be seen from figure 5.3.4 and figure 5.3.5 a convincing flow pattern develops in which relatively large churns of water are transported upwards with the flow. Flow in the T-junction part of the large riser develops in the same way as flow in the T-junction alone. In the elbows the liquid hold up is larger than

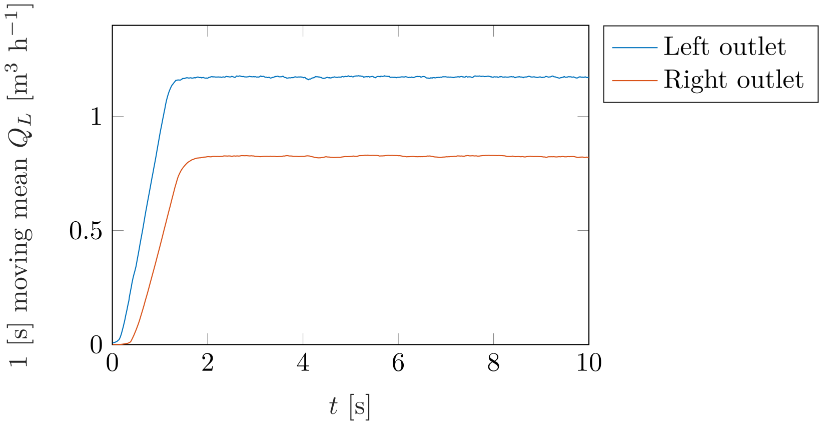


Figure 5.3.2: Production of water at the outlets with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa].

Table 5.3.1: Results for `interFoam` with mesh **T2**, $\dot{Q}_{total} = 64.2$ [m^3 h^{-1}].

$\dot{Q}_{l,in}$ [m^3 h^{-1}]	Δp [Pa]	$\dot{Q}_{l,left}$ [m^3 h^{-1}]		$\dot{Q}_{g,left}$ [m^3 h^{-1}]		$\dot{Q}_{l,right}$ [m^3 h^{-1}]		$\dot{Q}_{g,right}$ [m^3 h^{-1}]	
1	0	0.50	(49.8%)	31.7	(50.1%)	0.50	(50.1%)	31.5	(49.9%)
1	125	0.58	(58.3%)	41.0	(64.8%)	0.42	(41.6%)	22.2	(35.2%)
1	250	0.67	(66.8%)	50.7	(80.3%)	0.33	(33.0%)	12.5	(19.7%)
2	0	1.00	(50.0%)	30.6	(49.2%)	1.00	(49.9%)	31.6	(50.8%)
2	125	1.09	(54.5%)	35.0	(56.3%)	0.91	(45.5%)	27.2	(43.7%)
2	250	1.17	(58.7%)	40.4	(65.0%)	0.83	(41.3%)	21.8	(35.0%)
2	375	1.25	(62.6%)	45.2	(72.6%)	0.75	(37.3%)	17.0	(27.3%)
2	500	1.35	(67.3%)	50.0	(80.4%)	0.65	(32.3%)	12.2	(19.6%)
3	0	1.50	(50.1%)	31.0	(50.7%)	1.49	(49.7%)	30.1	(49.3%)
3	125	1.55	(51.7%)	32.4	(53.0%)	1.45	(48.2%)	28.7	(47.0%)
3	250	1.65	(54.9%)	36.1	(59.0%)	1.35	(45.0%)	25.1	(41.0%)
3	375	1.70	(56.6%)	38.2	(62.4%)	1.30	(43.5%)	23.0	(37.6%)
3	500	1.75	(58.5%)	41.0	(67.1%)	1.24	(41.4%)	20.1	(32.9%)

in other parts of the system. From that area of high hold up the droplets are formed and transported with the air.

5.3.2 Flow split correlation in the riser system with `interFoam`

As mentioned in the introduction of this section, the sampling window of 5 [s] is rather short, therefore errors in the results for the production can be expected. As can be seen from table 5.3.2, the production of the left riser is not strictly increasing with higher Δp over the outlets. Further research is needed to draw a stronger conclusion on the non-equal split of water and air in this system.

Figure 5.3.6 graphically displays the production of liquid through the left outlet, normalised with the volumetric flow rate at the inlet. A polynomial is fitted through the 27 data points which results in the

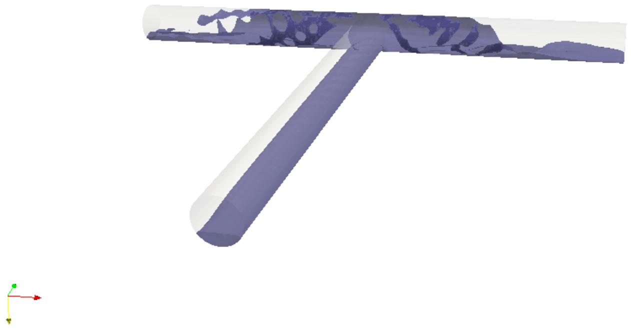


Figure 5.3.3: Snapshot of T-junction with mesh **T2** in `interFoam`, $\dot{Q}_{l,input} = 2$ [m^3 h^{-1}], $\Delta p = 250$ [Pa] and $t = 1$ [s]

Figure 5.3.4: Snapshot of riser system with mesh **R2** in `interFoam`, $\dot{Q}_{l,input} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa] and $t = 5$ [s]

following expression for the liquid production at the left outlet:

$$\frac{\dot{Q}_{l,left}}{\dot{Q}_{l,inlet}} = 0.5027 - 0.01798\,\dot{Q}_{l,inlet} + 0.0001839\,\Delta p + 0.006844\,\dot{Q}^2_{l,inlet} - 4.512 \cdot 10^{-5}\,\dot{Q}_{l,inlet}\Delta p \quad (5.3.1)$$

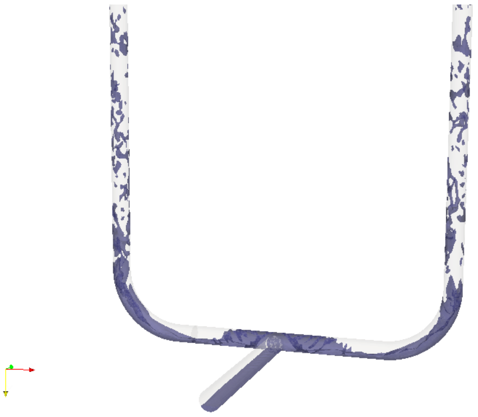


Figure 5.3.5: Zoomed in snapshot of riser system with mesh **R2** in `interFoam`, $\dot{Q}_{l,input} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa] and $t = 5$ [s]

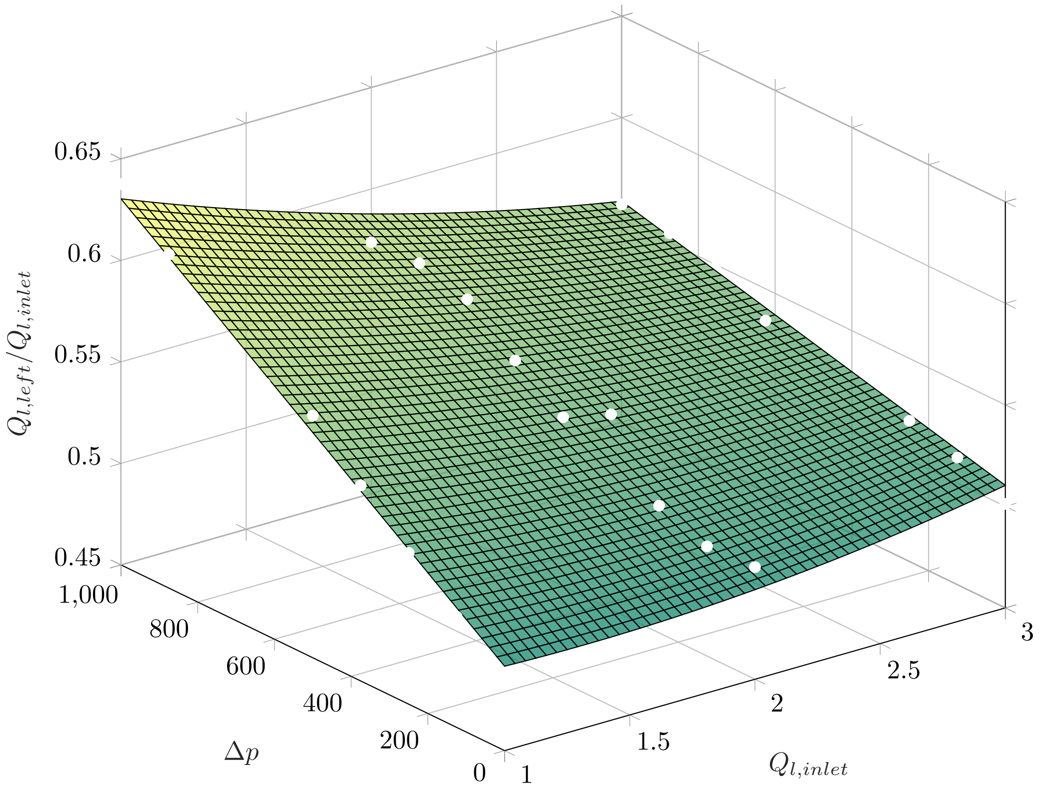


Figure 5.3.6: Normalised liquid production at the left outlet in `interFoam`.

Table 5.3.2: Production in `interFoam` with mesh **R2** from $t = 5$ [s] to $t_{end} = 10$ [s], $\dot{Q}_{total} = 64.2$ [m^3 h^{-1}].

$\dot{Q}_{l,in}$ [m^3 h^{-1}]	Δp [Pa]	$\dot{Q}_{l,left}$ [m^3 h^{-1}]		$\dot{Q}_{g,left}$ [m^3 h^{-1}]		$\dot{Q}_{l,right}$ [m^3 h^{-1}]		$\dot{Q}_{g,right}$ [m^3 h^{-1}]	
1	0	0.49	(48.7%)	31.9	(50.5%)	0.48	(47.9%)	31.3	(49.6%)
1	125	0.50	(50.5%)	33.3	(52.8%)	0.46	(45.7%)	29.9	(47.3%)
1	250	0.52	(52.5%)	34.1	(54.0%)	0.43	(43.4%)	29.1	(46.1%)
1	375	0.55	(54.6%)	35.5	(56.3%)	0.41	(41.4%)	27.7	(43.8%)
1	500	0.57	(56.9%)	36.9	(58.4%)	0.38	(38.4%)	26.4	(41.7%)
1	625	0.57	(57.4%)	37.3	(59.0%)	0.36	(35.5%)	26.0	(41.1%)
1	750	0.59	(59.1%)	38.4	(60.8%)	0.35	(34.5%)	24.8	(39.3%)
1	875	0.61	(61.4%)	39.5	(62.4%)	0.31	(31.2%)	23.8	(37.7%)
1	1000	0.64	(63.8%)	41.4	(65.5%)	0.28	(28.1%)	21.9	(34.6%)
2	0	1.01	(50.5%)	31.1	(49.9%)	1.01	(50.5%)	31.1	(50.0%)
2	125	1.01	(50.4%)	31.6	(50.8%)	0.98	(49.0%)	30.6	(49.2%)
2	250	1.02	(51.3%)	32.0	(51.4%)	0.96	(48.2%)	30.2	(48.6%)
2	375	1.09	(54.6%)	33.3	(53.6%)	0.93	(46.7%)	28.8	(46.4%)
2	500	1.07	(53.3%)	34.0	(54.7%)	0.93	(46.4%)	28.2	(45.3%)
2	625	1.10	(55.0%)	34.5	(55.4%)	0.88	(44.2%)	27.8	(44.6%)
2	750	1.14	(56.8%)	36.0	(57.9%)	0.85	(42.4%)	26.2	(42.1%)
2	875	1.15	(57.5%)	36.7	(59.1%)	0.84	(41.7%)	25.5	(40.9%)
2	1000	1.15	(57.4%)	38.0	(61.1%)	0.77	(38.5%)	24.3	(39.1%)
3	0	1.50	(50.1%)	30.3	(49.5%)	1.53	(51.1%)	30.8	(50.4%)
3	125	1.54	(51.3%)	31.0	(50.7%)	1.47	(49.1%)	30.2	(49.3%)
3	250	1.56	(51.9%)	32.1	(52.5%)	1.46	(48.7%)	29.0	(47.5%)
3	375	1.60	(53.5%)	33.3	(54.5%)	1.47	(49.1%)	27.8	(45.4%)
3	500	1.64	(54.6%)	32.9	(53.7%)	1.41	(47.0%)	28.3	(46.2%)
3	625	1.60	(53.4%)	33.9	(55.4%)	1.40	(46.6%)	27.3	(44.6%)
3	750	1.65	(55.2%)	35.0	(57.2%)	1.28	(42.6%)	26.3	(42.9%)
3	875	1.66	(55.4%)	36.1	(59.0%)	1.32	(43.9%)	25.1	(41.0%)
3	1000	1.67	(55.7%)	36.8	(60.0%)	1.09	(36.2%)	24.7	(40.3%)

5.4 QUALITATIVE ANALYSIS OF MULTIPHASEEULERFOAM RESULTS

The results from the Euler-Euler method with VOF coupling in `multiphaseEulerFoam` are analysed in this section. First the junction is investigated, which is simulated for $\dot{Q}_{l,inlet} = \{1, 2, 3\}$ [m^3 h^{-1}] and $\Delta p = \{0, 250\}$ [Pa]. The riser system is anaylsed for $\dot{Q}_{l,inlet} = \{1, 2, 3\}$ [m^3 h^{-1}] and $\Delta p = \{0, 500, 1000\}$ [Pa].

Contrary to the mixture momentum approach of `interFoam`, the Euler-Euler description of `multiphaseEulerFoam` results in a more physical *looking* flow. In figure 5.4.1 a screenshot of the T-section with mesh **T2** is shown. An inlet volumetric flow rate $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa] are used. Just after the junction the water congregates in large churns which, via the wall of the domain, are united with the water at the bottom. This behaviour results in a wavy flow near the outlets, due to conservation of momentum. Due to the large amount of water added in the inlet, the effect of non-equal production is hard to see in the figure.

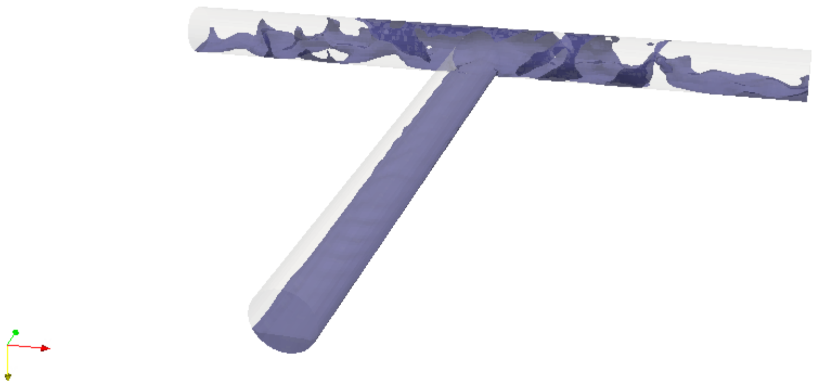


Figure 5.4.1: Snapshot of T-junction with mesh **T2** in `multiphaseEulerFoam` $\dot{Q}_{l,input} = 3$ [m^3 h^{-1}], $\Delta p = 250$ [Pa] and $t = 1$ [s]

When less water is being brought into the domain, the effect of non-equal production at the outlets is clearly visible in an instantaneous snapshot. In figure 5.4.2 the effect of the pressure difference is clearly showing the preferential direction of the water.

5.4.1 T-junction in multiphaseEulerFoam

A clear effect on the production can be seen when a pressure difference over the outlets of the T-junction domain is imposed. In table 5.4.1 the production results for the **T2** are summarised. In line with the observations from `interFoam` the effect on the production of air is larger than on the water.

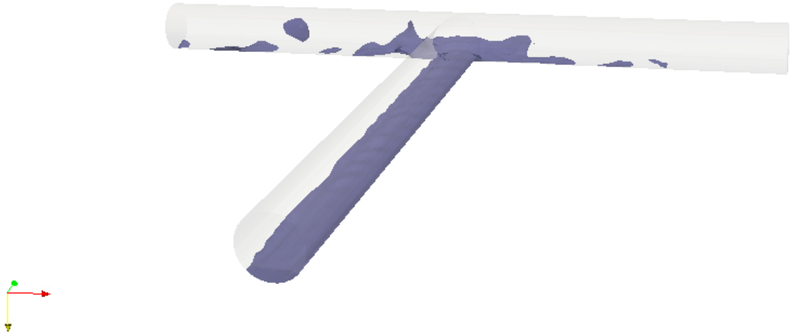


Figure 5.4.2: Snapshot of T-junction with mesh **T2** in `multiphaseEulerFoam` $\dot{Q}_{l,input} = 1$ [m^3 h^{-1}], $\Delta p = 250$ [Pa] and $t = 0.8$ [s]

Table 5.4.1: Production in `multiphaseEulerFoam` with mesh **T2**, $\dot{Q}_{total} = 64.2$ [m^3 h^{-1}].

$\dot{Q}_{l,in}$ [m^3 h^{-1}]	Δp [Pa]	$\dot{Q}_{l,left}$ [m^3 h^{-1}]		$\dot{Q}_{g,left}$ [m^3 h^{-1}]		$\dot{Q}_{l,right}$ [m^3 h^{-1}]		$\dot{Q}_{g,right}$ [m^3 h^{-1}]	
1	0	0.48	(48.4%)	31.6	(50.1%)	0.49	(49.0%)	31.6	(49.9%)
1	250	0.63	(62.7%)	49.6	(78.5%)	0.38	(37.6%)	13.6	(21.5%)
2	0	0.99	(49.5%)	31.1	(50.0%)	0.99	(49.7%)	31.1	(50.0%)
2	250	1.09	(54.6%)	41.1	(66.0%)	0.80	(44.2%)	21.1	(34.0%)
3	0	1.48	(49.4%)	30.9	(50.4%)	1.48	(49.4%)	30.4	(49.6%)
3	250	1.55	(51.8%)	37.6	(61.4%)	1.41	(47.1%)	23.7	(38.7%)

5.4.2 Riser system in multiphaseEulerFoam

The simulation times for the riser system in OpenFOAM's `multiphaseEulerFoam` solver are very long. The simulation times for the riser system are between one and three weeks on 4 cores on the `hpc12` cluster. No strong conclusions are drawn from these results. Longer simulation times are needed in order to investigate the behaviour of the flow simulated with `multiphaseEulerFoam`.

Figure 5.4.3 provides a snapshot of the flow in the riser in `multiphaseEulerFoam`. One can see the increased hold-up in the elbows, before the liquid in transported upwards by the inflowing air. Furthermore, the behaviour of the liquid in the T-junction looks physical in terms of droplet formation In table 5.4.2 the pressure calculations of the large riser are summarised.

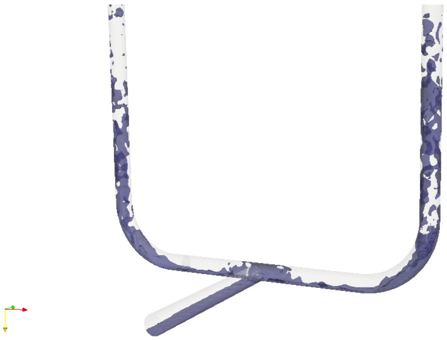


Figure 5.4.3: Screenshot of **R2** mesh in `multiphaseEulerFoam` at $t = 3$ [s] and $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 500$ [Pa] between the outlets.

Table 5.4.2: Pressure in `multiphaseEulerFoam` with mesh **R2** and $t \geq 5$ [s].

$\dot{Q}_{l,in}$ [m^3 h^{-1}]	Δp [Pa]	$\mu_{p_{inlet}}$ ($\sigma_{p_{inlet}}$) [Pa]	t_{end} [s]
1	0	2580 (732)	10.0
1	500	2686 (318)	10.0
1	1000	2916 (539)	10.0
2	0	4357 (568)	9.00
2	500	4682 (736)	8.00

5.5 SUMMARY OF QUALITATIVE ANALYSIS

This chapter deals with the analysis of the meshing procedure and the preliminary analysis of the results of both solvers. Three T-junction meshes are generated with SHM. From coarse to fine these are named **T1**, **T2** and **T3**. These meshes have 23042, 47544 and 95292 cells respectively. A low average non-orthogonality of under 7 is produced. The maximum skewness of the meshes (from coarse to fine) are 2.36, 2.00 and 1.22. The riser system geometry is meshed two times in mesh **R1** (67720 cells) and **R2** (276784 cells). The average non-orthogonality of the coarsest mesh is 10.6442 and its maximum skewness 1.42887. The medium mesh **R2** has an average non-orthogonality of 6.77096 and a maximum skewness of 60.486. Due to the nature of SHM the generated meshes are non-symmetrical, which can result in non-equal production of phases at the outlets.

In terms of mesh dependency one can see that the calculated inlet pressure of the finest mesh **T3** mostly lies between the results from mesh **T1** and **T2**. The liquid production difference between the coarsest mesh and the finest mesh are substantial. The medium mesh performs well and the mean production of this mesh is close to the finest mesh.

The `interFoam` results show some unrealistic flow behaviour right after the junction. Elongated droplets are formed which slide down against the walls. These droplets propagate in a regular fashion. The `multiphaseEulerFoam` solver shows very realistic flow behaviour through the domains. The droplet formation close to and after the junction appears to be physical.

The next chapter will compare the results of both solvers in terms of the calculated inlet pressure, the liquid and gas production at the outlet and the hold-up in the domains. The largest discrepancy will be in the pressure calculations.

Chapter 6

Comparison of results from the two solvers

In this chapter the results of `interFoam` and `multiphaseEulerFoam` are compared. First, the results for the inlet pressure, liquid production and hold-up in the T-junction are analysed. After that the results from the large riser system will be compared.

6.1 T-junction

This section will investigate the differences between the results of the solvers in the T-junction. The simulations are performed on the **T2** mesh (47544 cells). First, the flow patterns are compared. Next, the inlet pressure, the production at the outlets and the hold-up are compared.

To compare the results from both solvers, simulations are performed in the **T2** mesh with $\dot{Q}_{l,inlet} = \{1, 2, 3\}$ [m^3 h^{-1}] and $\Delta p = \{0, 250\}$ [Pa]. Differences in values between the solvers appear most prominently in low liquid hold-up conditions. The difference between the flow patterns in `interFoam` (left) and `multiphaseEulerFoam` (right) is shown in figure 6.1.1. The latter solver gives more droplet formation at the junction. This flow regime results in a higher liquid hold up (see section 6.1.3) and lower inlet pressure (see section 6.1.1).

6.1.1 Difference of inlet pressure in T-junction between solvers

In figures 6.1.2 to 6.1.4 the differences between the inlet pressure p_{inlet} are shown. In blue the 1 [s] moving mean of the inlet pressure from `interFoam` solver is displayed. The values from `multiphaseEulerFoam` are displayed in red. Overall, no large differences in the inlet pressure are found, except for the case with a low volumetric flow rate of water. An investigation for this effect is done and from visual inspection it can be concluded that a more chaotic flow pattern is present when using `multiphaseEulerFoam`. An example of the difference in flow pattern is shown in in figure 6.1.1.

6.1.2 Production of liquid and gas at the outlets

The production results for the air and water production at the outlets indicate that using `multiphaseEulerFoam` results in more oscillations, as is shown in figures 6.1.5 to 6.1.7. When a pressure difference is applied the liquid production at the left outlet is lower for `multiphaseEulerFoam`.

As mentioned earlier, the maldistribution of the gas production is higher than the maldistribution of the liquid production. This is expected, because due to higher density of the water the inertia of the liquid is

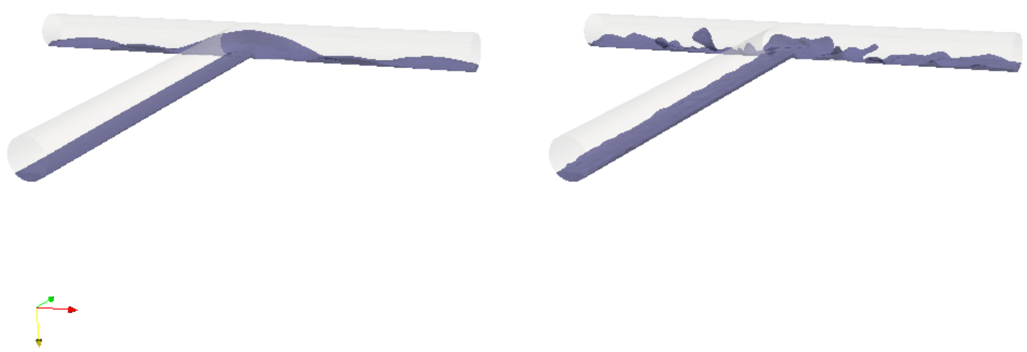


Figure 6.1.1: Comparison of flow pattern on the **T2** mesh with `interFoam` (left) and `multiphaseEulerFoam` (right), $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 0$ [Pa], $t = 1.7$ [s]

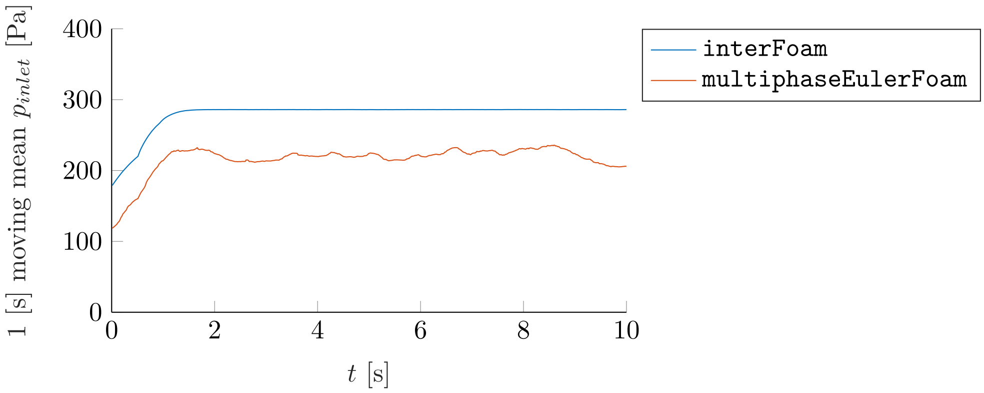


Figure 6.1.2: The inlet pressure for **T2** with $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

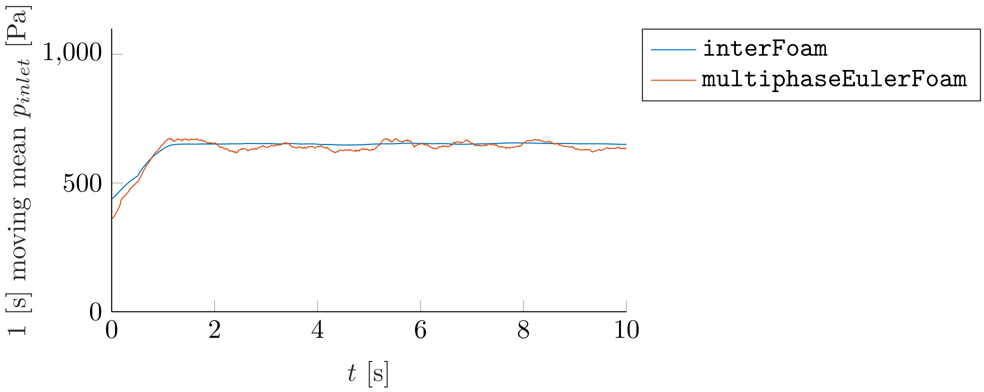


Figure 6.1.3: The inlet pressure for **T2** with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

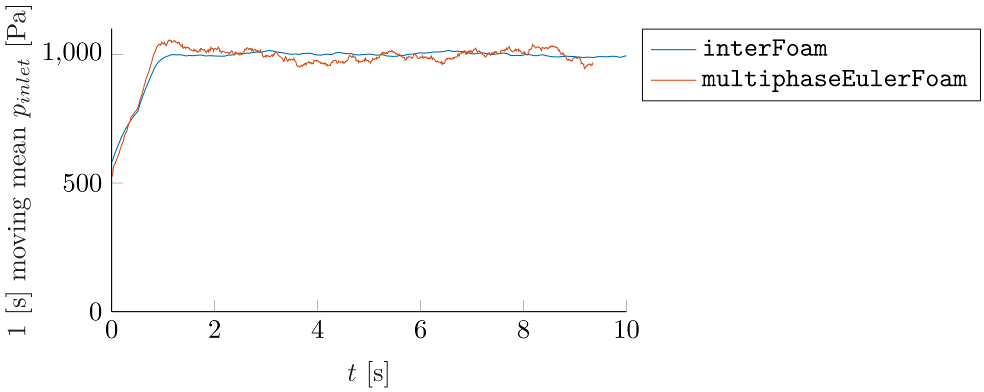


Figure 6.1.4: The inlet pressure for **T2** with $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

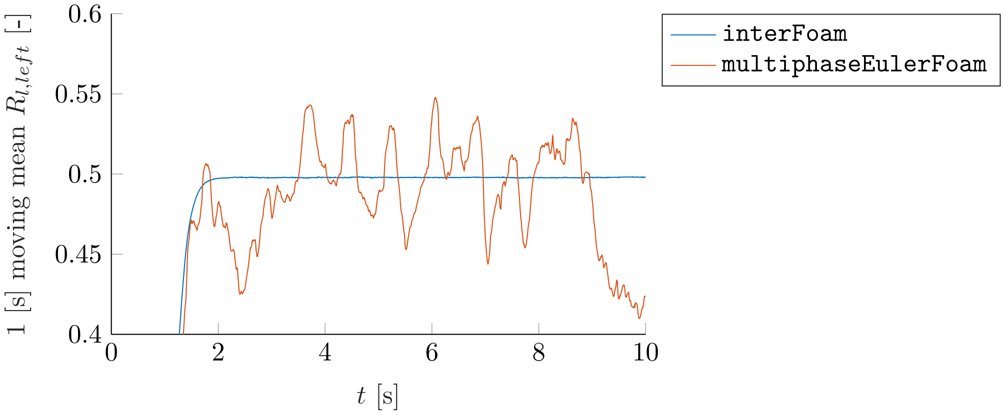


Figure 6.1.5: The normalised liquid production at the left outlet in mesh **T2**. With $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

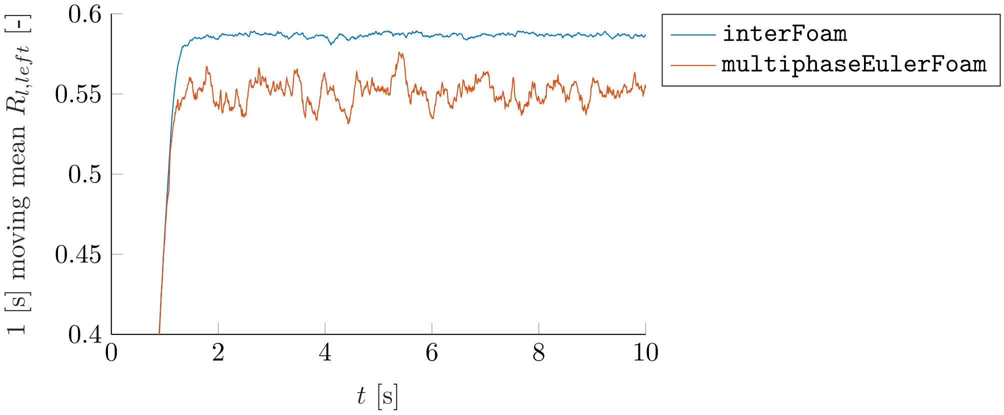


Figure 6.1.6: The normalised liquid production at the left outlet in mesh **T2**. With $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

higher. Contrary to the clear oscillations of liquid production in the `multiphaseEulerFoam` solver, the production of gas is more steady. However, the waves do not fully block the pipe and restrict the air flowing out the domain. In figures 6.1.8 to 6.1.10 a selection of three out of the six cases are shown.

Especially when no pressure difference between the outlets is imposed, the gas production is steady. When a pressure difference is applied, an increase in oscillations occurs. More interesting is the fact that

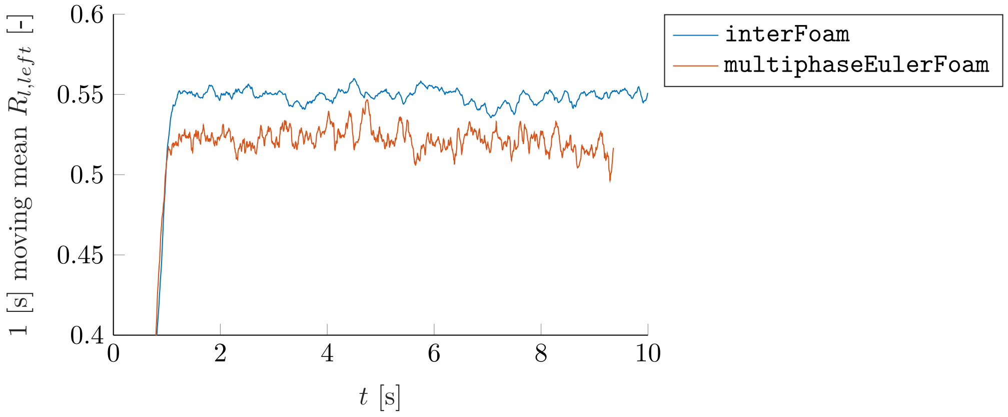


Figure 6.1.7: The normalised liquid production at the left outlet in mesh **T2**. With $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

`interFoam` calculates a lower gas production for $\dot{Q}_{l,inlet} = 2$ and 3 [m^3 h^{-1}] when a pressure difference of $\Delta p = 250$ [Pa] is applied. However, a lower production of the gas from `multiphaseEulerFoam` is obtained from when $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}]. In figure 6.1.8 the normalised gas production at the outlet is shown.

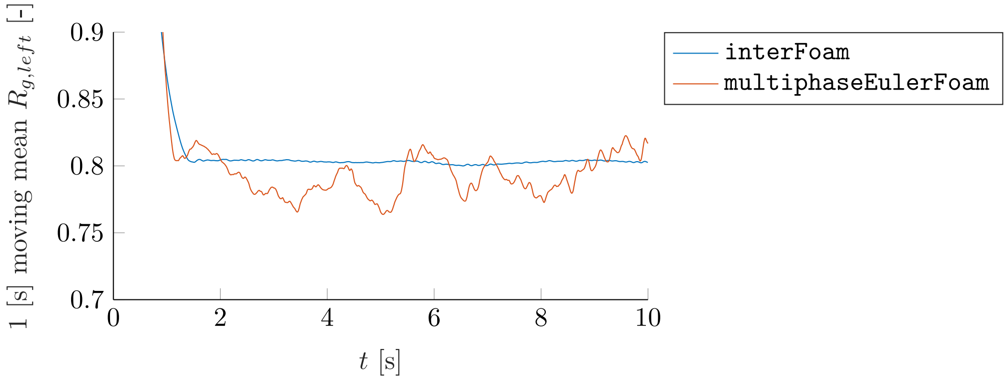


Figure 6.1.8: The production of gas at the left outlet in mesh **T2** for $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

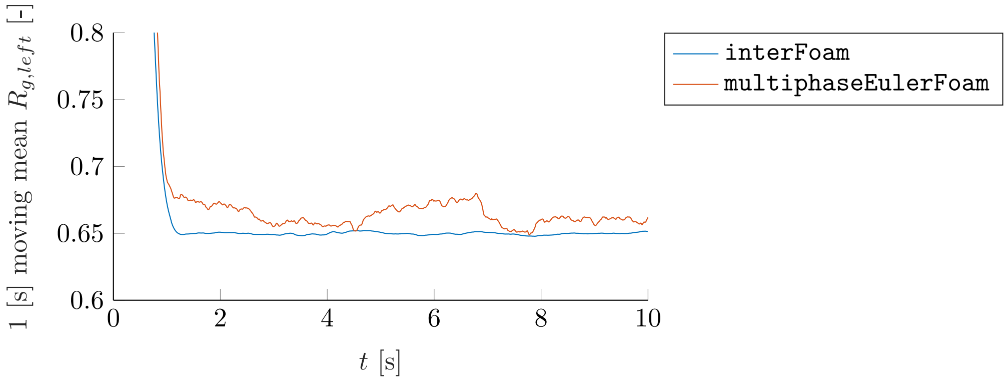


Figure 6.1.9: The production of gas at the left outlet in mesh **T2** for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 250$ [Pa]

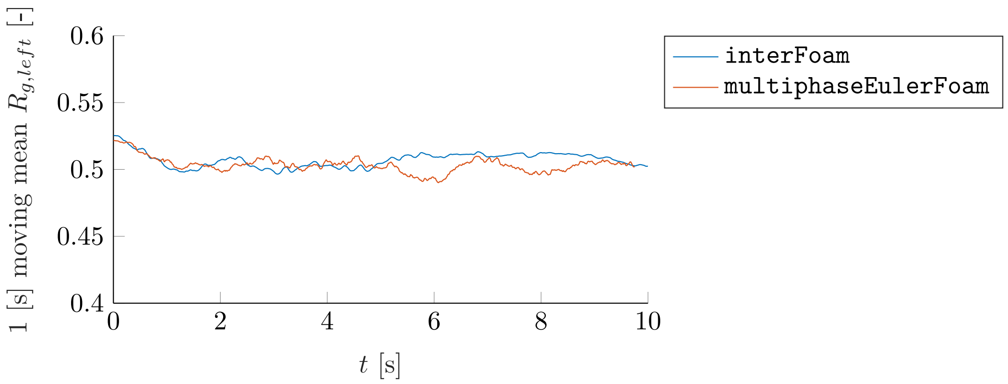


Figure 6.1.10: The production of gas at the left outlet in mesh **T2** for $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

6.1.3 Comparison of hold-up in interFoam and multiphaseEulerFoam

To find the liquid hold-up in the domain over time the following expression is used: (here V_i is the cell volume of cell i):

$$V_{l,domain}(t) = \sum_{i=1}^{\#Cells} \alpha_{l,i}(t) \cdot V_i \tag{6.1.1}$$

Figures 6.1.11 to 6.1.13 show the liquid hold-up in the domain. By using `checkMesh` the total volume of the domain is found to be $V_{domain} = 2.197 \cdot 10^{-3}$ [m^3]. The normalised hold-up is found by using the following definition:

Definition of normalised hold-up of phase k

$$Y_k \equiv \frac{V_{k,domain}}{V_{domain}} \tag{6.1.2}$$

As can be seen from table 6.1.1, the liquid hold-up in the domain is higher in `multiphaseEulerFoam` than in `interFoam` for $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}]. For the other cases the hold-up is lower. When the results are compared to the calculated pressure at the inlet, it can be seen that the hold-up and inlet pressure are inversely related. This is expected from the Fanning friction factor equation [29]:

$$\frac{\Delta p}{L} = \frac{2f\tilde{\rho}\mathcal{U}^2}{D} \tag{6.1.3}$$

Where:

- L Length of pipe
- f Fanning friction factor
- $\mathcal{U}$ Characteristic flow velocity
- $\tilde{\rho}$ Averaged density

For a lower characteristic flow velocity the pressure loss in a horizontal pipe is lower. The liquid in the domain has more inertia and therefore it is harder to accelerate the liquid, which thus gives that the characteristic velocity in the pipe is lower. This results in a lower calculated inlet pressure when the hold-up is higher.

Table 6.1.1: Liquid hold-up in T-junction with mesh **T2** and $2 \leq t \leq t_{end}$.

		interFoam			multiphaseEulerFoam		
$\dot{Q}_{l,inlet}$ [m^3 h^{-1}]	Δp [Pa]	μ_{Y_l}	(σ_{Y_l})	[-]	μ_{Y_l}	(σ_{Y_l})	[-]
1	0	0.12	$(2 \cdot 10^{-4})$		0.13	$(46 \cdot 10^{-4})$	
1	250	0.12	$(4 \cdot 10^{-4})$		0.14	$(35 \cdot 10^{-4})$	
2	0	0.21	$(6 \cdot 10^{-4})$		0.19	$(30 \cdot 10^{-4})$	
2	250	0.21	$(3 \cdot 10^{-4})$		0.19	$(37 \cdot 10^{-4})$	
3	0	0.25	$(24 \cdot 10^{-4})$		0.23	$(44 \cdot 10^{-4})$	
3	250	0.25	$(16 \cdot 10^{-4})$		0.23	$(40 \cdot 10^{-4})$	

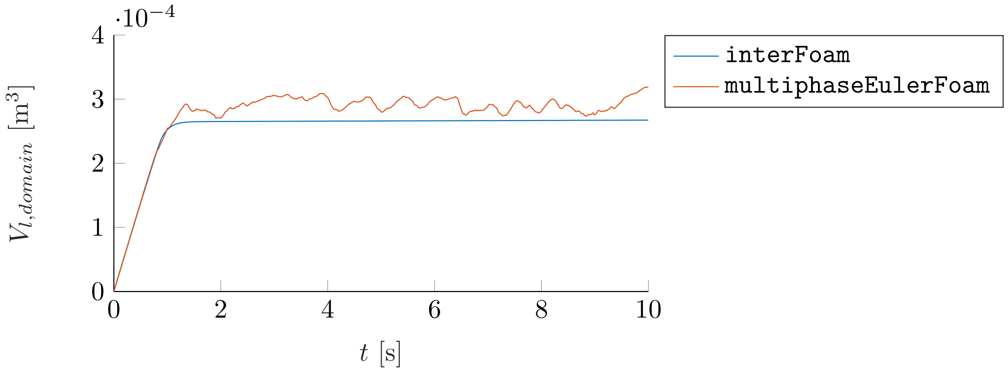


Figure 6.1.11: The liquid hold-up in mesh **T2** for $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

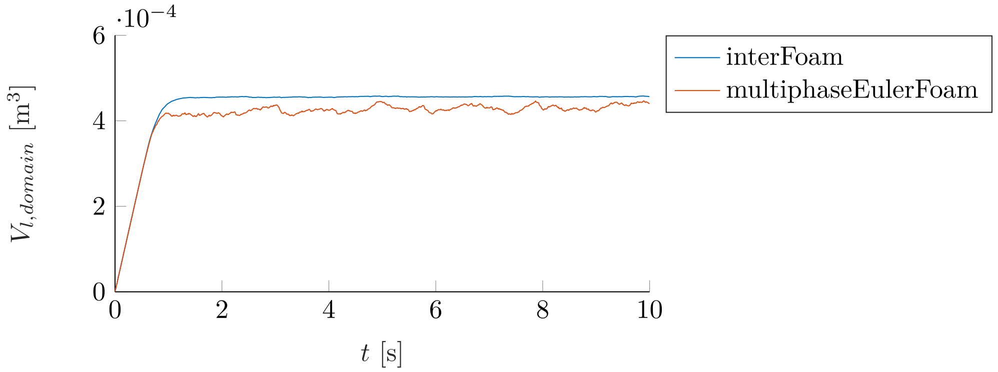


Figure 6.1.12: The liquid hold-up in mesh **T2** for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

6.1.4 Comparison of computational times for both solvers

In the paper of [93] the computational times of `interFoam` and `multiphaseEulerFoam` have been compared. For an annular mixer with 44800 hexahedral cells the computational time of `multiphaseEulerFoam` was 39% higher *per time step* than in `interFoam`. In this thesis the wall clock times of both solvers are compared for the **T2** mesh, once with a fixed time step of $\Delta t = 510^{-6}$ [s] and once with an adjustable time step and fixed CFL condition of $\text{Co}_{max} = 0.25$.

First, the fixed time step simulations are compared. By setting $\Delta = 5 \cdot 10^{-6}$ [s] both solvers take an equal amount of time steps, such that a comparison with the findings of Wardle and Weller can be made. One

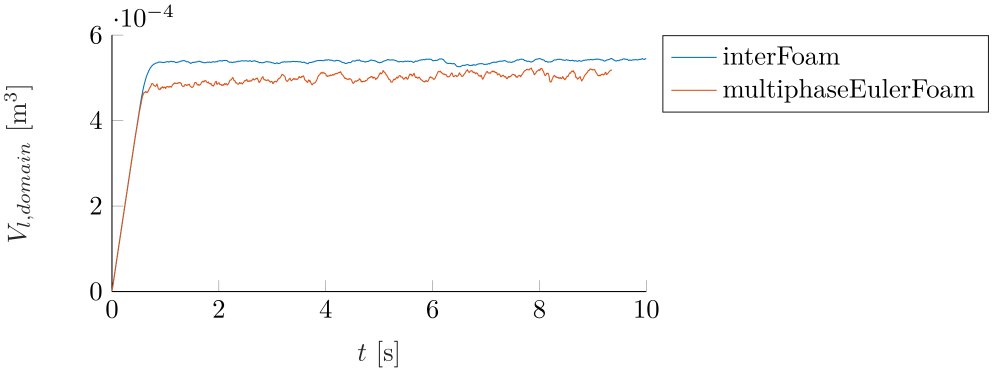


Figure 6.1.13: The liquid hold-up in mesh **T2** for $\dot{Q}_{l,inlet} = 3$ [m^3 h^{-1}], $\Delta p = 0$ [Pa]

processor core of `typei`[1] is used. The time per time step is calculated as follows:

$$t_{step} = \frac{\text{Wall clock time [s]}}{\text{Number of time steps taken [-]}} \tag{6.1.4}$$

In the simulations with the constant time steps, the following values are found at$t = 0.8$ [s]:

$$t_{step,IF} = \frac{53632.3}{160000} = 0.335 \text{ [s]} \tag{6.1.5}$$

$$t_{step,MPEF} = \frac{135211}{160000} = 0.845 \text{ [s]} \tag{6.1.6}$$

The results show that `multiphaseEulerFoam` ($t_{step,MPEF}$) needs 152% more wall clock time *per time step* than `interFoam` ($t_{step,IF}$) This result is much larger than the findings of Wardle and Weller.

Two other simulations on the `hpc12` cluster at Delft University of Technology are performed with the same boundary conditions and system settings. The initial time-step is taken as $\Delta t_{init} = 10^{-5}$ to force equal starting conditions. However, the solvers are allowed to dynamically change the time step while complying with the CFL condition of Co$_{max} = 0.25$. The comparison between run times in this thesis is done for a given amount of simulation time. That means that this run time comparison uses the following definition:

$$\eta_t = \frac{\text{Wall clock time [s]}}{\text{Simulation time [s]}} \tag{6.1.7}$$

Here η_t is a non-dimensional parameter which describes the ratio between the execution time and the time in the simulation. Here the values are stated for $t = 8.00$ [s].

$$\eta_{t,IF} = \frac{165567}{8.00} = 20.7 \cdot 10^3 \text{ [-]} \tag{6.1.8}$$

$$\eta_{t,MPEF} = \frac{371563}{8.00} = 46.4 \cdot 10^3 \text{ [-]} \tag{6.1.9}$$

As can be seen from the time ratios of `interFoam` ($\eta_{t,IF}$) and `multiphaseEulerFoam` ($\eta_{t,MPEF}$), the latter needs 124% more wall clock time per unit simulation time than `interFoam`. Higher local velocities are present in the simulation with `multiphaseEulerFoam`. The oscillations present in the results of this solver influence the CFL condition in such a way that a smaller time step is necessary to comply with the upper bound of Co$_{max} = 0.25$.

For the simulation with adjustable time steps the total number of time steps can be found by using the `grep -c` command. In the simulations with adjustable time step the following values for wall clock time

[1] A `typei` core is an Intel Xeon E5-2660v3 CPU

per time step are found:

$$t^*_{step,IF} = \frac{210822}{609697} = 0.346 \text{ [s]} \tag{6.1.10}$$

$$t^*_{step,MPEF} = \frac{416300}{559257} = 0.744 \text{ [s]} \tag{6.1.11}$$

Here the values are taken at different times. The values from `interFoam` are obtained at $t = t_{end} = 10$ [s]. The values from `multiphaseEulerFoam` are taken at $t = 8.97$ [s]. From this analysis can be concluded that `multiphaseEulerFoam` needs 115% more wall clock time *per time step* than `interFoam`. This result differs largely compared to the findings of Wardle and Weller.

Earlier, this report concluded on similarities between solutions in both solvers. When no pressure difference between the outlets is applied and enough water is brought into the domain, results of both solvers are similar. Besides the easier set-up of `interFoam`, the much smaller wall clock time is a strong incentive to choose the `interFoam` solver.

6.1.5 Conclusion on the T-junction

As can be seen from the pressure graphs and the graphs of the production of the liquid and gas at the outlets, results from `multiphaseEulerFoam` shows more oscillating behaviour for those variables. As can be seen from the results shown in figure 6.1.1, the solver shows droplet formation with **T2** mesh, which leads to the aforementioned variations in production.

An interesting inverse relationship between the hold-up and the pressure can be seen. When the liquid hold-up in the domain is higher, the calculated inlet pressure is lower. The Fanning pressure loss equation shows that a lower characteristic velocity leads to a lower pressure loss in a horizontal pipe. More liquid in the system results in a lower phase averaged velocity (due to the inertia of the water), which leads to the lower pressure losses.

6.2 Riser system compared for both solvers

In figure 6.2.1 and figure 6.2.2 snapshots of the large riser system at different time steps are displayed. Qualitatively, the flow of the water appears to be similar. The quantitative investigation on the calculated inlet pressure and production at the outlets will provide the differences between the solutions from `interFoam` and `multiphaseEulerFoam`.

An important difficulty in simulating the large riser system is the large time before the outlets start a steady production of water. After starting the simulation at $t = 0$ [s] it takes a long time before the outlets start to produce liquid in a steady rate. Due to the high computational costs of simulating such a system, drawing conclusions on the production should be done with care. In this thesis production values are sampled from $t = 5$ [s] to $t = t_{end} = 10$ [s]. Additional studies on this large riser system can use the case files added in the appendices of this report in order to obtain a good start in simulating the system. At the `hpc12` cluster at Delft University of Technology simulating the large system on four cores took approximately fourteen days for a simulation of 10 [s]. A longer simulation time is advised, to draw a good conclusion on the non-symmetrical split in production of water and air at the outlets.

Nevertheless, this section will continue with the analysis of the riser system in both solvers. Again, a quantitative analysis is given with instantaneous screenshots of the system. Then the results of the calculations are summarised, in terms of inlet pressure and production.

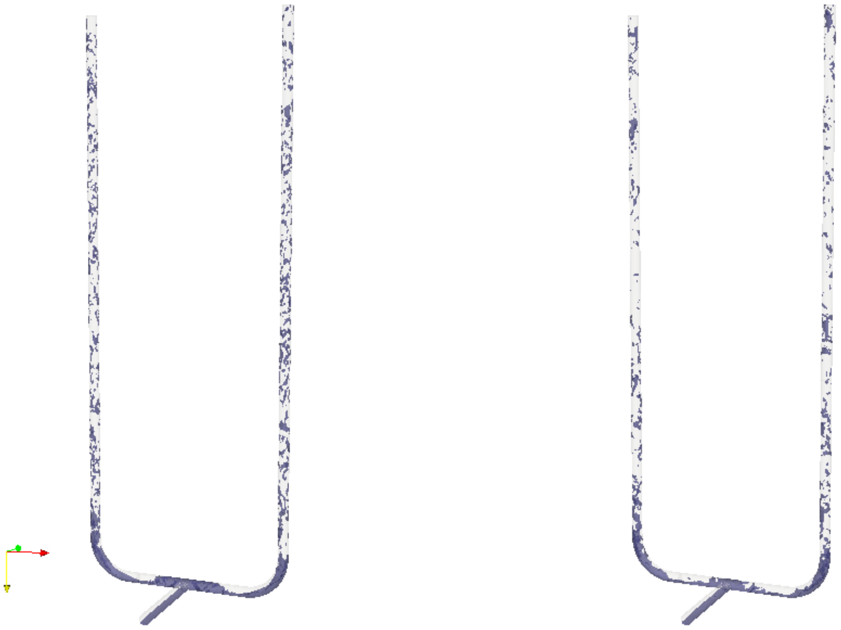


Figure 6.2.1: Snapshot of riser system with mesh **R2** in `interFoam` (left) and `multiphaseEulerFoam` (right), $\dot{Q}_{l,input} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa] and $t = 5$ [s]

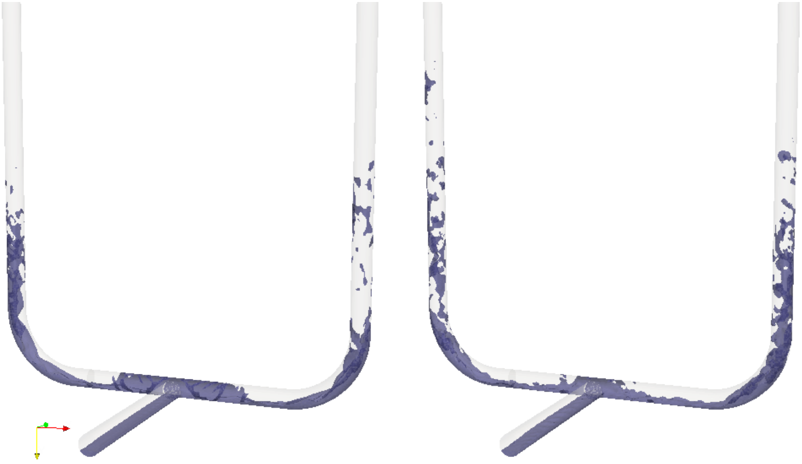


Figure 6.2.2: Snapshot of riser system with mesh **R2** in `interFoam` (left) and `multiphaseEulerFoam` (right), $\dot{Q}_{l,input} = 2$ [m^3 h^{-1}], $\Delta p = 0$ [Pa] and $t = 2$ [s]

6.2.1 Pressure calculations in large riser

Using the methodology of the previous chapter, the pressure at the inlet of the riser is calculated. Immediately, the higher inlet pressures can be identified. This is due to the additional hydrostatic pressure in the system. Theoretically, the pressure difference between the inlet and outlet of a vertical pipe (y pointing in opposite direction of gravity) is given by:

$$\Delta p = \Delta p_{friction} + \Delta p_{hydro} = \Delta p_{friction} - \rho g y \tag{6.2.1}$$

When an average density and a liquid hold-up of 0.2 are assumed:

$$\tilde{\rho} = \alpha_l \rho_l + (1 - \alpha_l) + \rho_g = 0.2 \cdot 999 + (1 - 0.2) \cdot 1.25 \approx 200 \text{ [kg m}^{-3}\text{]} \tag{6.2.2}$$

The additional pressure loss only due to the hydrostatic pressure in de 2.5 [m] riser is:

$$\Delta p_{hydro} = -\tilde{\rho} g y = -200 \cdot 9.81 \cdot (-2.5) = 4.9 \cdot 10^3 \text{ [Pa]} \tag{6.2.3}$$

The actual liquid hold-up has to be calculated to draw conclusions for the specific cases. As can be seen in figure 6.2.3 and figure 6.2.4 the pressures in `multiphaseEulerFoam` are varying, while by using `interFoam` the resulting inlet pressures are more stable. It seems that the liquid in the mixed momentum formulation is transported upwards more easily. Slugging in the risers occurs in cases where the liquid volumetric flow rate at the inlet is lower. This slugging leads to a higher inlet pressure due to the fixed inlet velocities of both fluids at the inlet as the gas and liquid flow rates at the inlet are fixed.

In figure 6.2.3 and figure 6.2.5 is shown that by using the `multiphaseEulerFoam` solver the resulting inlet pressure is lower than by using the `interFoam` solver, while both solvers take approximately the same time to reach a steady state for the the pressure. Another interesting observation is the absence of oscillating in the pressure. In the T-junction the `multiphaseEulerFoam` calculated a more oscillating pressure and production profile. Using `interFoam` resulted in more steady results. Therefore it can be concluded that the vertical risers have a dampening effect on the resulting calculated inlet pressure in `multiphaseEulerFoam`.

Earlier a study on the large riser is done by Worthen [95]. A VOF mixture momentum model with RANS was used in ANSYS FLUENT to simulate the large riser with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets. In figure 6.2.4 the results of this study for the inlet pressure are shown in yellow. Both OpenFOAM solvers calculate higher pressure losses in the domain.

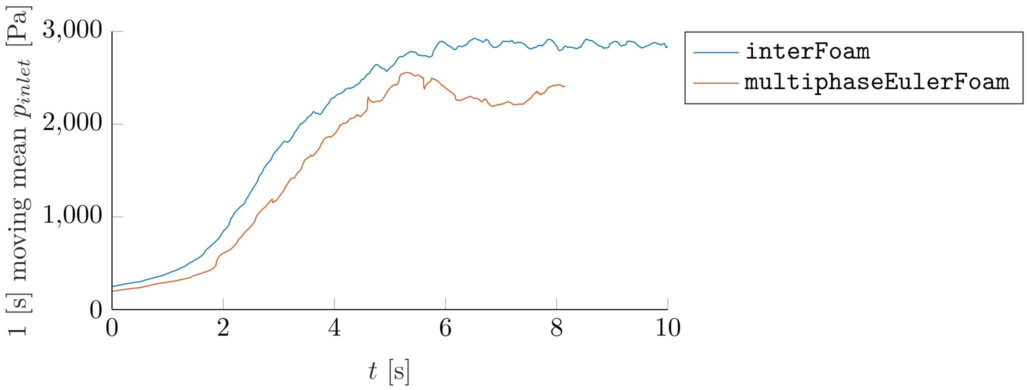


Figure 6.2.3: Comparison of the inlet pressure p_{inlet} for $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa].

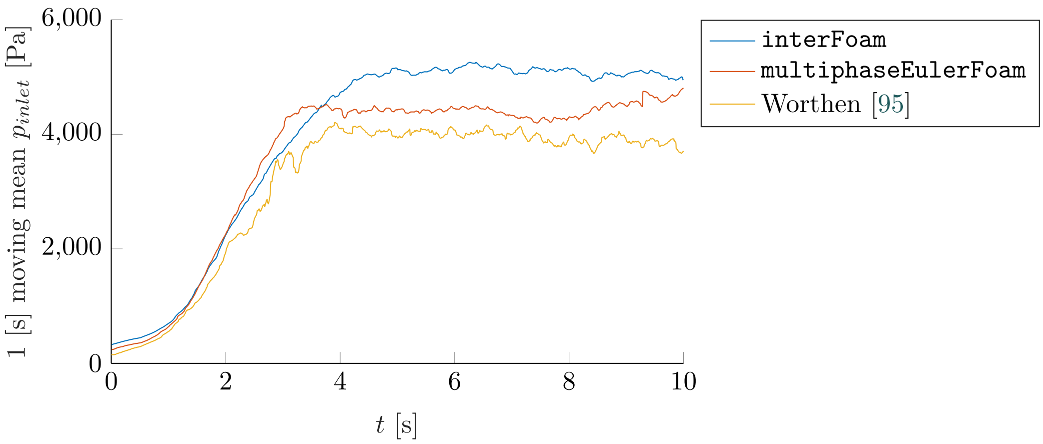


Figure 6.2.4: Comparison of the inlet pressure p_{inlet} for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa].

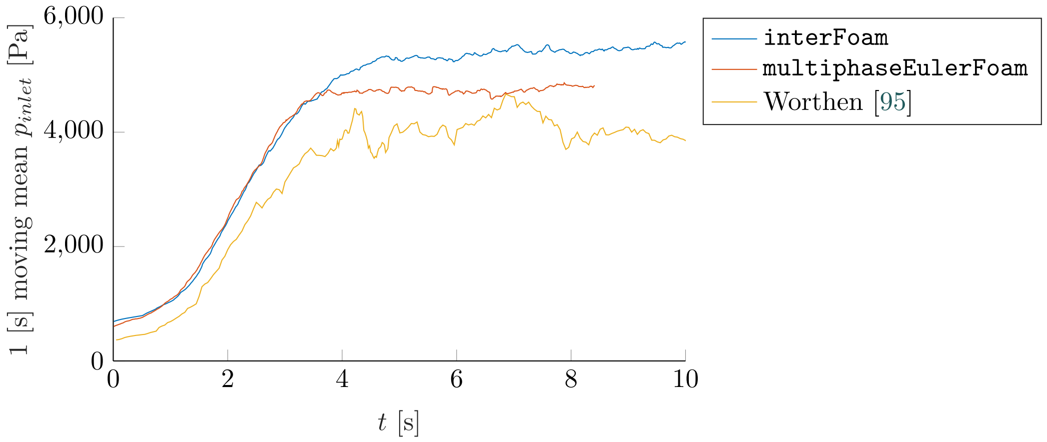


Figure 6.2.5: Comparison of the inlet pressure p_{inlet} for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 500$ [Pa].

6.2.2 Production of liquid in large riser

In figure 6.2.6 and figure 6.2.7 comparisons between the liquid production **interFoam** and **multiphaseEulerFoam** are shown. Both solvers show similar results in the magnitude of the oscillations. When $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] is brought into the domain, the left riser starts its production earlier in **multiphaseEulerFoam** than in **interFoam**. When a pressure difference is applied, using **multiphaseEulerFoam** results in slightly larger oscillations and a higher production.

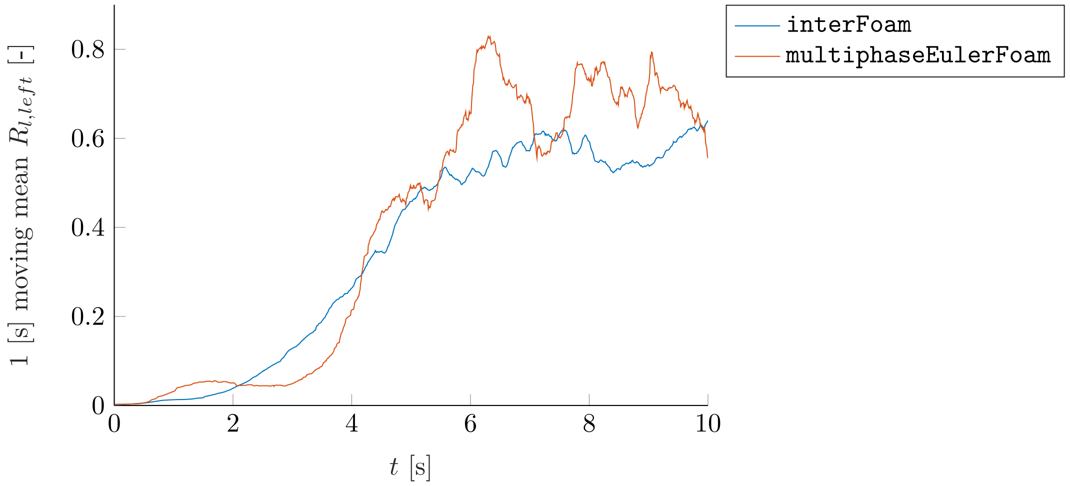


Figure 6.2.6: Comparison of the normalised liquid production for $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}] and $\Delta p = 500$ [Pa].

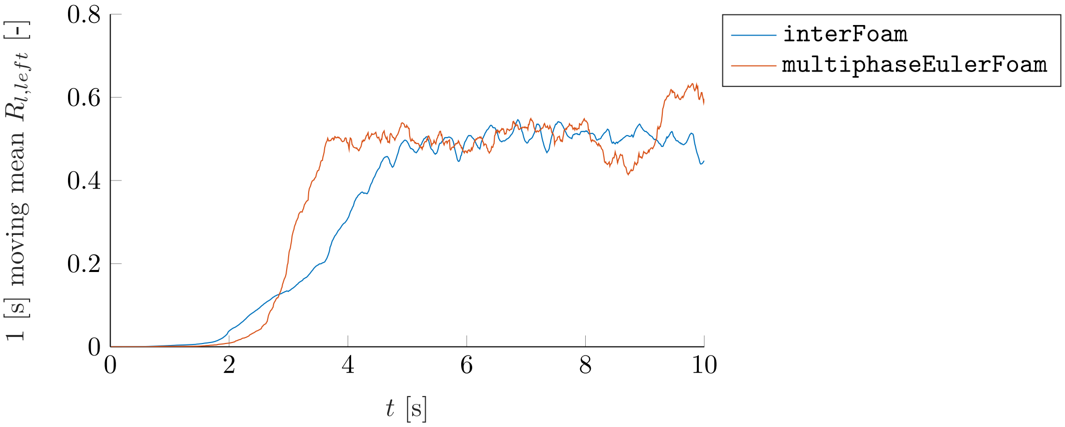


Figure 6.2.7: Comparison of the normalised liquid production for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa].

6.3 Liquid hold-up in riser and pressure drop

The total liquid hold-up in the riser system is shown in figures 6.3.1 to 6.3.2. Interestingly, by using `interFoam` a higher liquid hold-up in the entire domain is obtained for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}]. For $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}] a similar hold-up profile is obtained for both solvers, with a more oscillating behaviour in `multiphaseEulerFoam`.

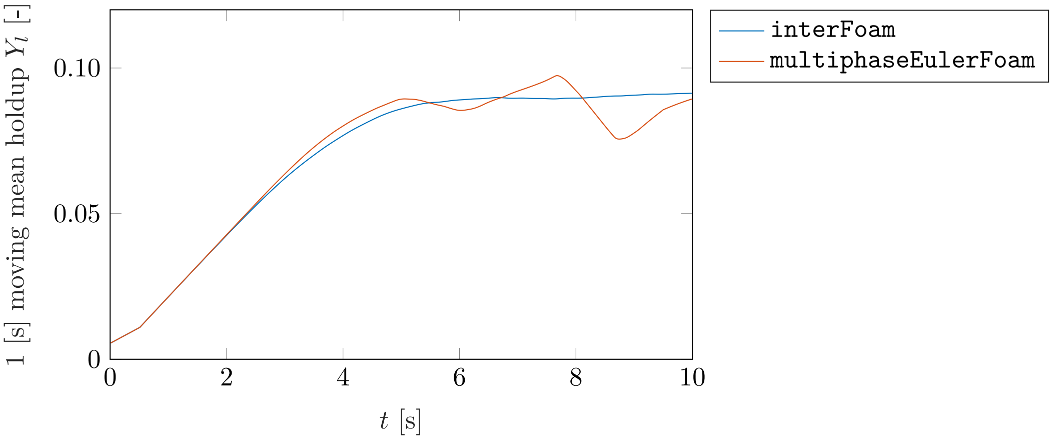


Figure 6.3.1: Comparison of the hold-up in total domain for $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa].

The liquid hold-up is also analysed from $t = 0$ [s] to $t = 10$ [s] for the left riser from $y = -1.03$ [m] to $y_{outlet} = -2.68$ [m]. This results in a sampling interval of $L = 1.65$ [m]. The interval is shown in figure 6.3.3. The pressure drop in the left riser is analysed by taking the same sampling interval.

6.3.1 Liquid hold-up

The results for the liquid hold-up in the riser section are shown in figure 6.3.4. By using `interFoam` a clearly higher liquid hold-up is obtained. The dual riser set-up is experimentally investigated earlier. The liquid hold-up in a single vertical riser has been estimated as 0.24 [96]. The initial conditions used in that research differ slightly from the simulations. The diameter of the riser used is $d = 0.0508$ [m], while in this research a diameter of $d = 0.05$ [m] is used. In the experiments the inlet conditions are equal to this

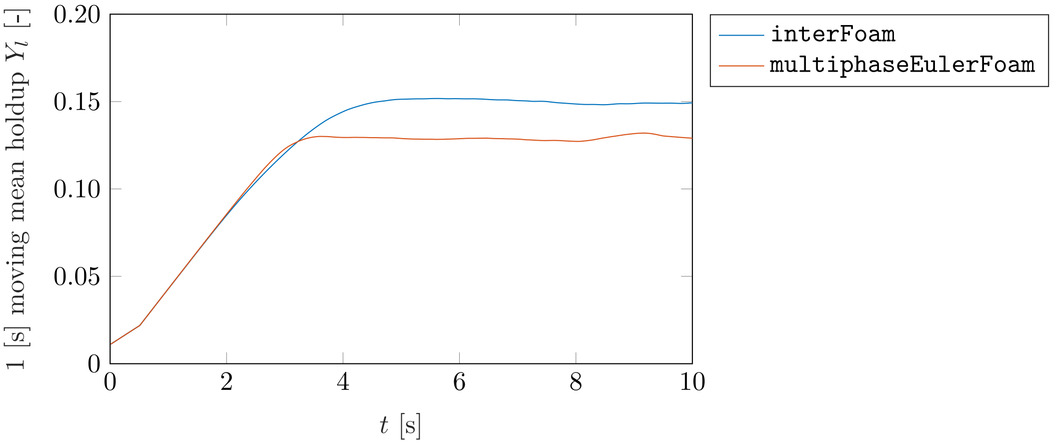


Figure 6.3.2: Comparison of the hold-up in total domain for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa].

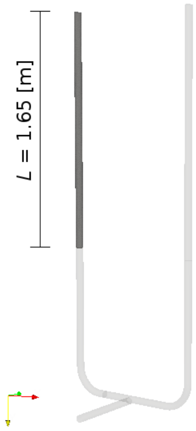


Figure 6.3.3: Sample interval of the left riser with a length of $L = 1.65$ [m].

research: $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\dot{Q}_{g,inlet} = 62.2$ [m^3 h^{-1}]. However, as can be seen from figure 6.3.4, both solvers fail to accurately predict the liquid hold-up from experiments. The liquid hold-up in the `multiphaseEulerFoam` solver varies between 0.06 and 0.9 and the results from `interFoam` are around hold-ups of 0.10 and 0.11.

A finer mesh would increase the capturing of the interfaces better, which results in a better flow regime prediction. As can be seen in the snapshots added to this research, the pockets of water tend to break up in the riser, while in experiments the churn flow regime reduces the velocity of the water. This higher hold-up results in a higher pressure drop in the riser, as is stated in equation 6.2.1.

6.3.2 Pressure drop in riser

The pressure drop in the riser is directly related to the liquid hold-up, as explained earlier. As is shown in figure 6.3.5 and figure 6.3.6, the pressure drop is the highest by using `interFoam`: around $\Delta p_{riser} = 1.4{\cdot}10^3$ [Pa m^{-1}]. While the results from Worthen show a similar hold-up as the `multiphaseEulerFoam` solver, the pressure drop is predicted lower than the pressure drop of `multiphaseEulerFoam`.

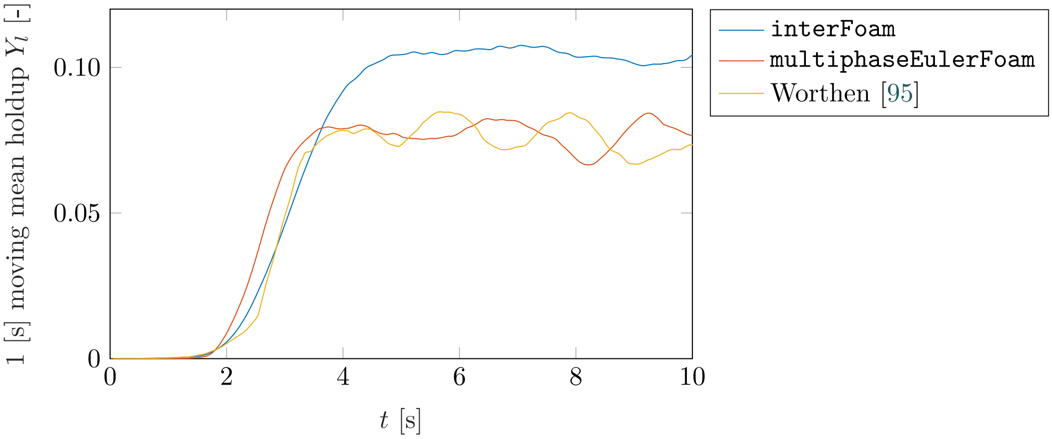


Figure 6.3.4: Liquid hold-up in the riser section with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets.

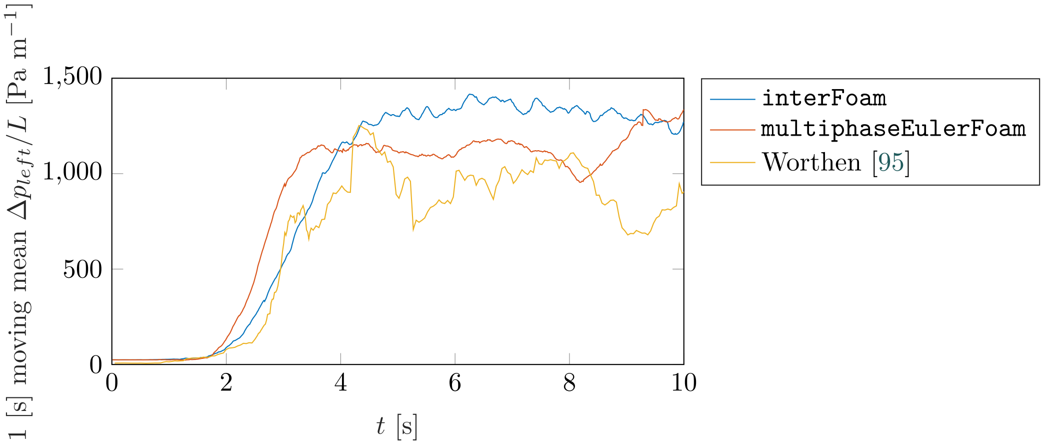


Figure 6.3.5: Pressure drop in the riser section with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets.

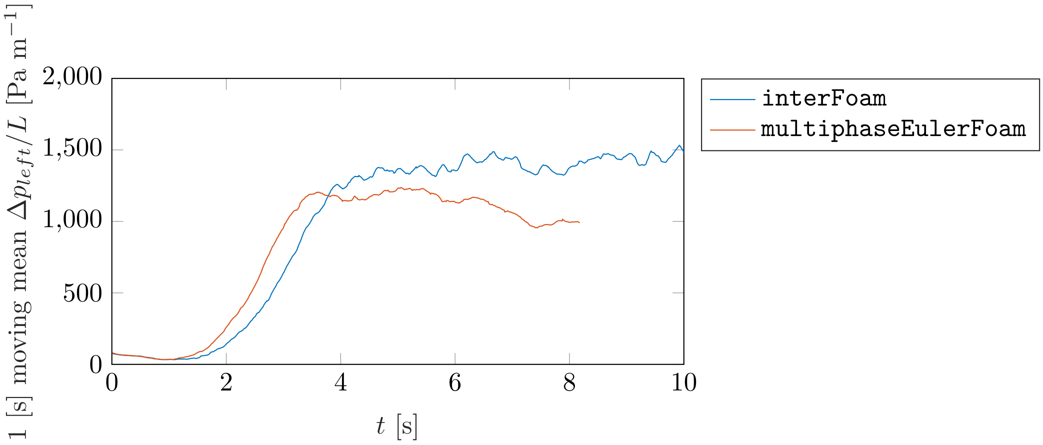


Figure 6.3.6: Pressure drop in the riser section with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 500$ [Pa] between the outlets.

6.4 CONCLUSION

In this chapter the differences between the solvers are analysed. For the T-junction the differences in pressure and production are most noticeable when the liquid hold-up is low. Furthermore, applying a pressure difference yields differences in the liquid production between solvers. The results of the inlet pressure in the riser system differ. Results from the `interFoam` solver show a slightly higher inlet pressure than the results from `multiphaseEulerFoam`. The liquid hold-up is higher in `interFoam` compared to `multiphaseEulerFoam` and ANSYS FLUENT. This results in a higher pressure loss in `interFoam` than in the other solvers.

Chapter 7

Sensitivity analysis

In this chapter additional research opportunities are discussed. To promote further research with `multiphaseEulerFoam` for pipe flow, the case files are added to the appendix, see sections B.2 to B.3. As mentioned in chapter 3, one of the disadvantages of using OpenFOAM is the lack of a proper manual which can make it difficult to start working with OpenFOAM. Therefore we have tried to provide a good description of the cases that were simulated so far. This can help to proceed with the work in the near future. This chapter provides recommendations for new work.

7.1 LONGER SIMULATION TIMES

The previous chapter already briefly touched upon the fact that the simulation end time t_{end} is rather small. At the `hpc12` cluster at Delft University of Technology simulating the large system on four cores took approximately fourteen days for a simulation of $t = 10$ [s]. To increase the confidence in whether a stationary end state has been reached, longer sample times are preferred. Faster computers [53] or more wall clock time is needed to draw additional firm conclusions on the possible maldistribution in the riser system.

It took the author time to correctly set-up the riser system, in terms of a correct mesh and applicable boundary conditions. Due to the sheer number of control variables that SHM offers, meshing of the riser system proved difficult. Additionally, due to the lack of a clear manual, setting up the `multiphaseEulerFoam` simulation was hard. By combining searches in the CFD forums, the tutorial cases and the paper of Wardle and Weller [93] the case was set-up as can be found in the appendices. These case files can be used as a reference towards a proper, working case set-up in `multiphaseEulerFoam`. The author encourages further research on applying the solver for pipe flow problems.

7.2 ENTRANCE LENGTH

To properly simulate a given flow domain, the inlet and outlet conditions must be known. In this study the proper flow regime is not modelled. Worthen [95] estimate the flow pattern to be either stratified wavy or slug flow. Research is done on the correct entrance length for the flow to fully develop. Estimates for the entrance length range between $50D$ and $70D$ [51] and even up to $150D$ [47].

A simulation in OpenFOAM is set-up, with the `multiphaseEulerFoam` solver and a pipe of $z_{pipe} = 150D = 7.5$ [m]. A high-quality mesh consisting of 147200 cells is set up with an average non-orthogonality of 6.5 [deg], a maximum non-orthogonality of 35.0 [deg] and a maximum skewness of 0.48. Again the PISO algorithm is used with the same boundary conditions as the T-junction and the riser system.

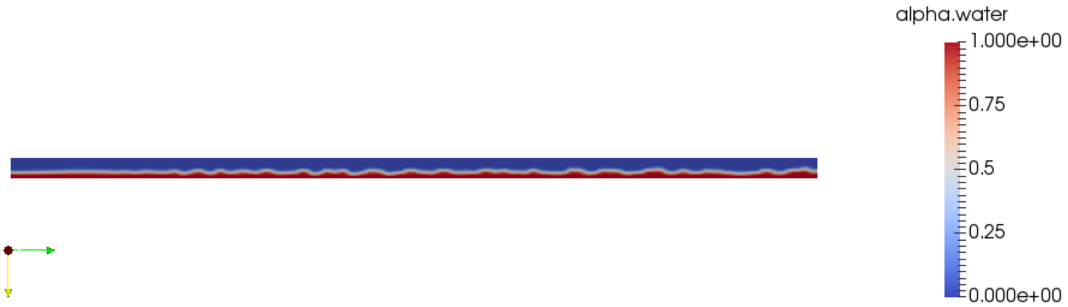


Figure 7.2.1: Mid-line ($x = 0$) liquid phase fraction of long ($z_{pipe} = 7.5$ [m]) pipe simulated with `multiphaseEulerFoam`. Here displayed: $z = 0$ [m] to $z = 2$ [m]. Snapshot taken at $t = 7.5$ [s].

The simulation uses $t_{end} = 10$ [s]. When the flow pattern is qualitatively investigated one can conclude that stratified wavy flow will develop in the pipe. The first 2 [m] of the pipe can be seen in figure 7.2.1. As already found in previous chapter, the wavy pattern appears at around $z = 0.5$ [m]. After the initial onset of wavy motion, the pattern develops into a distinct wavy pattern.

The mesh used in this preliminary simulation is rather coarse. A recommendation thus is to further study the effect on entrance conditions in multiphase pipe flow. Coupled with a right post-processing tool, one could set up an OpenFOAM model in which several initial hold-ups are simulated to see how the flow pattern develops. These results can be compared to existing one-dimensional models, such as Schlumberger's OLGA or The Shell Flow Explorer tool.

7.3 TURBULENCE: CHOICE OF SGS MODEL

Throughout this paper LES is used to simulate turbulence. The used SGS is the Smagorinsky SGS model. The downsides of using this model are mentioned earlier. The main shortcoming of using this model is the poor performance near walls, due to the fact that the size of the eddies is not reduced in cells in that region.

The `multiphaseEulerFoam` solver does solely supports LES as turbulence model, therefore changes in the used turbulence modelling can only be made by changing the code or changing the SGS. In this section alternative choices for the SGS are listed. Further research opportunities exists on the investigating of these models.

When the `LESmodel` keyword is omitted in the `constant/turbulenceProperties` file, OpenFOAM will return an error upon the start of the `multiphaseEulerFoam` program. When an invalid choice is made for the SGS, the program will return a list of valid turbulence models. In OpenFOAM 3.0.1 this list contains the following entries:

- DeardorffDiffStress
- Smagorinsky
- SpalartAllmarasDDES
- SpalartAllmarasDES
- SpalartAllmarasIDDES
- WALE
- dynamicKEqn
- dynamicLagrangian

- kEqn

All these models are tested with `multiphaseEulerFoam`. Only the Smagorinksy and WALE model proved to work without further additions to the code of the solver. WALE has the following advantages over the Smagorinksy SGS [56]:

Turbulence structures: All turbulent structures (i.e. the strain and rotational rates) for turbulent kinetic energy dissipation are used in the model

Eddy viscosity: Due to the wall-adapting capabilities of the model, the eddy viscosity goes to zero near the wall

Shear: In pure shear no eddy viscosity is produced.

Especially in complex geometries and pipe flows the WALE model can aid in a better prediction of turbulence and wall shear stresses.

Two T-junction simulations with WALE are set-up to check for discrepancies in the calculated pressure losses between Smagorinsky and WALE. Unfortunately WALE proved to be not very robust and the simulations in meshes **T2** and **T3** both crashed at around $t \approx 0.5$ [s]. It seems that WALE is sensitive to non-orthogonality. Additional research opportunities are in the area of simulations with other models, including WALE on various meshes.

7.4 THE EFFECT OF INTERFACE COMPRESSION

Interface compression is the addition of a non-physical velocity near the interface between two fluids, in order to reduce the diffusivity at the interface. As described in chapter 3, the advection relation for the phase fraction of fluid k has been derived by Weller [94]:

$$\frac{\mathrm{D}\alpha_k}{\mathrm{D}t} + \nabla \cdot \left(\boldsymbol{u}_c \alpha_k \left(1 - \alpha_k\right)\right) = 0 \tag{7.4.1}$$

Where the compression velocity is given as:

$$\boldsymbol{u}_c = \min\left(C_\alpha |\boldsymbol{u}|, \max\left(|\boldsymbol{u}|\right)\right) \frac{\nabla\alpha}{|\nabla\alpha|} \tag{7.4.2}$$

In `interFoam` the interface compression coefficient is governed by the `cAlpha` keyword in the `alpha.*` dictionary entry in the `system/fvSolution` file. In `multiphaseEulerFoam` this coefficient is provided in the file `constant/transportProperties`.

In this report, the interface compression is made active. By setting using $C_\alpha = 1$ in the solver. Additional research opportunities exist in investigating the influence of the interface compression coefficient by turning it off ($C_\alpha = 0$) or even by adding additional sharpening with $C_\alpha > 1$.

7.4.1 Pressure drop

The influence of turning the interface compression off is investigated for the **T2** mesh by using pressure differences between the outlet of $\Delta p = \{0, 250\}$ [Pa]. The results are shown in figure 7.4.1 and figure 7.4.2. A clear discrepancy can be seen in terms of the inlet pressure for both cases. Interestingly enough, turning interface compression off results in a higher pressure loss.

Further investigation with ParaView and with the post-processing tool `foamCalc` has been done. Figure 7.4.3 shows the pressure at the walls of the domain. It can be clearly seen that the difference in pressure loss is largest in the inlet part of the domain, before the T-junction.

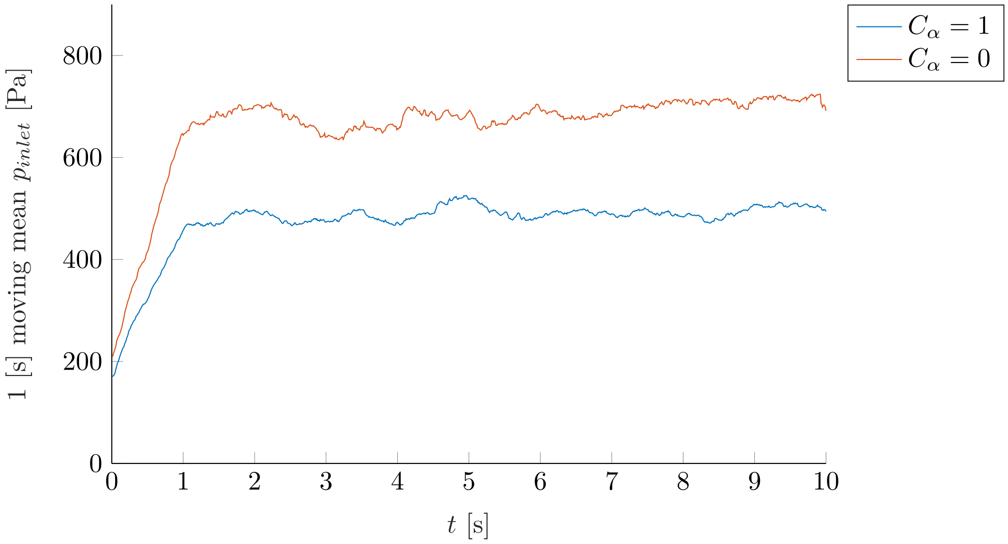


Figure 7.4.1: The effect of the interface compression on the inlet pressure over time with the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets.

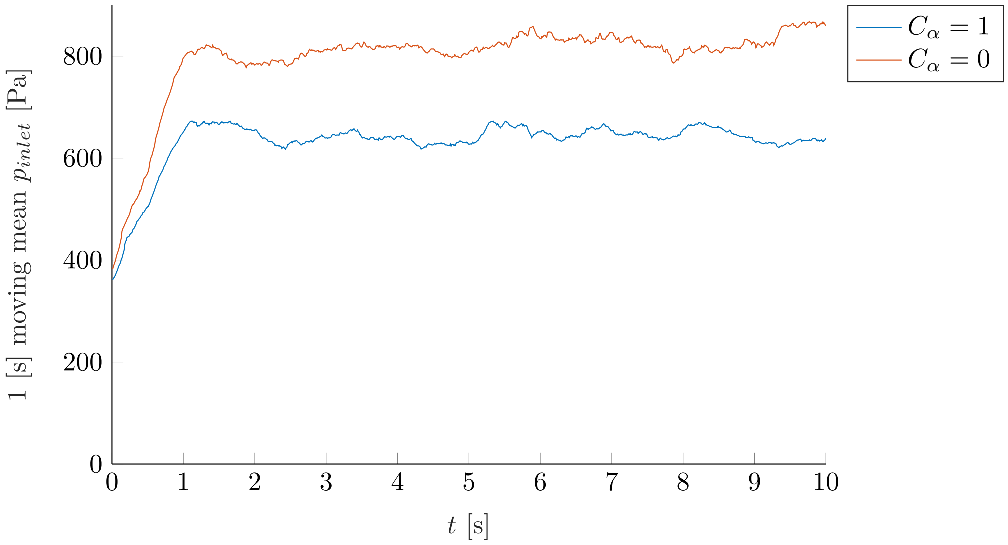


Figure 7.4.2: The effect of the interface compression on the inlet pressure over time with the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa] between the outlets.

The magnitude of the velocity gradient at the wall is investigated, since the wall shear stress is proportional to the derivative of the velocity at the wall:

$$\tau_w = \rho\left(\nu + \nu_t\right)\nabla_{\boldsymbol{n}}\boldsymbol{u} \tag{7.4.3}$$

Where:

- ν_t Turbulent eddy viscosity: field `nut` in OpenFOAM
- $\nabla_{\boldsymbol{n}}\boldsymbol{u}$ Derivative of $\boldsymbol{u}$ normal to wall

One can see that if ν is taken equal for both cases, the wall shear stress is always larger when ν_t *and* $\nabla_{\boldsymbol{n}}\boldsymbol{u}$ are larger. The investigation on ν_t for the cases with $C_\alpha = 1$ and $C_\alpha = 0$ are done by using the following chain of commands:

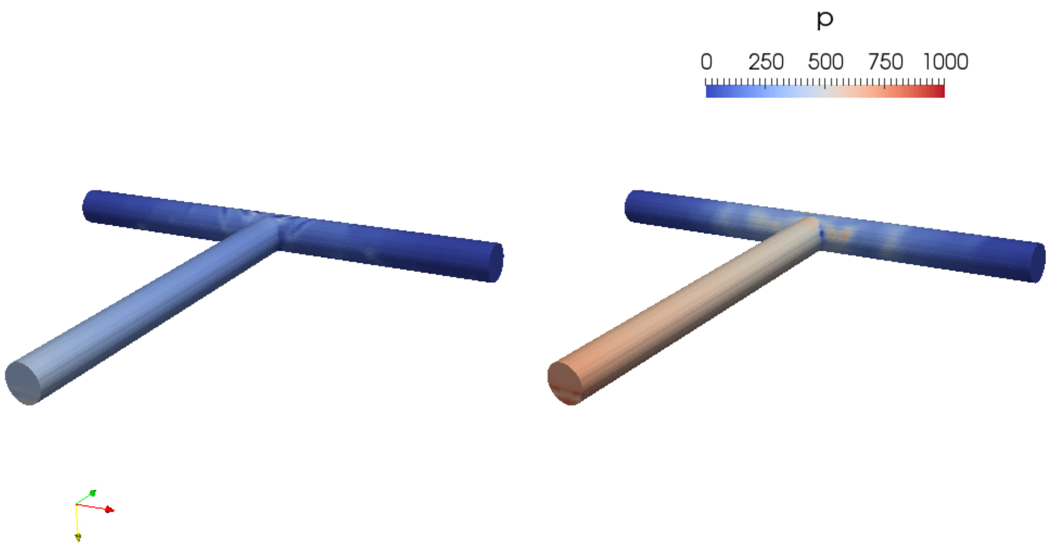


Figure 7.4.3: The effect of interface compression on the wall pressure for $C_\alpha = 1$ (left) and $C_\alpha = 0$ (right). Snapshot taken at $t = 5$ [s] with the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets.

```
alias nut="
patchAverage nut walls | grep -oP ’(?<=\[3\] = )[\- 0-9.]+’ | tail -n +2  > nut_x.txt;
ls -d */ | grep ’[0-9]’ | sort -n  | tail -n +2  |  sed ’s/.$//’ > times.txt;
paste times.txt nut_x.txt > nut.txt; rm -rf nut_x.txt";
```

When the calculated values at the first inner node from the wall for ν_t are calculated for both cases the values from table 7.4.1 are found.

Table 7.4.1: Calculated values of ν_t at the first inner node from the wall for the cases with $C_\alpha = 1$ and $C_\alpha = 0$ for $t = 2$ [s] to $t_{end} = 10$ [s].

Case	μ_{ν_t} (σ_{ν_t}) [m^2 s^{-1}]
$C_\alpha = 1$	$5.91 \cdot 10^{-5}$ $(0.17 \cdot 10^{-5})$
$C_\alpha = 0$	$6.88 \cdot 10^{-5}$ $(0.39 \cdot 10^{-5})$

The calculation of $\nabla_{\boldsymbol{n}} \boldsymbol{u}$ is done by `magGradU`, which applies `foamCalc`:

```
foamCalc magGrad U
```

This result is used in the calculation of the average of `magGradU` at the walls:

```
alias magGradU="
patchAverage magGradU walls  | grep -oP ’(?<=\[3\] = )[\- 0-9.]+’ > magGradU_x.txt;
ls -d */ | grep ’[0-9]’ | sort -n  | tail -n +2  |  sed ’s/.$//’ > times.txt;
paste times.txt magGradU_x.txt > magGradU.txt; rm -rf magGradU_x.txt";
```

The results for $\nabla_{\boldsymbol{n}} \boldsymbol{u}$ are summarised in table 7.4.2 and are shown in figure 7.4.4.

Table 7.4.2: Calculated $\nabla_{\boldsymbol{n}}\boldsymbol{u}$ at the walls for the cases with $C_\alpha = 1$ and $C_\alpha = 0$ for $t = 2$ [s] to $t_{end} = 10$ [s].

Case	$\mu_{\nabla_{\boldsymbol{n}}\boldsymbol{u}}$	$(\sigma_{\nabla_{\boldsymbol{n}}\boldsymbol{u}})$ [m s^{-2}]
$C_\alpha = 1$	$5.6 \cdot 10^3$	$(0.18 \cdot 10^3)$
$C_\alpha = 0$	$6.9 \cdot 10^3$	$(0.45 \cdot 10^3)$

These simulation results can help to explain the additional pressure loss when interface compression is turned off. The additional turbulent viscosity and the larger velocity gradients at the wall are causing this larger pressure drop. From this analysis can be concluded that the correct simulation of the flow regime is necessary for the correct prediction of the pressure drop in multiphase pipe flow.

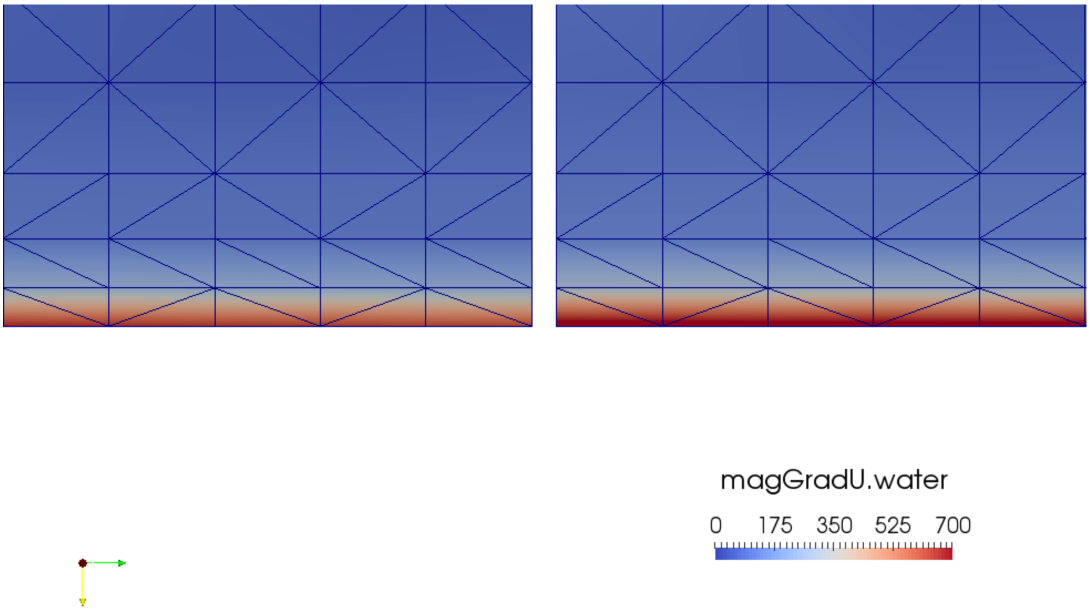


Figure 7.4.4: Zoomed visualisation of `mag grad U` at the wall of the inlet pipe for $C_\alpha = 1$ (left) and $C_\alpha = 0$ (right). Snapshot taken at $t = 5$ [s] on the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets.

7.4.2 Production difference

A smaller relative difference for the production is found than was found for the pressure for cases with and without interface compression enabled. A small discrepancy is found when a pressure difference is applied over the outlets, as can be seen in figure 7.4.6 and figure 7.4.8 for the liquid production and the gas production, respectively. When no pressure difference between the outlets is applied, both methods estimate symmetrical outflow, as expected. When interface compression is disabled, one can see that steady production is reached earlier with respect to the $C_\alpha = 1$ case.

As expected, the effect of interface compression on the estimation of the production of gas and liquid is small. Table 7.4.3 summarises the production for the four investigated conditions. It can be concluded that further research towards interface compression can be beneficial. While this report focusses on `multiphaseEulerFoam`, there exists an OpenFOAM solver called `twoPhaseEulerFoam`, in which there is no interface compression. Comparison of these solvers and validation against experiments can help in the further development of these multiphase flow solvers.

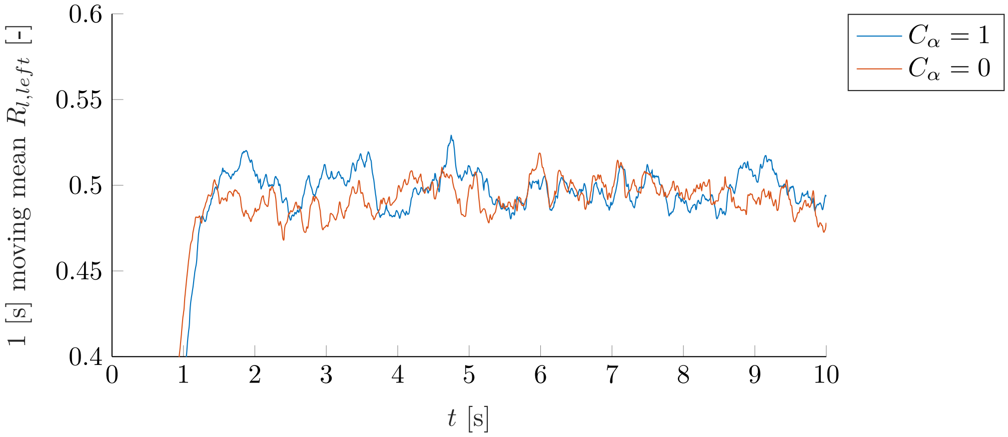


Figure 7.4.5: The effect of interface compression (1 [s] moving mean) on the liquid production with the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets.

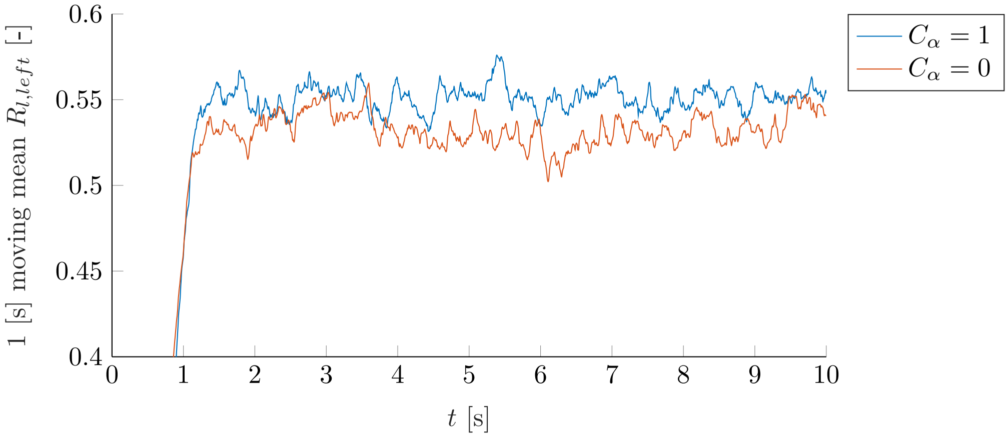


Figure 7.4.6: The effect of interface compression (1 [s] moving mean) on the liquid production with the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa] between the outlets.

Table 7.4.3: Calculated $R_{k,l}$ for cases $C_\alpha = 1$ and $C_\alpha = 0$ for $t = 2$ [s] to $t_{end} = 10$ [s].

C_α	Δp	$\mu_{R_{l,left}}$ ($\sigma_{R_{l,left}}$) [-]		$\mu_{R_{g,left}}$ ($\sigma_{R_{g,left}}$) [-]	
1	0	0.50	(0.12)	0.5	(0.021)
1	250	0.55	(0.16)	0.66	(0.032)
0	0	0.49	(0.13)	0.50	(0.014)
0	250	0.53	(0.15)	0.57	(0.020)

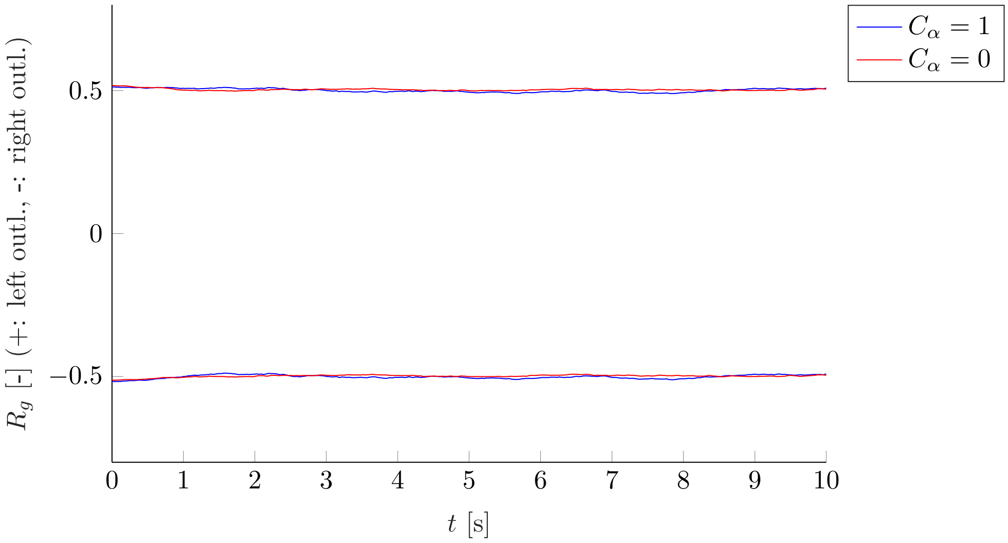


Figure 7.4.7: The effect of interface compression (1 [s] moving mean) on the gas production with the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 0$ [Pa] between the outlets.

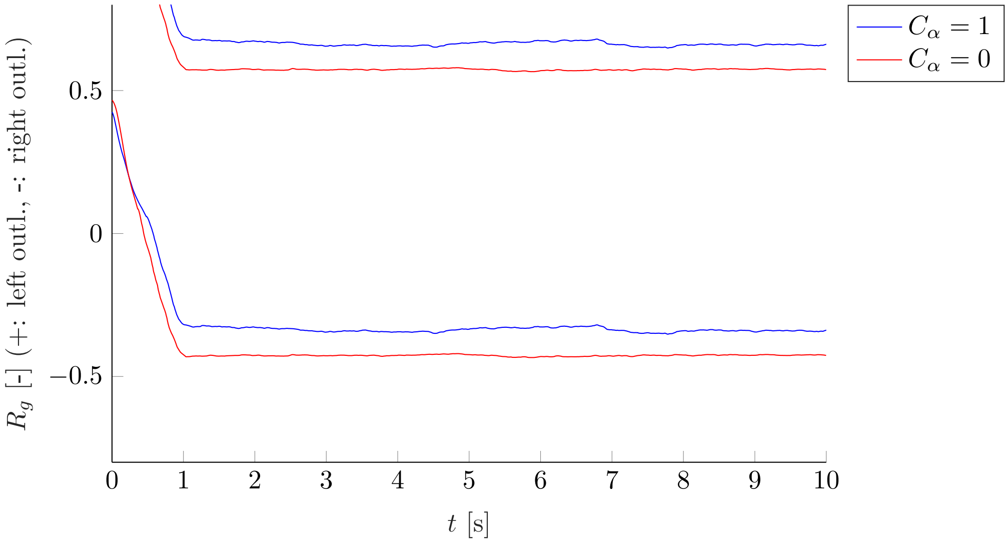


Figure 7.4.8: The effect of interface compression (1 [s] moving mean) on the gas production with the **T2** mesh with $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa] between the outlets.

7.5 UTILISATION OF CRANK-NICOLSON SCHEMES INSTEAD OF THE EULER SCHEME

In this thesis the Euler scheme (equation 3.3.1) is used. Contrary to all spatial schemes used in the simulations, this is a first order accurate scheme. The temporal discretisation can be improved to second-order accuracy by using the Crank-Nicolson scheme which is already tested with varying results [64].

A simulation of the T-junction mesh is done for a pure Crank-Nicolson time discretisation. Unfortunately, the scheme does not guarantee boundedness and the simulation crashes before $t = 0.01$ [s]. In order to stabilise the simulation, a blended scheme with $\psi = 0.5$ (while a scheme with is $\psi = 0.9$ recommended[1]) is used.

$$\psi = \begin{cases} 1 & \text{Corresponds to pure } \texttt{CrankNicolson} \\ 0 & \text{Corresponds to pure } \texttt{Euler} \end{cases} \tag{7.5.1}$$

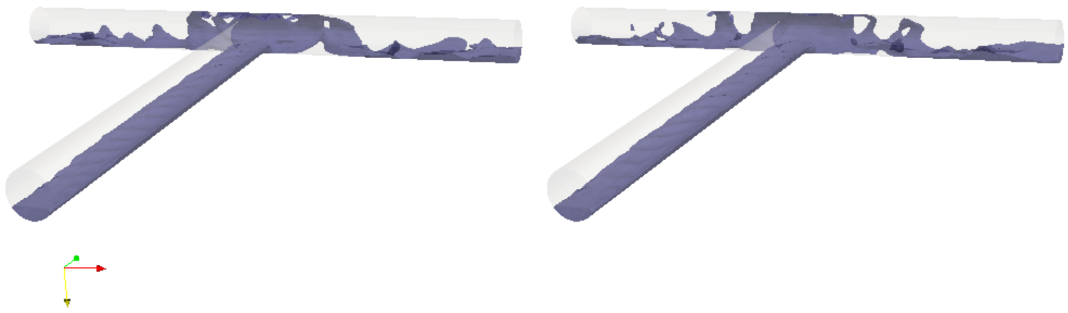


Figure 7.5.1: A comparison between results from the `Euler` scheme (left) and the `CrankNicolson` scheme (right) at $t = 4.10$ [s].

As can be seen from figure 7.5.1, the liquid behaves in a similar way. The development of the pressure and the liquid production is shown in figure 7.5.2 and figure 7.5.3. Table 7.5.1 provides an overview of the differences between the results from both simulations. The differences between the schemes are small. More simulations with larger time periods could definitely provide more insight on the effect of second order time schemes.

[1] Recommended by the OpenFOAM manual

Table 7.5.1: Calculated inlet pressure and normalised production for two time schemes for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa]. For `Euler`: $t_{end} = 10$ [s], for `CrankNicolson`: $t_{end} = 8.67$ [s].

Time scheme	$\mu_{p_{inlet}}$ ($\sigma_{p_{inlet}}$) [Pa]	$\mu_{R_{l,left}}$ ($\sigma_{R_{l,left}}$) [-]	$\mu_{R_{g,left}}$ ($\sigma_{R_{g,left}}$) [-]
`Euler`	643 (106)	0.55 (0.16)	0.66 (0.032)
`CrankNicolson`	639 (106)	0.55 (0.16)	0.66 (0.032)

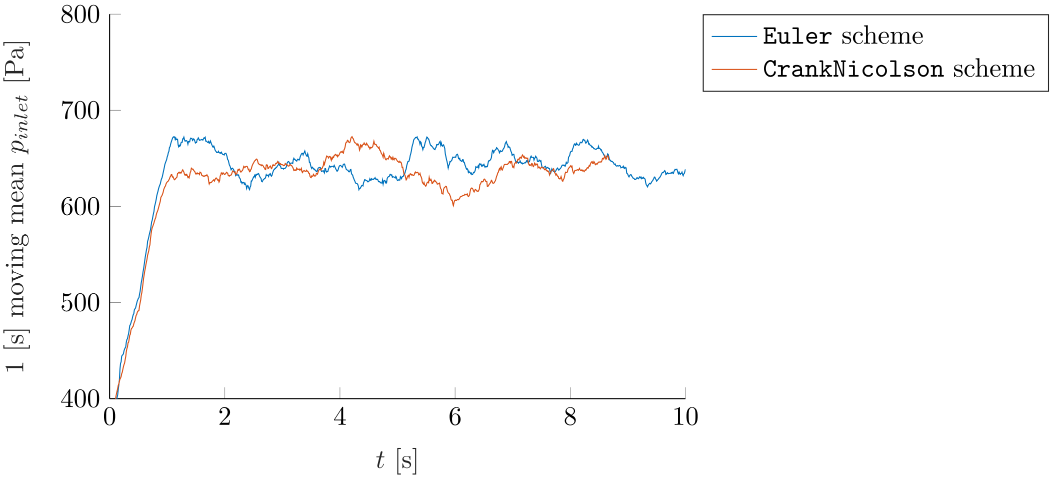


Figure 7.5.2: Difference between inlet pressure for the `Euler` and `CrankNicolson` time discretisation scheme. With $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa] between the outlets.

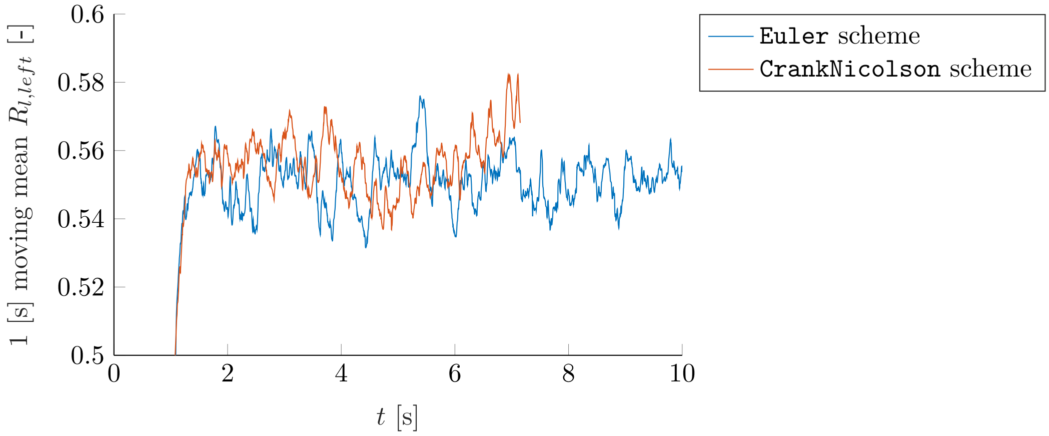


Figure 7.5.3: Liquid production difference for the `Euler` and `CrankNicolson` time discretisation scheme. With $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}] and $\Delta p = 250$ [Pa] between the outlets.

7.6 CONCLUDING REMARKS

This chapter discussed future research opportunities for using OpenFOAM. The most important conclusion that is drawn is that the results can be improved with longer simulation times and a longer entrance length. Besides those improvements, there are additional research opportunities in the field of `multiphaseEulerFoam`, such as by using an improved SGS, by investigating the effect of interface compression or by using second order time schemes. Numerous research can be done in the field of simulating pipe flow with the Euler-Euler solver. Therefore this thesis can additionally serve as a resource for future simulations to be set-up in `multiphaseEulerFoam`.

Chapter 8

Conclusions and recommendations

Multiphase flows are common in engineering applications such as nuclear facilities and oil and gas production systems. In this study a T-junction to a dual riser system with an inner diameter of $d = 0.05$ [m] is investigated. This geometry is one of the proposed designs for the upwards transport of fluids from a sub-sea pipeline to a floating liquefied natural gas vessel. The scope of this thesis is limited to two OpenFOAM solvers: `interFoam` which uses a mixture momentum formulation and `multiphaseEulerFoam` which uses a separate momentum equation for each phase. Both solvers use a Volume of Fluid (VOF) formulation as a method to capture the interface between the phases. The analysis of the differences and similarities in the resulting flow regime, the inlet pressure, the production at the outlets and the hold-up is the central research objective of this thesis.

The `snappyHexMesh` (SHM) utility is used to construct three T-junction meshes. From coarse to fine these are named **T1**, **T2** and **T3**. These meshes have 23042, 47544 and 95292 cells, respectively. Two riser system meshes are built: **R1** (67720 cells) and **R2** (276784 cells). The generated meshes are non-symmetrical, which is a result of the non-symmetrical snapping procedure of SHM. This non-symmetry may trigger a maldistribution of each phase at the outlets.

Air and water are brought into the domain with a total volumetric flow rate of 64.2 [m^3 h^{-1}]. The volumetric flow rate of water in the simulations is set to 1, 2 and 3 [m^3 h^{-1}]. A pressure difference between the outlets of the risers is applied with values between $\Delta p = 0$ and 1000 [Pa]. The Courant-Friedrichs-Lewy condition of 0.25 and the Pressure Implicit with Splitting of Operator (PISO) algorithm with two corrector loops are used to obtain sufficient numerical stability.

Various differences and similarities in the results from both solvers are found in this study. The most important conclusions are the following:

- The pressures at the inlet of the T-junction with an average cell volume of $9.50 \cdot 10^{-8}$ [m^3] (**T1**) and $4.60 \cdot 10^{-8}$ [m^3] (**T2**) do not converge to the inlet pressure results of the mesh with an average cell volume of $2.30 \cdot 10^{-8}$ [m^3] (**T3**). A finer mesh is needed to conclude whether the applied finest mesh can provide accurate results for the inlet pressure.
- The liquid and gas production rates at the outlets of the T-junction mesh with an average cell volume of $4.60 \cdot 10^{-8}$ [m^3] (**T2**) are similar to the values obtained on the finest mesh with an average cell volume of $2.30 \cdot 10^{-8}$ [m^3] (**T3**).
- For a water flow rate of $\dot{Q}_{l,inlet} = 2$ or 3 [m^3 h^{-1}], the inlet pressure is similar with both solvers. However, when the discharge is as low as $\dot{Q}_{l,inlet} = 1$ [m^3 h^{-1}], the resulting inlet pressure from `interFoam` is higher.
- The production through the two outlets is similar with both solvers when no pressure difference is applied. When at the right outlet a pressure of 250 [Pa] is applied, `multiphaseEulerFoam` gives a

lower outflow at the left outlet than interFoam.

- The liquid hold-up in the T-junction is higher for multiphaseEulerFoam for low values of the water flow rate at the inlet. However, when the inflow is increased, interFoam calculates a higher liquid hold-up in the domain.
- While the mean results in the T-junction for the production of water and air at the two outlets are the same when no pressure difference between the outlets is applied, using multiphaseEulerFoam results in a more oscillating production pattern over time.
- Wardle and Weller [93] concluded that multiphaseEulerFoam has a 39% higher computational load *per time step* than interFoam. However, in this research the difference per time step is much larger, namely between 115 and 152%. When an adjustable run time with a Courant-Friedrichs-Lewy (CFL) condition of 0.25 is used, multiphaseEulerFoam needs 124% more wall clock time than interFoam. The interFoam solver is thus preferred in simulations where the results from interFoam and multiphaseEulerFoam are similar, for example when the T-junction has enough liquid throughput and when there is no pressure difference over the outlets.
- The flow pattern in interFoam is non-physical after the T-junction, while multiphaseEulerFoam gives a churn-like behaviour.
- Due to the lower liquid hold-up in the riser system in multiphaseEulerFoam, the resulting inlet pressure is lower than in interFoam.
- The production at the outlets over time for the riser system shows a more oscillating pattern in multiphaseEulerFoam compared to the results of interFoam. This is in line with the conclusion for the oscillations found in the results in the T-junction.
- The inlet pressure and pressure loss in the left riser of both OpenFOAM solvers are higher compared to the results of ANSYS FLUENT [95].
- The liquid hold-up is in the left riser is similar with multiphaseEulerFoam and ANSYS FLUENT for $\dot{Q}_{l,inlet} = 2$ [m^3 h^{-1}]. By using interFoam, a higher liquid hold-up in the left riser is obtained.
- The hold-up and pressure in the riser system in both solvers differ substantially from the experiments [96]. The relatively coarse mesh is not able to capture the interface in such a way that a churn/slug flow regime is obtained. In interFoam and multiphaseEulerFoam the liquid hold-up and the pressure losses in the riser are vastly underestimated. A recommendation for accurate modelling of the vertical riser section is to increase the fineness of the mesh.

From the conclusions and the sensitivity analysis a number of recommendations are formulated. These recommendations are the following:

- Due to the lower computational loads, the use of interFoam is recommended when simulating an impacting T-junction with enough liquid inflow and no pressure difference between the outlets.
- In this thesis a simulation time of 10 [s] is used. An increase of the simulation time will allow to verify that full stationary results have been obtained with the Euler-Euler model for pipe flow
- An additional analysis on the riser system should use a finer mesh in order to correctly model the flow regime, such that the liquid hold-up and pressure losses from the experiments can be found.
- A stratified flow regime is imposed on the inlet, while a wavy stratified or a slug flow regime is found in the experiments. An additional study on the flow pattern modelling in a horizontal pipe is recommended
- From the literature it is known that the Smagorinsky Sub-Grid Scale (SGS) model does not perform well in the vicinity of walls. Simulations with other SGS models are thus recommended.
- Turning interface compression on has a large effect on the inlet pressure. An opportunity for further study is to investigate how the interface compression coefficient influences the results.

This study presented the results of simulations of multiphase pipe flow obtained with both the OpenFOAM mixed-momentum VOF solver `interFoam` and the OpenFOAM Euler-Euler coupled VOF solver `multiphaseEulerFoam`. The latter is a rather new solver and results available on the performance of this solver in the open literature are scarce. It is found that the flow regime and the liquid hold-up have a noteworthy influence on the calculated pressure, which means that interface compression is required to correctly simulate multiphase pipe flow. By using `multiphaseEulerFoam` a realistic flow pattern is obtained, due to its Euler-Euler formulation. The use of this open-source solver can therefore be beneficial for many engineering applications, especially in the oil and gas industry. This research can therefore not only be seen as an analysis of two solvers, but also as a valuable resource for new research projects with the OpenFOAM `multiphaseEulerFoam` solver.

Bibliography

[1] Abdelsamie, A., Fru, G., Oster, T., Dietzsch, F., Janiga, G. & Thévenin, D. (2016). Towards direct numerical simulations of low-Mach number turbulent reacting and two-phase flows using immersed boundaries. *Computers and Fluids*, 131, 123–141. doi:10.1016/j.compfluid.2016.03.017.

[2] Abdul-Majeed, G. H. & Abu Al-Soof, N. B. (2000). Estimation of gas-oil surface tension. *Journal of Petroleum Science and Engineering*, 27(3-4), 197–200. doi:10.1016/S0920-4105(00)00058-9.

[3] Abdulkadir, M., Hernandez-perez, V., Lo, S., Lowndes, I. S. & Azzopardi, B. J. (2015). Comparison of experimental and Computational Fluid Dynamics (CFD) studies of slug flow in a vertical riser. *Experimental Thermal and Fluid Science*, 68, 468–483. doi:10.1016/j.expthermflusci.2015.06.004.

[4] Aidun, C. K., Clausen, J. R. & Woodruff, G. W. (2010). Lattice-Boltzmann Method for Complex Flows. *Annual Review of Fluid Mechanics*, 42, 439–72. doi:10.1146/annurev-fluid-121108-145519.

[5] Albadawi, A., Delauré, Y., Donoghue, D. B., Robinson, A. & Murray, D. B. (2012). Numerical investigation of volume of fluid and level set interface capturing methods for bubble growth and detachment. *Journal of Physics: Conference Series*, 395(1), 012166. doi:10.1088/1742-6596/395/1/012166.

[6] Alfonsi, G. & Primavera, L. (2007). The structure of turbulent boundary layers in the wall region of plane channel flow. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2078), 593–612. doi:10.1098/rspa.2006.1785.

[7] Ali, S. F. & Yeung, H. (2010). Experimental Investigation and Numerical Simulation of Two-Phase Flow in a Large-Diameter Horizontal Flow Line Vertical Riser. *Petroleum Science and Technology*, 28(11), 1079–1095. doi:10.1080/10916460902780327.

[8] Auton, T. R. (1987). The lift force on a spherical rotational flow. *J . Fluid Mech*, 183(1987), 199–218. doi:10.1017/S002211208700260X.

[9] Badalassi, V. E., Ceniceros, H. D. & Banerjee, S. (2003). Computation of multiphase systems with phase field models. *Journal of Computational Physics*, 190(2), 371–397. doi:10.1016/S0021-9991(03)00280-8.

[10] Baehr, H. D. & Stephan, K. (1998). *Heat and Mass transfer*. Springer-Verlag Berlin Heidelberg, 2nd edition. ISBN 9783540295266. doi:10.1007/3-540-29527-5.

[11] Berselli, L. C., Iliescu, T. & William, J. (2005). *Mathematics of Large Eddy Simulation of Turbulent Flows, First Edition*. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-26316-6. doi:10.1007/b137408.

[12] Breuer, M., Lakehal, D. & Rodi, W. (1996). Flow around a surface mounted cubical obstacle: Comparison of LES and RANS–results. *Notes on Numerical Fluid Mechanics*, 49, 22–30. doi:10.1007/978-3-322-89838-8_4.

[13] Caboussat, A. (2005). Numerical Simulation of Two-Phase Free Surface Flows. *Archives of Computational Methods in Engineering*, 12(2), 165–224. doi:10.1007/BF03044518.

[14] Capecelatro, J., Pepiot, P. & Desjardins, O. (2014). Numerical characterization and modeling of particle clustering in wall-bounded vertical risers. *Chemical Engineering Journal*, 245, 295–310. doi:10.1016/j.cej.2014.02.040.

[15] Černe, G., Petelin, S. & Tiselj, I. (2001). Coupling of the Interface Tracking and the Two-Fluid Models for the Simulation of Incompressible Two-Phase Flow. *Journal of Computational Physics*, 171(2), 776–804. doi:10.1006/jcph.2001.6810.

[16] Chen, J. L., He, L. M., Luo, X. M., Bai, H. T. & Wei, Y. H. (2012). Simulation of oil-water two phase flow and separation behaviors in combined T junctions. *Journal of Hydrodynamics*, 24(6), 848–857. doi:10.1016/S1001-6058(11)60312-0.

[17] Chen, Q., Xu, J., Sun, D., Cao, Z., Xie, J. & Xing, F. (2013). Numerical simulation of the modulated flow pattern for vertical upflows by the phase separation concept. *International Journal of Multiphase Flow*, 56, 105–118. doi:10.1016/j.ijmultiphaseflow.2013.05.014.

[18] Courant, R., Friedrichs, K. & Lewy, H. (1928). Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1), 32–74. doi:10.1007/BF01448839.

[19] Crafton, J. (1997). Oil and gas well evaluation using the reciprocal productivity index method. In *The 1997 SPE Production Operations Symposium*, 199–208. doi:10.2118/37409-MS.

[20] Crank, J. & Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(01), 50–67. doi:10.1007/BF02127704.

[21] Da Riva, E. & Del Col, D. (2009). Numerical simulation of churn flow in a vertical pipe. *Chemical Engineering Science*, 64(17), 3753–3765. doi:10.1016/j.ces.2009.04.049.

[22] Damián, S. M. *An Extended Mixture Model for the Simultaneous Treatment of Short and Long Scale Interfaces*. PhD thesis, (2013). doi:10.13140/RG.2.1.3182.8320.

[23] Day, M. A. (1990). The no-slip condition of fluid dynamics. *Erkenntnis*, 33(3), 285–296. doi:10.1007/BF00717588.

[24] Dean, J. A. (1999). *Lange's Handbook of Chemistry*. McGraw-Hill, 15th edition. ISBN 0070163847. doi:10.1080/10426919008953291.

[25] Deardorff, J. W. (1970). A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, 41(02), 453–480. doi:10.1017/S0022112070000691.

[26] Deshpande, S. S., Anumolu, L. & Trujillo, M. F. (2012). Evaluating the performance of the two-phase flow solver interFoam. *Computational Science & Discovery*, 5(1), 014016. doi:10.1088/1749-4699/5/1/014016.

[27] Eckhardt, B., Schneider, T. M., Hof, B. & Westerweel, J. (2007). Turbulence Transition in Pipe Flow. *Annual Review of Fluid Mechanics*, 39(1), 447–468. doi:10.1146/annurev.fluid.39.050905.110308.

[28] Fabritius, B. & Tabor, G. (2016). Improving the quality of finite volume meshes through genetic optimisation. *Engineering with Computers*, 32(3), 425–440. doi:10.1007/s00366-015-0423-0.

[29] Fanning, J. T. (1896). *A practical treatise on hydraulic and water-supply engineering*. ISBN 978-5-87581-042-8.

[30] Fox, R. O. (2012). Large-Eddy-Simulation Tools for Multiphase Flows. *Annual Review of Fluid Mechanics*, 44(1), 47–76. doi:10.1146/annurev-fluid-120710-101118.

[31] Geurst, J. A. (1985). Virtual mass in two-phase bubbly flow. *Physica A: Statistical Mechanics and its Applications*, 129(2), 233–261. doi:10.1016/0378-4371(85)90168-2.

[32] Ghannam, M. T., Hasan, S. W., Abu-Jdayil, B. & Esmail, N. (2012). Rheological properties of heavy & light crude oil mixtures for improving flowability. *Journal of Petroleum Science and Engineering*, 81, 122–128. doi:10.1016/j.petrol.2011.12.024.

[33] Gopala, V. R. & van Wachem, B. G. M. (2008). Volume of fluid methods for immiscible-fluid and free-surface flows. *Chemical Engineering Journal*, 141(1-3), 204–221. doi:10.1016/j.cej.2007.12.035.

[34] Hart, J., Hamersma, P. J. & Fortuin, J. M. H. (1989). Correlations predicting frictional pressure drop and liquid holdup during horizontal gas-liquid pipe flow with a small liquid holdup. *International Journal of Multiphase Flow*, 15(6), 947–964. doi:10.1016/0301-9322(89)90023-2.

[35] Harten, A. (1982). High Resolution Schemes for Hyperbolic Conservation Laws. *Journal of Computational Physics*, 135, 260–278. doi:10.1006/jcph.1997.5713.

[36] Hernandez-Perez, V., Abdulkadir, M. & Azzopardi, B. (2011). Grid Generation Issues in the CFD Modelling of Two-Phase Flow in a Pipe. *The Journal of Computational Multiphase Flows*, 3(1), 13–26. doi:10.1260/1757-482x.3.1.13.

[37] Hirt, C. W. & Nichols, B. D. (1981). Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1), 201–225. doi:10.1016/0021-9991(81)90145-5.

[38] Höhne, T. & Vallée, C. (2010). Experiments and numerical simulations of horizontal two-phase flow regimes using an interfacial area density model. *The Journal of Computational Multiphase Flows*, 2(3), 131–143. doi:10.1260/1757-482X.2.3.131.

[39] Issa, R. I. (1986). Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1), 40–65. doi:10.1016/0021-9991(86)90099-9.

[40] Jasak, H. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, (1996). doi:10.1016/S0020-7683(02)00168-3.

[41] Jiang, X., Siamas, G. A., Jagus, K. & Karayiannis, T. G. (2010). Physical modelling and advanced simulations of gas-liquid two-phase jet flows in atomization and sprays. *Progress in Energy and Combustion Science*, 36(2), 131–167. doi:10.1016/j.pecs.2009.09.002.

[42] Kang, M., Fedkiw, R. & Liu, X. (2000). A Boundary Condition Capturing Method for Multiphase Incompressible Flow. *Journal of Scientific Computing*, 15(3), 323–360. doi:10.1023/A:1011178417620.

[43] Keough, S. Optimising the Parallelisation of OpenFOAM Simulations. Technical report, (2014).

[44] Kolmogorov, A. N. (1962). A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high Reynolds number. *Journal of Fluid Mechanics*, 13(01), 82–85. doi:10.1017/S0022112062000518.

[45] Kunkelmann, C. & Stephan, P. (2009). CFD Simulation of Boiling Flows Using the Volume-of-Fluid Method within OpenFOAM. *Numerical Heat Transfer, Part A: Applications*, 56(8), 631–646. doi:10.1080/10407780903423908.

[46] Leonard, A. (1975). Energy Cascade in Large-Eddy Simulations of Turbulent Fluid Flows. *Advances in Geophysics*, 18(A), 330–340. doi:10.1016/S0065-2687(08)60464-1.

[47] Lien, K., Monty, J. P., Chong, M. S. & Ooi, A. (2004). The Entrance Length for Fully Developed Turbulent Channel Flow. In *15th Australasian Fluid Mechanics Conference*, number December, 1–4.

[48] Liu, Y. & Hinrichsen, O. (2014). CFD modeling of bubbling fluidized beds using OpenFOAM®: Model validation and comparison of TVD differencing schemes. *Computers and Chemical Engineering*, 69, 75–88. doi:10.1016/j.compchemeng.2014.07.002.

[49] Magnini, M., Pulvirenti, B. & Thome, J. R. (2016). Characterization of the velocity fields generated by flow initialization in the CFD simulation of multiphase flows. *Applied Mathematical Modelling*, 40, 6811–6830. doi:10.1016/j.apm.2016.02.023.

[50] Malekzadeh, R., Henkes, R. A. W. M. & Mudde, R. F. (2012). Severe slugging in a long pipeline-riser system: Experiments and predictions. *International Journal of Multiphase Flow*, 46, 9–21. doi:10.1016/j.ijmultiphaseflow.2012.06.004.

[51] Mayor, T. S., Pinto, A. M. F. R. & Campos, J. B. L. M. (2007). Hydrodynamics of gas-liquid slug flow along vertical pipes in the laminar regime-experimental and simulation study. *Industrial and Engineering Chemistry Research*, 46(11), 3794–3809. doi:10.1021/ie0609923.

[52] McNamara, G. R. & Zanetti, G. (1988). Use of the boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61(20), 2332–2335. doi:10.1103/PhysRevLett.61.2332.

[53] Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8), 1–4. doi:10.1109/N-SSC.2006.4785860.

[54] Moser, R. D., Kim, J. & Mansour, N. N. (2015). Direct numerical simulation of turbulent channel flow up to. *Journal of Fluid Mechanics*, 774(1999), 395–415. doi:10.1017/jfm.2015.268.

[55] Murakami, S. (1998). Overview of turbulence models applied in CW-1997. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76, 7–22. doi:10.1016/S0167-6105(98)00004-X.

[56] Nicoud, F. & Ducros, F. (1999). Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow, Turbulence and Combustion*, 62(3), 183–200. doi:10.1023/A:1009995426001.

[57] Noh, W. F. & Woodward, P. (1976). SLIC (Simple Line Interface Calculation). In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede*, 330–340. Springer Berlin Heidelberg. ISBN 978-3-540-37548-7. doi:10.1007/3-540-08004-X_336.

[58] Oliemans, R. V. A. (2001). *Applied Multiphase Flow*. Delft University of Technology.

[59] Patankar, S. (1980). *Numerical heat transfer and fluid flow*. CRC Press. ISBN 9780891165224. doi:10.1016/j.watres.2009.11.010.

[60] Pohorecki, R., Moniuk, W., Bielski, P., Sobieszuk, P. & Dabrowiecki, G. (2005). Bubble diameter correlation via numerical experiment. *Chemical Engineering Journal*, 113(1), 35–39. doi:10.1016/j.cej.2005.08.007.

[61] Pope, S. B. (2010). Self-conditioned fields for large-eddy simulations of turbulent flows. *Journal of Fluid Mechanics*, 652, 139–169. doi:10.1017/S0022112009994174.

[62] Pope, S. B. (2010). *Turbulent Flows*, volume 1. Cambridge University Press, 1st edition. ISBN 9780511840531. doi:10.1088/0957-0233/12/11/705.

[63] Puckett, E. G., Almgren, A. S., Bell, J. B., Marcus, D. L. & Rider, W. J. (1997). A High-Order Projection Method for Tracking Fluid Interfaces in Variable Density Incompressible Flows. *Journal of Computational Physics*, 130, 269–282. doi:10.1006/jcph.1996.5590.

[64] Rasam, A., Wallin, S., Brethouwer, G. & Johansson, A. V. (2013). Large-Eddy Simulation Using the Explicit Algebraic Subgrid Model in Complex Geometries. *TSFP Digital library online*, 1, 1–5.

[65] Reynolds, O. (1895). On the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion. *Philosophical Transactions of the Royal Society of London*, 186, 123–164. doi:10.1098/rsta.1895.0004.

[66] Richardson, L. W. (1922). *Weather Prediction by Numerical Process*. Cambridge University Press. ISBN 9788578110796. doi:10.1017/CBO9780511618291.

[67] Robertson, E., Choudhury, V., Bhushan, S. & Walters, D. K. (2015). Validation of OpenFOAM numerical methods and turbulence models for incompressible bluff body flows. *Computers and Fluids*, 123, 122–145. doi:10.1016/j.compfluid.2015.09.010.

[68] Rusche, H. *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*. PhD thesis, (2002). doi:10.1145/1806799.1806850.

[69] Saad, Y. & van der Vorst, H. A. (2000). Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1), 1–33. doi:10.1016/S0377-0427(00)00412-X.

[70] Santarelli, C. & Fröhlich, J. (2016). Direct Numerical Simulations of spherical bubbles in vertical turbulent channel flow. Influence of bubble size and bidispersity. *International Journal of Multiphase Flow*, 81, 27–45. doi:10.1016/j.ijmultiphaseflow.2016.01.004.

[71] Scala, F. (2013). *Particle characterization and behavior relevant to fluidized bed combustion and gasification systems*. ISBN 9780857095411. doi:10.1533/9780857098801.1.42.

[72] Scardovelli, R. & Zaleski, S. (1999). Direct Numerical Simulation of Free-Surface and Interfacial Flow. *Annual Review of Fluid Mechanics*, 31(1), 567–603. doi:10.1146/annurev.fluid.31.1.567.

[73] Schiller, L. & Naumann, Z. A. (1933). A Drag coefficient correlation. *Z. Ver. Deutsch Ing.*, 77, 318–320.

[74] Schmitt, F. G. (2007). About Boussinesq's turbulent viscosity hypothesis: historical remarks and a direct evaluation of its validity. *Comptes Rendus - Mecanique*, 335(9-10), 617–627. doi:10.1016/j.crme.2007.08.004.

[75] Smagorinsky, J. (1963). General Circulation Experiments With the Primitive Equations. *Monthly Weather Review*, 91(3), 99–164. doi:10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[76] Spedding, P. L. & Chen, J. J. J. (1984). Holdup in Two Phase Flow.Pdf. *International Journal of Multiphase Flow*, 10(3), 307–339. doi:10.1016/0301-9322(84)90024-7.

[77] Sridhar, G. & Katz, J. (1995). Drag and lift forces on microscopic bubbles entrained by a vortex. *Physics of Fluids*, 7(2), 389. doi:10.1063/1.868637.

[78] Stüben, K. (2001). A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1-2), 281–309. doi:10.1016/S0377-0427(00)00516-1.

[79] Sussman, M. *A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow*. PhD thesis, (1994). doi:10.1006/jcph.1994.1155.

[80] Sussman, M. & Puckett, E. G. (2000). A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows. *Journal of Computational Physics*, 162(2), 301–337. doi:10.1006/jcph.2000.6537.

[81] Sweby, P. K. (1984). High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws. *SIAM Journal on Numerical Analysis*, 21(5), 995–1011. doi:10.1137/0721062.

[82] Tabor, G. R. & Baba-Ahmadi, M. H. (2010). Inlet conditions for large eddy simulation: A review. *Computers and Fluids*, 39(4), 553–567. doi:10.1016/j.compfluid.2009.10.007.

[83] Taha, T. & Cui, Z. F. (2006). CFD modelling of slug flow in vertical tubes. *Chemical Engineering Science*, 61(2), 676–687. doi:10.1016/j.ces.2005.07.022.

[84] Taitel, Y. & Dukler, A. E. (1976). A model for predicting flow regime transitions in horizontal and near horizontal gas-liquid flow. *AlChe Journal*, 22(1), 47–55. doi:10.1002/aic.690220105.

[85] Taitel, Y., Bornea, D. & Dukler, A. E. (1980). Modelling flow pattern transitions for steady upward gas-liquid flow in vertical tubes. *AlChe Journal*, 26(3), 345–354. doi:10.1002/aic.690260304.

[86] Temmerman, L., Leschziner, M. A., Mellen, C. P. & Fröhlich, J. (2003). Investigation of wall-function approximations and subgrid-scale models in large eddy simulation of separated flow in a channel with streamwise periodic constrictions. *International Journal of Heat and Fluid Flow*, 24 (2), 157–180. doi:10.1016/S0142-727X(02)00222-9.

[87] Tomaselli, P. D. & Christensen, E. D. (2015). Investigation on the Use of a Multiphase Eulerian CFD solver to simulate breaking waves. In *International conference on Ocean, Offshore and Arctic Engineering*, volume 2, 1–10. ISBN 9780791856482. doi:10.1115/OMAE2015-41640.

[88] Turkel, E. (1993). Review of preconditioning methods for fluid dynamics. *Applied Numerical Mathematics*, 12(1-3), 257–284. doi:10.1016/0168-9274(93)90122-8.

[89] van Leer, B. (1979). Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1), 101–136. doi:10.1016/0021-9991(79)90145-1.

[90] van 't Westende, J. M. C. *Droplets in annular-dispersed gas-liquid pipe-flows*. PhD thesis, Delft University of Technology, (2008).

[91] Wang, Z., Yang, J. & Stern, F. (2008). Comparison of particle level set and CLSVOF methods for interfacial flows. *46th AIAA Aerospace Sciences Meeting and Exhibit*, (January), 1–18. doi:10.2514/6.2008-530.

[92] Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., ... Visbal, M. (2013). High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72, 811–845. doi:10.1002/fld.3767.

[93] Wardle, K. E. & Weller, H. G. (2013). Hybrid multiphase CFD solver for coupled dispersed/segregated flows in liquid-liquid extraction. *International Journal of Chemical Engineering*, 1–32. doi:10.1155/2013/128936.

[94] Weller, H. G. A new approach to VOF-based interface capturing methods for incompressible and compressible flow. Technical report, OpenCFD, (2008).

[95] Worthen, R. A. CFD Simulations to Predict Gas-Liquid Flow Splitting from a Single Flowline to Two Vertical Risers. Technical report, Shell Global Solutions International B.V., Amsterdam, (2015).

[96] Worthen, R. A. & Henkes, R. A. W. M. (2015). CFD for the multiphase flow splitting from a single flowline into a dual riser. In *17th International Conference on Multiphase Production Technology, 10-12 June, Cannes, France*, 15.

[97] Yang, L. & Azzopardi, B. J. (2007). Phase split of liquid-liquid two-phase flow at a horizontal T-junction. *International Journal of Multiphase Flow*, 33(2), 207–216. doi:10.1016/j.ijmultiphaseflow.2006.08.004.

A

Mesh Quality

A.1 CHECKMESH OUTPUT FOR MESH T2

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Create polyMesh for time = 0

Time = 0

Mesh stats
    points:           50291
    faces:            145088
    internal faces:   135960
    cells:            47544
    faces per cell:   5.91132
    boundary patches: 4
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     43196
    prisms:        4246
    wedges:        0
    pyramids:      0
    tet wedges:    0
    tetrahedra:    0
    polyhedra:     102
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            5   34
            6   4
            7   64

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
                   Patch    Faces   Points                  Surface topology
                   inlet      176      185  ok (non-closed singly connected)
             outlet_left      176      185  ok (non-closed singly connected)
            outlet_right      176      185  ok (non-closed singly connected)
                   walls     8600     8665  ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-0.325 -0.0249998 -0.5) (0.325 0.0249998 0.0249999)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (1.65485e-18 -1.04128e-17 7.60193e-17) OK.
    Max cell openness = 3.30769e-16 OK.
    Max aspect ratio = 4.85866 OK.
    Minimum face area = 1.65447e-06. Maximum face area = 3.58335e-05.  Face area magnitudes OK.
    Min volume = 7.47209e-09. Max volume = 9.20331e-08.  Total volume = 0.00219733.  Cell volumes OK.
    Mesh non-orthogonality Max: 55.7175 average: 6.25268
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 1.99614 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

Checking geometry...
    Overall domain bounding box (-0.325 -0.0249998 -0.5) (0.325 0.0249998 0.0249999)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (1.65485e-18 -1.04128e-17 7.60193e-17) OK.
    Max cell openness = 3.30769e-16 OK.
    Max aspect ratio = 4.85866 OK.
    Minimum face area = 1.65447e-06. Maximum face area = 3.58335e-05.  Face area magnitudes OK.
    Min volume = 7.47209e-09. Max volume = 9.20331e-08.  Total volume = 0.00219733.  Cell volumes OK.
    Mesh non-orthogonality Max: 55.7175 average: 6.25268
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 1.99614 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

A.2 CHECKMESH OUTPUT FOR MESH R2

```
Create time

Create polyMesh for time = 0

Time = 0

Mesh stats
    points:           294701
    faces:            845716
    internal faces:   794616
    cells:            276784
    faces per cell:   5.9264
    boundary patches: 4
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     250748
    prisms:        21498
    wedges:        0
    pyramids:      0
    tet wedges:    20
    tetrahedra:    0
    polyhedra:     4518
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            4   984
            5   522
            6   4
            7   2360
            8   648

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
                   Patch    Faces   Points                  Surface topology
                   inlet      176      185  ok (non-closed singly connected)
             outlet_left      182      200  ok (non-closed singly connected)
            outlet_right     182      200  ok (non-closed singly connected)
                   walls   50560    51531  ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-0.505 -2.68 -0.5) (0.505 0.025 0.0249998)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (-1.17539e-17 -1.68547e-17 -2.721e-16) OK.
    Max cell openness = 3.56144e-16 OK.
    Max aspect ratio = 9.7143 OK.
    Minimum face area = 1.65634e-06. Maximum face area = 3.61582e-05.  Face area magnitudes OK.
    Min volume = 2.05461e-09. Max volume = 9.12388e-08.  Total volume = 0.0129395.  Cell volumes OK.
    Mesh non-orthogonality Max: 60.486 average: 6.77096
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 2.33386 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

B

Case files

B.1 MESHING SET-UP OF T

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// block definition for a porosity with an angled inlet/outlet
// the porosity is not aligned with the main axes
//

// convertToMeters 1;

a 10;       // Number of elements in x direction
b 50       // Number of elements in y direction
d 100;       // Number of elements in z direction
xi -0.75;      // Minimum x coordinate
xf  0.75;      // Maximum x coordinate
yi -2.9;      // Minimum y coordinate
yf  0.10;      // Maximum y coordinate
zi  -0.60;      // Minimum z coordinate
zf  0.40;      // Maximum z coordinate


vertices
(
    ($xi $yi $zi) //0
    ($xf $yi $zi) //1
    ($xf $yf $zi) //2
    ($xi $yf $zi) //3
    ($xi $yi $zf) //4
    ($xf $yi $zf) //5
    ($xf $yf $zf) //6
    ($xi $yf $zf) //7
);


blocks
(
    // X = 3 Y = 6, Z = 2
    hex (0 1 2 3 4 5 6 7) (30 60 20) simpleGrading (1 1 1)
);

edges
(
);

```

```
patches
(
  patch maxY
  (
    (3 7 6 2)
  )
  patch minX
  (
    (0 4 7 3)
  )
  patch maxX
  (
    (2 6 5 1)
  )
  patch minY
  (
    (1 5 4 0)
  )
  patch minZ
  (
    (0 3 2 1)
  )
  patch maxZ (
    (4 5 6 7)
  )
);

mergePatchPairs
(
);
```

Listing B.1: system/blockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to run
castellatedMesh true;    // make basic mesh ?
snap            true;    // decide to snap back to surface ?
addLayers       true;   // decide to add viscous layers ?


geometry // Load in STL files here
{
  inlet.stl        {type triSurfaceMesh; name inlet;}
  outlet_left.stl  {type triSurfaceMesh; name outlet_left;}
  outlet_right.stl {type triSurfaceMesh; name outlet_right;}
  walls.stl        {type triSurfaceMesh; name walls;}
  //volume.stl       {type triSurfaceMesh; name volume;}
  refinementBox    {type searchableBox; min (-0.5 -2.9 -0.6); max (0.5 0.1 0.4); name domain;}
};

castellatedMeshControls
{
    maxLocalCells 2000000;  //max cells per CPU core
    maxGlobalCells 4000000; //max cells to use before mesh deletion step
    minRefinementCells 0;  //was 0 - zero means no bad cells are allowed during refinement stages
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 100; // was 1 // expansion factor between each high & low refinement zone

    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    features // taken from STL from each .eMesh file created by "SurfaceFeatureExtract" command
    (
       // {file "inlet.eMesh"; level 2;}
      //  {file "outlet_left.eMesh"; level 2;}
       // {file "outlet_right.eMesh"; level 2;}
       // {file "walls.eMesh"; level 0;}
```

```
        {file "volume.eMesh"; level 3;}  // was 2

    );

    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    refinementSurfaces // Surface-wise min and max refinement level
    {
  inlet        {level (0 0);}
        outlet_left  {level (0 0);}
  outlet_right {level (0 0);}
        walls        {level (1 3);}
        //volume       {level (1 3);}
    }


    resolveFeatureAngle 30;  // Resolve sharp angles // Default 30
    refinementRegions        // In descending levels of fine-ness
    {walls {mode distance; levels ((0.001 3) (0.005 3) (0.01 3));}} // was ((0.001 4) (0.003 3) (0.01 2)) || (0.0006 4) (0.002 3) (0.01 2)
    locationInMesh (0 0 0);  //to decide which side of mesh to keep **
    allowFreeStandingZoneFaces false;
}


// Settings for the snapping.
snapControls
{
    nSmoothPatch 5;
    tolerance 4.0;
    nSolveIter 50;
    nRelaxIter 5;
    nFeatureSnapIter 20; // default is 10

// New settings from openfoam 2.2 onwards for SHMesh

implicitFeatureSnap false; // default is false - detects without doing surfaceFeatureExtract
explicitFeatureSnap true; // default is true
multiRegionFeatureSnap false; // deafault is false - detects features between multiple surfaces

}



// Settings for the layer addition.
addLayersControls //add the PATCH names from inside the STL file so STLpatchName_insideSTLName
{
    relativeSizes false;

    layers
    {
        walls
        {
            nSurfaceLayers 1;
        }
    }

    expansionRatio 1.2;
    finalLayerThickness 0.00124;
    minThickness 0.00120;
    nGrow 1;

    featureAngle 110;
    nRelaxIter 3;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.9;
    maxThicknessToMedialRatio 0.9;
    minMedianAxisAngle 130;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
}



// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    maxNonOrtho 55; // was 65
    maxBoundarySkewness 15; // was 20
```

```
    maxInternalSkewness 3; // was 4
    maxConcave 80;// was 80
    minFlatness 0.5;
    minVol 1e-13; //was 1e-13;
    minTetQuality 1e-9; //was 1e-9
    minArea -1;
    minTwist 0.02;
    minDeterminant 0.001;
    minFaceWeight 0.02;
    minVolRatio 0.01;
    minTriangleTwist -1;

    // Advanced

    nSmoothScale 4;
    errorReduction 0.75;
}

// Advanced

debug 0;


// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1E-6;


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

Listing B.2: system/snappyHexMeshDict

B.2 INTERFOAM

B.2.1 0 folder interFoam

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField uniform 0;

boundaryField
{
    inlet
    {
        type            groovyBC;
        refValue        uniform 0;
        value           uniform 0;
        valueExpression "(pos().y > 0.0122965) ? 1 : 0";
    }
    outlet_left
    {
        type            inletOutlet;
        phi             phi;
        inletValue      uniform 0;
        value           uniform 0;
    }
    outlet_right
    {
        type            inletOutlet;
        phi             phi;
        inletValue      uniform 0;
        value           uniform 0;
    }
    walls
    {
        type            zeroGradient;
    }
}


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

Listing B.3: 0/alpha.water

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           $internalField;
    }
    outlet_left
    {
        type            calculated;
        value           $internalField;
    }
    outlet_right
    {
        type            calculated;
        value           $internalField;
    }
    walls
    {
        type            nutkWallFunction;
        value           $internalField;
    }
}

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

Listing B.4: 0/nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

```

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            fixedFluxPressure;
        value           $internalField;
    }
    outlet_left
    {
        type            prghPressure;
        p               $internalField;
        value           $internalField;
    }
    outlet_right
    {
        type            prghPressure;
        p               uniform 0;
        value           uniform 0;
    }
    walls
    {
        type            fixedFluxPressure;
        value           $internalField;
    }
}

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

Listing B.5: 0/p__rgh

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            groovyBC;
        refValue        uniform (0 0 0);
        value           uniform (0 0 0);
        valueExpression "(pos().y > 0.0122965) ? vector(0, 0, 1.339810699) : vector(0, 0,
     11.259495232)";
    }
    outlet_left
    {
        type            pressureInletOutletVelocity;
        phi             phi;
        value           uniform (0 0 0);
    }
    outlet_right
    {
        type            pressureInletOutletVelocity;
        phi             phi;
        value           uniform (0 0 0);
    }
    walls
    {
        type            fixedValue;
  value   $internalField;
    }
}
```

```

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

Listing B.6: 0/U

B.2.2 constant folder interFoam

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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


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

Listing B.7: constant/g

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (water air);

water
{
    transportModel  Newtonian;
    nu              [0 2 -1 0 0 0 0] 1.301301301301301e-6;
    rho             [1 -3 0 0 0 0 0] 999;
}

air
{
    transportModel  Newtonian;
    nu              [0 2 -1 0 0 0 0] 1.424e-5;
    rho             [1 -3 0 0 0 0 0] 1.25;
}

sigma           [1 0 -2 0 0 0 0] 0.0742;

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

Listing B.8: constant/transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  LES;

LES {
        LESModel        Smagorinsky;
  turbulence      on;
  printCoeffs     on;

  delta           cubeRootVol;

  cubeRootVolCoeffs
  {
  }
}
// ************************************************************************* //
```

Listing B.9: constant/turbulenceProperties

B.2.3 system folder interFoam

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


libs (
      "libOpenFOAM.so"
      "libsimpleSwakFunctionObjects.so"
      "libswakFunctionObjects.so"
      "libgroovyBC.so"
     );
functions
{
    #include "sampledSets"
    #include "sampledSurf"
}

application     interFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         10;

deltaT          1e-4;

writeControl    adjustableRunTime;

writeInterval   0.01;

purgeWrite      0;

```

```
writeFormat     ascii;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable no;

adjustTimeStep  yes;

maxCo           1;

maxAlphaCo      0.25;

maxDeltaT       1;

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

Listing B.10: system/controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    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
{
    default    none;

    div(Rc)    Gauss linear;

    "div\(phi,alpha.*\)"                 Gauss vanLeer;
    "div\(phirb,alpha.*\)"               Gauss vanLeer;
    "div\(phir,alpha.*,alpha.*\)"        Gauss vanLeer;

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

    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
    div(rhoPhi,U)     Gauss limitedLinearV 1;

}

laplacianSchemes
{
    default    Gauss linear corrected;
}

interpolationSchemes
{
    default    linear;
}

snGradSchemes
{
    default     corrected;
}

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

Listing B.11: system/fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.*"
    {
    solver               smoothSolver;
    smoother             DICGaussSeidel;
    tolerance            1e-7;
    relTol               0;

    nAlphaCorr           1;
    nAlphaSubCycles      1;
    cAlpha    1;
    }
    p_rgh
    {
        solver           GAMG;
        smoother         DICGaussSeidel;
        tolerance        1e-6;
        relTol           0;

        nCellsInCoarsestLevel 200;
    }
    p_rghFinal
    {
        solver           GAMG;
        smoother         DICGaussSeidel;
        tolerance        1e-7;
        relTol           0;

        nCellsInCoarsestLevel 200;
    }
    pcorr
    {
        $p_rghFinal;
        tolerance        1e-8;
        relTol           0;
    }
    U
    {
        solver           smoothSolver;
        smoother         GaussSeidel;
        tolerance        1e-7;
        relTol           0;
        minIter          1;
    }
    UFinal
    {
        $U;
        tolerance        1e-7;
        relTol           0;

    }
}

PIMPLE
{
  nOuterCorrectors 1;
  nCorrectors      2;
  nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
}

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

Listing B.12: sysem/fvSolution

B.3 MULTIPHASEEULERFOAM

B.3.1 0 folder multiphaseEulerFoam

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions    [0 0 0 0 0 0 0];

internalField uniform 1;

boundaryField
{
    inlet
    {
        type              groovyBC;
        value             uniform 0;
        valueExpression "(pos().y > 0.0122965) ? 0 : 1";
    }
    outlet_left
    {
        type             inletOutlet;
        phi              phi.air;
        inletValue       uniform 1;
        value            uniform 1;
    }
    outlet_right
    {
        type             inletOutlet;
        phi              phi.air;
        inletValue       uniform 1;
        value            uniform 1;
    }
    walls
    {
        type             zeroGradient;
    }
}

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

Listing B.13: 0/alpha.air

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions    [0 0 0 0 0 0 0];

internalField    uniform 0;

boundaryField
{
    inlet
    {
        type            groovyBC;
        value           uniform 0;
        valueExpression "(pos().y > 0.0122965) ? 1 : 0";
    }
    outlet_left
    {
        type            inletOutlet;
        phi             phi.water;
        inletValue      uniform 0;
        value           uniform 0;
    }
    outlet_right
    {
        type            inletOutlet;
        phi             phi.water;
        inletValue      uniform 0;
        value           uniform 0;
    }
    walls
    {
        type            zeroGradient;
    }
}

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

Listing B.14: 0/alpha.water

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    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
    {
  type              calculated;
        value           $internalField;
    }

    outlet_left
    {
        type            calculated;
        value           $internalField;
    }
    outlet_right
    {
        type            calculated;
        value           $internalField;
    }
    walls
    {
        type            nutkWallFunction;
        value           $internalField;
    }

}

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

Listing B.15: 0/nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type    fixedFluxPressure;
        value   $internalField;
    }
    outlet_left
    {
        type    prghPressure;
        p       $internalField;
        value   $internalField;
    }
    outlet_right
    {
        type    prghPressure;
        p       uniform 0;
        value   uniform 0;
    }
    walls
    {
        type    fixedFluxPressure;
        value   $internalField;
    }
}

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

Listing B.16: 0/p_rgh

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U.air;
}
```

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

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

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type                flowRateInletVelocity;
        volumetricFlowRate constant 0.021946554;
        value               uniform (0 0 0);
    }
    outlet_left
    {
        type                pressureInletOutletVelocity;
        phi                 phi.air;
        value               uniform (0 0 0);
        inletValue          uniform (0 0 0);
    }
    outlet_right
    {
        type                pressureInletOutletVelocity;
        phi                 phi.air;
        value               uniform (0 0 0);
        inletValue          uniform (0 0 0);
    }
    walls
    {
        type                fixedValue;
        value               uniform (0 0 0);
    }
}

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

Listing B.17: 0/U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type                flowRateInletVelocity;
        volumetricFlowRate constant 0.002611504;
        value               uniform (0 0 0);
    }
    outlet_left
    {
        type                pressureInletOutletVelocity;
        phi                 phi.air;
        value               uniform (0 0 0);
        inletValue          uniform (0 0 0);
    }
    outlet_right
    {
        type                pressureInletOutletVelocity;
        phi                 phi.air;
        value               uniform (0 0 0);
        inletValue          uniform (0 0 0);
    }
    walls
    {
        type                fixedValue;
        value               uniform (0 0 0);
    }
}

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

Listing B.18: 0/U

B.3.2 constant folder multiphaseEulerFoam

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
  version   2.0;
  format    ascii;
  class   uniformDimensionedVectorField;
  location  "constant";
  object    g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

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

Listing B.19: constant/g

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases
(
    water
    {
        nu        1.301e-6;
        kappa     0;
        Cp        0;
        rho       999;

        diameterModel    constant;
        constantCoeffs
        {
            d     3e-4;
        }
    }
    air
    {
        nu        1.424e-5;
        kappa     0;
        Cp        0;
        rho       1.25;

        diameterModel    constant;
        constantCoeffs
        {
            d     2e-3;
        }
    }
);

sigmas
(
    (air water)   0.0742
);

interfaceCompression
(
    (air water)   1
);

virtualMass
(
    (air water)   0.5
);

lift (
    (air water)   0.5
);

drag
(
    (water air)
    {
       type blended;
       air
       {
          type SchillerNaumann;
          residualPhaseFraction 0;
          residualSlip 0;
       }
       water
       {
          type SchillerNaumann;
          residualPhaseFraction 0;
          residualSlip 0;
       }
       residualPhaseFraction 1e-3;
       residualSlip 1e-3;
    }
);

// This is a dummy to support the Smagorinsky model
transportModel    Newtonian;
nu                [0 2 -1 0 0 0 0]    0;

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

Listing B.20: constant/transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType      LES;

LES {
    LESModel        Smagorinsky;
    turbulence      on;
    printCoeffs     on;

    delta           cubeRootVol;

    cubeRootVolCoeffs
    {
    }
}
// ************************************************************************* //
```

Listing B.21: constant/turbulenceProperties

B.3.3 system folder multiphaseEulerFoam

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


libs (
    "libOpenFOAM.so"
    "libsimpleSwakFunctionObjects.so"
    "libswakFunctionObjects.so"
    "libgroovyBC.so"
);

functions
{
    #include "sampledSurf"
    #include "sampledSets"
}

application         multiphaseEulerFoam;

startFrom           latestTime;

startTime           0;

stopAt              endTime;

endTime             10;

deltaT              1e-5;

writeControl        adjustableRunTime;

writeInterval       0.01;

purgeWrite          0;

writeFormat         ascii;

writePrecision      6;

writeCompression    on;

timeFormat          general;

timePrecision       6;

runTimeModifiable   no;

adjustTimeStep      yes;

maxCo               0.25;

maxDeltaT           1;

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

Listing B.22: system/controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    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
{
    default                        none;

    div(Rc)                        Gauss linear;

    "div\(phi,alpha.*\)"           Gauss vanLeer;
    "div\(phir,alpha.*,alpha.*\)"  Gauss vanLeer;

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

laplacianSchemes
{
    default                Gauss linear corrected;
}

interpolationSchemes
{
    default                linear;
}

snGradSchemes
{
    default                corrected;
}

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

Listing B.23: system/fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

```

```
solvers
{
    "alpha.*"
    {
        nAlphaSubCycles        1;
    }
    p_rgh
    {
        solver                 GAMG;
        tolerance              1e-7;
        relTol                 0;
        smoother               DICGaussSeidel;

        nCellsInCoarsestLevel  200;
    }
    p_rghFinal
    {
        $p_rgh;
    }
    pcorr
    {
        $p_rgh;
        tolerance              1e-8;
        relTol                 0;
    }
}

PIMPLE
{
    nOuterCorrectors           1;
    nCorrectors                2;
    nNonOrthogonalCorrectors   0;
}

relaxationFactors
{

}

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

Listing B.24: system/fvSolution